sid.inpe.br/mtc-m19/2011/12.05.20.15-TDI

APLICAÇÃO DOS CONCEITOS DO SMP2 PARA COMUNICAÇÃO ENTRE SISTEMAS LEGADOS DE SIMULAÇÃO

Miguel Adrian Carretero

Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Gerenciamento de Sistemas Espaciais, orientada pela Dra. Ana Maria Ambrosio, aprovada em 29 de fevereiro de 2012.

URL do documento original:
<http://urlib.net/8JMKD3MGP7W/3ATMEH2>

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-m19/2011/12.05.20.15-TDI

APLICAÇÃO DOS CONCEITOS DO SMP2 PARA COMUNICAÇÃO ENTRE SISTEMAS LEGADOS DE SIMULAÇÃO

Miguel Adrian Carretero

Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Gerenciamento de Sistemas Espaciais, orientada pela Dra. Ana Maria Ambrosio, aprovada em 29 de fevereiro de 2012.

URL do documento original:
<http://urlib.net/8JMKD3MGP7W/3ATMEH2>

INPE
São José dos Campos
2012

Dados Internacionais de Catalogação na Publicação (CIP)

Carretero, Miguel Adrian.
C233a Aplicação dos conceitos do SMP2 para comunicação entre sistemas legados de simulação / Miguel Adrian Carretero. – São José dos Campos : INPE, 2012.
xxvi + 117 p. ; (sid.inpe.br/mtc-m19/2011/12.05.20.15-TDI)

Dissertação (Mestrado em Gerenciamento de Sistemas Espaciais) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2012.
Orientadora : Dra. Ana Maria Ambrosio.

1. simulação. 2. modelos. 3. sistemas. 4. satélite artificial. 5. legado. I.Título.

CDU 629.7:004.75

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

Dr. Marcelo Lopes de Oliveira e Souza

Presidente / INPE / SJCampos - SP

Dra. Ana Maria Ambrosio

Orientador(a) / INPE / São José dos Campos - SP

Dr. Mauricio Gonçalves Vieira Ferreira

Membro da Banca / INPE / SJCampos - SP

Dr. Leonel Fernando Perondi

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

Dr. Gilberto da Cunha Trivelato

Convidado(a) / MECTRON / São José dos Campos - SP

Este trabalho foi aprovado por:

( ) maioria simples

(X) unanimidade

Aluno (a): Miguel Adrian Carretero

São José dos Campos, 29 de fevereiro de 2012

"Todas as grandes coisas são simples. E muitas podem ser expressas numa só palavra: liberdade; justiça; honra; dever; piedade; esperança".

Winston Churchill

"A grandeza não consiste em receber honras, mas merecê-las".

Jean de La Fontaine

Aos meus pais, familiares e amigos que junto comigo trilharam no caminho do saber e os quais são o grande alicerce para a conclusão deste trabalho. E que tudo por que passei sirva a muitos dos meus como um exemplo de nunca desistir e sempre perseverar no caminho do grande arquiteto do universo.

AGRADECIMENTOS

A todos os familiares, amigos e colegas que sempre estiveram ao meu lado nos piores e melhores momentos de minha vida. Com deferência especial a minha esposa e filhas as quais são os elos que fortalecem a corrente de uma família.

Aos Dr. Osmar Pinto Junior e Dr. Kleber P. Naccarato pelo apoio pessoal no cumprimento deste trabalho; e aos grandes amigos Mario Baruel e Edson Ribeiro por suas contribuições pessoais, técnicas e pelas pescarias que me deram descanso mental e com isso pude dar prosseguimento às atividades deste trabalho. E principalmente a Dra. Ana Maria Ambrosio que com a sua paciência e tranquilidade possibilitou chegar ao caminho correto para a escrita deste trabalho.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pelo auxílio financeiro.

RESUMO

Um Simulador de Satélites é composto por modelos que representam o comportamento dos vários subsistemas de um satélite podendo ainda, incluir modelo do ambiente espacial e de estações terrenas de recepção de dados. Muitos destes modelos, em especial os do ambiente espacial, geralmente, não sofrem alterações de um simulador para outro, podendo ser reutilizados ao longo de anos, em diferentes simuladores. Entretanto, esta prática de reutilização não tem sido rotineiramente aplicada. Visando melhorar a interoperabilidade e aumentar o reúso neste tipo de aplicação, a Agência Espacial Européia (ESA) definiu o padrão Simulation Model Portability (SMP), Portabilidade de Modelos de Simulação, que orienta a estruturação de Simuladores de Satélite. O objetivo dessa dissertação é explorar a aplicabilidade dos conceitos do SMP para viabilizar a comunicação entre modelos legados de simulação. Identificados os limites na arquitetura atual do SMP foi proposta uma extensão aos mecanismos de comunicação desta arquitetura de modo a permitir a inclusão dos modelos legados a um simulador que seja conforme o SMP. Uma demonstração da solução dada ao mecanismo de comunicação proposto é feita através de um estudo de caso. Essa dissertação mostra uma solução para aplicação dos mecanismos para interoperabilidade entre modelos legados de simulação se beneficiando da padronização para reforçar o reúso de modelos e de mecanismos que dão suporte a um simulador. Entretanto, a solução ainda mantém a dependência da plataforma computacional e de uma linguagem orientada a objetos.

APPLICATION OF SMP2 CONCEPTS TO THE COMMUNICATION AMONG LEGACY SIMULATION SYSTEMS

ABSTRACT

A Satellite Simulator is composed of a set of models that represents the behavior of the various satellite subsystems. It may also include a model of the space environment and of the ground stations. Many of these models, especially the space environment model do not change from one simulator to another and can be reused over the years. However, this is not a common practice. In order to improve interoperability and increase re-use of the models, the European Space Agency (ESA) created a standard named Simulation Model Portability (SMP) that guides the development of Satellite Simulators. The objective of this work is to explore the applicability of the concepts of SMP to enable communication between legacy simulation models. After identifying limitations in the current SMP architecture it was proposed an extension of the SMP communication mechanisms to allow the addition of legacy models to a SMP simulator. A case study was used to demonstrate the results of the proposed communication mechanism. This work also presents a computational implementation of the interoperability mechanisms between the legacy simulation models (which takes advantage from standardization) to enhance the reuse of models and mechanisms that support a simulator system. However, this solution still preserves its dependence on the computing platform and an on an object-oriented language.

LISTA DE FIGURAS

Pag.

LISTA DE TABELAS

Pag.

LISTA DE SIGLAS E ABREVIATURAS

CBERS	-	*Satélite Sino-Brasileiro de Recursos Naturais (Modelos 1, 2, 2B)(China-Brazil Earth Resources Satellite)*
CORBA	-	*Common Object Request Broker Architecture*
CTOS	-	*Commercial Off-The Shelf*
DIS	-	*Distributed Interactive Simulation*
DMSO	-	*Defense Modeling and Simulation Office*
DoD	-	*Departament of Defense*
ESA	-	*European Space Agency*
FBMSIM	-	*French-Brazilian Micro-satellite Simulator*
HLA	-	*High Level Architecture*
IIOP	-	*Internet Inter-ORB Protocol*
INPE	-	*Instituto Nacional de Pesquisas Espaciais*
MDA	-	*Model Driven Architecture*
MDK	-	*Model Development Kit*
OMG	-	*Object Management Group*
PCD	-	*Plataforma de Coleta de Dados*
PIM	-	*Platform Independent Model*
PSM	-	*Platform Specific Model*
SCD	-	*Satélite de Coleta de Dados (modelo 1, 2 2A)*
SIMC	-	*Simulator CBERS*
SIMS	-	*Simulator SCD*
SMDL	-	*Simulation Model Definition Language*
SMP	-	*Simulation Model Portability*
UML	-	*Unified Modeling Language*
XMI	-	*XML Metadata Interchange*
XML	-	*Extensive Markup Language*
XSD	-	*XML Schema Definition*

SUMÁRIO

Pág.

CAPÍTULO 1

INTRODUÇÃO

"*Não se faz ciência sem registrar o que se aprende.*"
(Dante Alighieri).

1.1 Contexto e motivação

Simulação é o processo de projetar um modelo do sistema real e conduzir experimentos com tal modelo com o propósito de entender o comportamento do sistema ou avaliar as várias estratégias, dentro dos limites impostos por um critério ou conjunto de critérios para a operação. (SHANNON, 1975) Quando a experimentação para obter as informações com o sistema real é impossível ou indesejável, pelo alto custo ou risco, é conveniente o uso da simulação.

O uso de simulação em computadores busca criar uma representação de ambientes físicos, ações de pessoas e equipamentos em modelos que possam ser testados e avaliados. Um modelo é uma representação de um objeto, um sistema ou uma ideia de alguma outra forma que não a própria entidade. (GORDON, 1969)

Um sistema de simulação corresponde a um conjunto de programas computacionais que foram projetados para um trabalho específico e que podem ter sido desenvolvidos em diferentes linguagens de programação, executados em diferentes plataformas e utilizarem formatos de configuração de dados distintos.

Os programas computacionais que, apesar de serem antigos, ainda fornecem serviços essenciais são referenciados como sistemas legados. Normalmente, são aplicações cuja manutenção é raramente requerida, de alta confiabilidade, podem representar um alto custo de modernização e como satisfazem a necessidade do cliente, continuam ativas.

A interoperabilidade é a capacidade de sistemas se comunicarem independentemente do fabricante ou da tecnologia usada no desenvolvimento do sistema. A característica de

interoperabilidade permite a produção de soluções flexíveis e com qualidade já que a comunicação entre sistemas diferentes não representa um obstáculo para um projeto. (SILVA, 2006)

Um simulador de satélite, nesta dissertação, diz respeito "a um sistema computacional que auxilia o treinamento de operadores em procedimentos rotineiros e contingências, bem como, o planejamento das operações da missão espacial." (RAINEY, 2004).

Mirshams (2010) afirma que a simulação é um dos melhores métodos para a educação e compreensão dos eventos utilizados em muitos campos da ciência, como exemplo as pesquisas espaciais. E explica em parte do seu trabalho, como o desenvolvimento de um simulador sobre uma ferramenta de engenharia traz resultados para validação sobre os componentes e os modelos para os simuladores de satélites.

Tais simuladores de satélite, comumente, lidam com questões relacionadas a sistemas legados devido ao alto grau de comonalidade entre os requisitos de uma missão espacial para outra. Reggestad et al. (2004) comparam as características dos requisitos entre os vários simuladores desenvolvidos pela Agência Espacial Européia, do Inglês *European Space Agency* (ESA). A conclusão é que 70% dos requisitos são comuns para todos os simuladores comparados. Este valor reforça a ideia de estabilidade, não somente nos conjuntos de requisitos, mas também no reúso da arquitetura da simulação.

Visando incentivar o reúso e facilitar a interoperabilidade nos sistemas de simulação de satélites a *European Space Agency* (ESA) definiu um conjunto de memorandos técnicos chamado de SMP, do Inglês *Simulation Model Portability* (SMP), contendo especificações para as interfaces dos modelos e uma arquitetura de interoperabilidade, especificações para codificação, uso de interfaces e geração de documentação, reunidas no chamado padrão SMP. O objetivo é melhorar a padronização no desenvolvimento de simuladores de satélites. Entre suas especificações está a de um formato padronizado das interfaces para troca de mensagens entre os modelos de um simulador.

A mais recente referência que buscou capturar as melhores práticas para desenvolvimento de modelos e simulação o ECSS (2010) apresenta o SMP como parte importante na padronização das interfaces de modelos.

O tema desta dissertação foi motivado primeiro, pela possibilidade de se aproveitar modelos e mecanismos de simulação desenvolvidos por diferentes grupos no INPE. Podemos citar alguns exemplos de trabalhos em simulação desenvolvidos no INPE, como as implementações do modelo J77 da alta atmosfera aplicadas em propagadores de órbita e atitude para previsão de passagens realizadas por Carrara (1990); um histórico dos simuladores de satélite desenvolvidos para dar apoio às atividades do Centro de Controle de Satélites do INPE é apresentado em Ambrosio et al. (2006); Hoffman e Perondi (2010) descrevem um mecanismo para escalonamento de tarefas de um simulador para a missão CBERS3; modelos para manobra de rendezvous (encontro) e docking (acoplamento) para satélites artificiais foram desenvolvidos por Gentina (2011). Em Kuga et al. (2011) apresentam-se conceitos de vários modelos do ambiente espacial. Outro fato motivador para o desenvolvimento deste trabalho é a crescente necessidade de novos simuladores operacionais para atender a demanda de novos satélites a serem desenvolvidos pelo INPE como proposto por Tominaga (2010).

1.2 Objetivo e relevância do trabalho

O objetivo dessa dissertação é explorar a aplicabilidade dos conceitos do SMP para viabilizar a comunicação entre modelos legados de simulação, além disso, entender as vantagens e os limites dos conceitos do SMP como padrão para o desenvolvimento de simuladores de satélite.

A escolha do padrão SMP foi motivada pela sua promessa de promover o reúso e a interoperabilidade em sistemas de simulação de satélites. Para alcançar o objetivo dessa dissertação, foi feito um estudo aprofundado sobre as especificações do SMP, a sua aplicação nas interfaces a serem usadas nos sistemas, ambientes e modelos legados de simulação e nas soluções para comunicação de sua arquitetura para interoperabilidade.

A importância deste trabalho é a de prover uma base para que modelos existentes, desenvolvidos por diferentes grupos do INPE ou por terceiros, possam ser usados para compor futuros simuladores, seja para apoio a operação dos mais variados satélites no INPE, seja para análise do comportamento parcial de satélites. A tecnologia usada na solução e programação dos modelos faz com que as modificações para a outra plataforma de execução possa ser trabalhosa e custosa, o que impede o seu reúso. O aproveitamento dos programas computacionais legados poderá oferecer um potencial de agilidade no reúso e uma economia em tempo e recursos financeiros no desenvolvimento de novos projetos de simuladores.

1.3 Metodologia aplicada ao trabalho

O presente trabalho consta de uma pesquisa científica aplicada, uma vez que tem como objetivo a geração de conhecimento para aplicação prática dirigida à solução de problemas específicos (SILVA; MENEZES, 2001).

Foram estudados os conceitos básicos em sistemas distribuídos, simulação de satélites e os memorandos técnicos do padrão SMP. O conhecimento científico foi adquirido com a leitura de artigos científicos.

Um protótipo de um simulador de satélites simplificado foi desenvolvido para demonstrar as facilidades de interoperabilidade entre modelos legados de simulação. Como prova de conceito do mecanismo de comunicação aplicado para comunicação entre modelos legados de um simulador, foram usados dois modelos típicos de simuladores de satélites existentes, um programado em linguagem Fortran99 e outro em linguagem Pascal. O protótipo serviu para consolidar os conhecimentos em SMP e em simuladores e mostrar também a possibilidade dos modelos estarem distribuídos, graças à extensão do SMP proposta nesta dissertação.

1.4 Organização do trabalho

O texto desta dissertação está organizado na sequência de Capítulos conforme apresentado a seguir.

No Capítulo 2 são apresentados alguns conceitos básicos sobre sistemas, simulação e a revisão bibliográfica.

O Capítulo 3 apresenta aspectos importantes do padrão SMP, mostrando como ele pode ser usado para a interoperabilidade em modelos legados de simulação.

O Capítulo 4 apresenta a proposta para comunicação e interoperabilidade entre sistemas legados de simulação em conformidade com o formato padronizado nas especificações do SMP.

O Capítulo 5 desenvolve uma prova de conceito para demonstrar a solução da proposta evidenciando a comunicação entre modelos e um simulador com base nas especificações em SMP, através da implementação de um simulador simplificado.

O Capítulo 6 apresenta a conclusão, lições aprendidas e propostas de trabalhos futuros com base no aprendizado resultante da aplicação dos conceitos de comunicação do SMP.

O Apêndice A lista em formato texto os vários arquivos que foram gerados e usados no estudo de caso desta dissertação.

CAPÍTULO 2

REVISÃO BIBLIOGRÁFICA

"Estudar é como remar contra a corrente; se não se fazem progressos, ande-se para trás."

(Provérbio Chinês)

Neste Capítulo são abordados os conceitos utilizados no desenvolvimento dessa dissertação, tais como, sistemas computacionais, sistemas legados, arquiteturas distribuídas, distribuição de processamento, interoperabilidade e o padrão CORBA, simuladores de satélite e finalmente, uma referência mostrando a experiência em desenvolvimento de simuladores de satélites pelo INPE.

2.1 Sistemas computacionais e sistemas legados

Um *sistema* (do grego *sietemiun*) é um conjunto de elementos interconectados, de modo a formar um todo organizado. Vindo do grego o termo "*sistema*" significa "*combinar*", "*ajustar*", "*formar um conjunto*". Todo sistema possuí um objetivo geral a ser atingido. O sistema é um conjunto de órgãos funcionais, componentes, entidades, partes ou elementos. A integração entre esses componentes pode se dar por interagentes, fluxo de matéria, fluxo de sangue, fluxo de energia, entre outros, ou seja, há comunicação entre os órgãos componentes de um sistema.

Um sistema, o de computação neste caso, não é representado somente pelos componentes físicos (CPU, entradas e saídas, periféricos, etc.), operacional (sistema operacional, programas do sistema operacional, arquivos, etc.) e aplicativo (programas, jogos, etc.), mas também pelos componentes que trafegam por sua periferia como operadores, usuários, gerentes, administradores ou qualquer elemento (físico ou lógico) que necessite interagir com o ambiente (NEVES et al., 1997). Os sistemas computacionais representam um conjunto de elementos que são criados, desenvolvidos

e precisam ser mantidos em funcionamento por um longo período de tempo, pois continuam trazendo, os benefícios que se esperam deles.

Os sistemas computacionais podem ser executados em um único processador ou ter seu processamento distribuído. Quando distribuídos requerem protocolos para comunicação e a definição de uma arquitetura. Tais sistemas podem se comunicar entre si através de uma rede de comunicação de dados onde somente trafegam os dados; ou utilizar a rede de comunicação para a troca de informações e de entidades, neste caso diz-se que há interoperabilidade entre os sistemas comunicantes.

Ao perguntar a algum usuário, gerente, programador ou a qualquer indivíduo que faça parte ou use a Informática como meio de suas atividades "o que é um sistema legado?", teremos como resposta: é aquele que funciona bem há muito tempo. Por esta resposta tem-se uma visão simplista de algo muito mais abrangente e complexo para a era baseada na informação digital. Estes sistemas computacionais que continuam fornecendo os serviços essenciais por um longo período de tempo, apesar das constantes evoluções da computação, são denominados de sistemas legados. Em geral são aplicações ou programas que são fundamentais para a operação de muitas empresas que, apesar de antigos, são fundamentais e imprescindíveis ao seu dia-a-dia (SOMMERVILLE, 2007). "A expressão sistema legado (do Inglês "*legacy system*") passa a ideia de uso de uma tecnologia ultrapassada ou limitada" (GANDIN, 2003).

Penteado et al. (1995) afirmam que os sistemas legados são aqueles sistemas que estão em uso por muito tempo e que atendem aos requisitos dos usuários. Além disso, eles são de difícil substituição por que a sua reimplementação é inviável financeira ou técnicamente; e por serem imprencidíveis, esses sistemas não podem ficar inativos por muito tempo.

Sistemas legados são encontrados em um número expressivo de organizações de médio e grande porte, estabelecidas há no mínimo uma década, e que têm suas atividades apoiadas por certo nível de informatização operacional.

Estima-se que existam entre 1.500 a 2.500 aplicações desenvolvidas na linguagem COBOL, sendo a 37ª linguagem mais usada nos dias atuais (PAULSON, 2006), segundo pesquisa nas 1.000 maiores empresas americanas listadas na Fortune 1000. A 22ª linguagem mais usada nos dias atuais é o Fortran, segundo o Índice da Comunidade de Programação TIOBE (2011). Essas afirmações mostram a importância de transpor os vários sistemas legados para as novas plataformas tecnológicas, sem grande intrusão.

O INPE possuí um parque de sistemas computacionais administrativos e gerenciais com aproximadamente 80% escrito em linguagem procedural. Na área de pesquisa o uso é de quase 100% pois a linguagem de programação mais usada é o Fortran e alguns casos a linguagem C, em aplicações científicas e em sistemas embarcados usados em experimentos técnicos e em satélites. Esses dados foram obtidos em conversas e levantamentos feitos com representantes do Serviço de Tecnologia da Informação (STI) do INPE.

Muitos dos sistemas legados, de fato, não fazem uso dos novos recursos tecnológicos gerados ao longo dos anos, e por isso, sob certo aspecto, se tornam limitados. Entretanto, há um rol de motivos para que estes sistemas não sejam convertidos para as novas tecnologias, alguns desses motivos são:

a) alto custo financeiro e humano para o desenvolvimento usando tecnologia mais avançada, e para a manutenção do sistema em operação durante a sua renovação;

b) desconhecimento do real impacto das novas tecnologias no sistema resultante da evolução para a nova plataforma;

c) falta de documentação e ausência de pessoal técnico com conhecimento no seu desenvolvimento podendo levar a problemas como: dificuldade de compreensão das regras de negócio neles implementadas, desconhecimento das razões que levaram a determinadas decisões, problemas na estruturação dos módulos de código, miscelânea de estilos de programação, não

obsolescência das ferramentas de desenvolvimento e impossibilidade de reaproveitamento dos equipamentos nos quais são executados os softwares mais atuais;

d) esses sistemas ainda são críticos para os negócios, isto é, eles são essenciais para o funcionamento normal do negócio.

Algumas razões para mantê-los em atividade são:

a) o sistema funciona de forma satisfatória e o interessado não vê razões para alterá-lo;

b) os custos de redesenhar ou substituir o sistema são proibitivos porque é grande, monolítico e/ou complexo;

c) reciclagem de um novo sistema seria cara em tempo e dinheiro, em comparação com os benefícios previstos em substituí-lo (que pode ser do zero);

d) o interresado espera que o sistema possa ser facilmente substituído quando isso se tornar necessário.

Pesquisas e aplicação de conceitos da Engenharia de Software estão promovendo facilidades para introduzir novas tecnologias aos sistemas legados. Atualmente, um universo de transposição e integração de novos componentes pode ser feito sem a perda do funcionamento dos sistemas legados. Um exemplo de migração de um sistema legado é mostrado em Gandin (2003), que tem como principal objetivo a tradução de um sistema legado escrito em COBOL para o PROGRESS com o uso de Engenharia Reversa e Reengenharia sobre o código fonte do sistema para manter o conhecimento adquirido dos sistemas e utilizar os conhecimentos obtidos como uma base de evolução contínua e estruturada do sistema. Santader e Silva (2004) apresentam um processo de evolução dos sistemas legados baseado não somente no conhecimento do código fonte, mas também em outros artefatos como fluxograma, manuais, etc. para facilitar a

elicitação, análise e entendimento dos requisitos que compõem o sistema legado. O processo constitui um conjunto de diretrizes para apoiar os desenvolvedores na evolução de sistemas legados mapeando as informações contidas no Diagrama de Fluxos de Dados (DFD) e código fonte para Casos de Uso em UML.

Pinto e Braga (2005) procuram mostrar que sistemas legados críticos que estão em operação podem ter seu ciclo de vida estendido e ser integrado a novas tecnologias. Apresentam um panorama sobre as tecnologias que podem ser utilizadas para "empacotamento" de um sistema legado. Os autores dizem que a modernização de um sistema legado pode ser feita a partir de duas formas de conhecimento: (i) modificação da lógica interna do sistema, chamada de "modernização *white-box*"; (ii) integração dos sistemas via interfaces externas, chamada de "modernização *black-box*". Esta última permite um aproveitamento do sistema legado com menos esforço, mas requer o uso de mecanismos de comunicação.

2.2 Sistemas distribuídos

Esta seção apresenta conceitos relacionados a sistemas distribuídos e a idéia geral de arquitetura orientada a modelos.

2.2.1 Arquitetutra distribuída

Uma arquitetura distribuída deve possuir os seguintes elementos básicos: linguagem de definição de interfaces, administrador de objetos e serviço de nomes.

A **linguagem de definição de interface** é importante no suporte a aplicações distribuídas porque requerem um nível abstrato de comunicação entre as aplicações. As interfaces são a forma ideal de interoperabilidade entre objetos distribuídos. Em sistemas orientados a objetos, um objeto deve assumir o mínimo possível sobre a programação do método, pois esse pode possuir referência a outro método localizado em outro computador. A interface é um contrato estabelecido entre os objetos e contêm uma lista de métodos que serão disponibilizados para outros objetos.

O **administrador de objetos** é responsável por instanciar os objetos requisitados, passar a referência destes objetos aos clientes e realizar o "*marshalling*" do pedido de objetos de máquinas diferentes. O administrador de objetos esconde os detalhes sobre a localização dos mesmos, tornando indiferente se um método será invocado local ou remotamente.

O **serviço de nomes** é o mecanismo através do qual o servidor informa os clientes sobre os objetos que estão disponíveis para acesso. Conhecendo quais objetos podem ser servidos, os clientes facilmente descobrem a assinatura e os argumentos dos vários métodos que podem ser invocados do objeto. Desta forma, uma aplicação distribuída adquire um comportamento flexível e dinâmico, sendo possível estabelecer a comunicação entre objetos apenas em tempo de execução.

2.2.2 Distribuição do processamento

A tecnologia para distribuição de processamento facilita a construção de ambientes computacionais em larga escala e em plataformas heterogêneas e geograficamente distribuídas. São quatro os principais benefícios do uso da distribuição do processamento:

a) redução do tempo de execução: subdividir a computação em computadores menores e executá-los concorrentemente leva a redução do tempo de execução;

b) distribuição geográfica: a execução de um programa em um conjunto de computadores geograficamente distribuídos permite a criação de mundos virtuais com múltiplos participantes, fisicamente localizados em diferentes locais;

c) integração de diferentes fabricantes reduz os custos de interconexão: cada ambiente concorre ao processamento em seu próprio computador. O que leva a uma computação distribuída entre várias plataformas;

d) tolerância a falhas: a utilização de vários computadores leva a um aumento na tolerância a falhas. Caso um computador ou processador falhar, é possível que outro retome o seu lugar.

2.2.3 Interoperabilidade

A comunicação entre computadores requer a presença de protocolos, como por exemplo, o TCP/IP, que permite a interconexão em rede local ou remota de muitos computadores. Com relação à comunicação entre sistemas computacionais, em um nível de abstração maior, existe o conceito de interoperabilidade.

"A **interoperabilidade** é a possibilidade de sistemas coexistirem e comunicarem-se independente de fabricantes ou tecnologias, permitindo soluções computacionais flexíveis." (SILVA, 2006) Exemplos de arquiteturas de interoperabilidade aplicadas em sistemas são:

a) *Service Oriented Architecture* (SOA): arquitetura com a proposta de interoperabilidade de sistemas por meio de um conjunto de interfaces de serviços fracamente acoplados, altamente coesos e com alta reutilização. Nela, os serviços não necessitam de detalhes técnicos da plataforma de outros serviços para a troca de informações;

b) *Common Object Request Broker Architecture* (CORBA): arquitetura padrão criada pela Object Management Group para estabelecer e simplificar a troca de dados entre sistemas distribuídos heterogêneos. Atua de modo que os objetos (componentes de software) possam se comunicar de forma transparente ao usuário;

c) *Enterprise Service Bus* (ESB): arquitetura de desenvolvimento de sistemas tipicamente baseados em padrões flexíveis para integração de aplicações, suportando vários meios de transportes.

Os **protocolos de interoperabilidade** descrevem uma forma padronizada de interação entre sistemas, exemplos de tais protocolos são:

(i) O ***Aggregate Level Simulation Protocol*** (ALSP): é um protocolo que habilita um sistema de simulação, pode interoperar com outro sistema de simulação e consiste das seguintes características (WEARTHERLY et al., 1996):

a) software de infra-estrutura: chamado *ALSP Infrastructure Software* (AIS) que providencia suporte e administração em tempo de execução da simulação;

b) uma interface em ALSP que possuí um protocolo para troca de mensagens de dados genéricos.

(ii) ***Web Services***: são interfaces independentes de plataforma que permitem a comunicação entre aplicações utilizando as tecnologias existentes na internet (PINTO; BRAGA, 2005). Três padrões-chaves permitem essa comunicação, independente da plataforma na qual estão sendo executadas as aplicações, são elas:

a) *Simple Object Access Protocol* (SOAP): específica um formato para as mensagens passada entre os *Web Services*;

b) *Web Service Description Language* (WSDL): a qual descreve o *web service*, permitindo que outros *Web Services* saibam como acessá-lo, o que enviar de entrada e o que esperar de saída;

c) *Universal Description Discovery and Integration Standard* (UDDI): que é um sistema de registro que permite aos serviços *Web Services* publicarem documentos WSDL, para que outros *Web Services* possam ler esses documentos. A UDDI também fornece a especificação sobre os formatos de entrada de dados, modelos de segurança, protocolos e formatos de saída.

(iii) ***Object Request Borker (***ORB) (OMG, 2011c): é o protocolo de interoperabilidade usado pela arquitetura CORBA. (OMG, 2011b) O CORBA está incluido em muitas plataformas computacionais e faz parte das especificações do SMP.

2.2.4 CORBA

Em 1989 foi formado o Object Management Group (OMG, 2011a) com o intuito de promover a adoção de sistemas distribuídos de objetos a serem usados em sistemas distribuídos e obter vantagens da utilização da programação orientada a objetos no desenvolvimento de aplicações. Em 1991, foi concluída a primeira especificação da arquitetura Common Object Request Broker Architecture (OMG, 2011b) mostrada na Figura 2.1 e aprovada pelo consórcio de empresas que são associadas ao grupo OMG.

Common Object Request Broker (CORBA) como proposto por Siegel (2000) "é uma abordagem para objetos distribuídos recomendada pela Object Management Group" (OMG, 2011a), os componentes que formam o CORBA são:

a) *Object Request Broker* (ORB): permite aos objetos uma transparência no envio e recebimento de solicitações e respostas em um ambiente distribuído;

b) *Common Object Services* (COS): coleção de serviços, interfaces e objetos, que fornece funções básicas para o uso de objetos, esses serviços são necessários para a construção de aplicações que serão distribuídas e necessitam ser independentes dos domínios da aplicação;

c) *Common Facilities* (CF): conjunto de serviços que as aplicações podem compartilhar, mas que não são fundamentais;

d) *Application Objects* (AO): produtos que são executados em cima da arquitetura CORBA. Objetos de aplicação correspondem aos conceitos tradicionais de sistemas e eles não são padronizados pelo OMG.

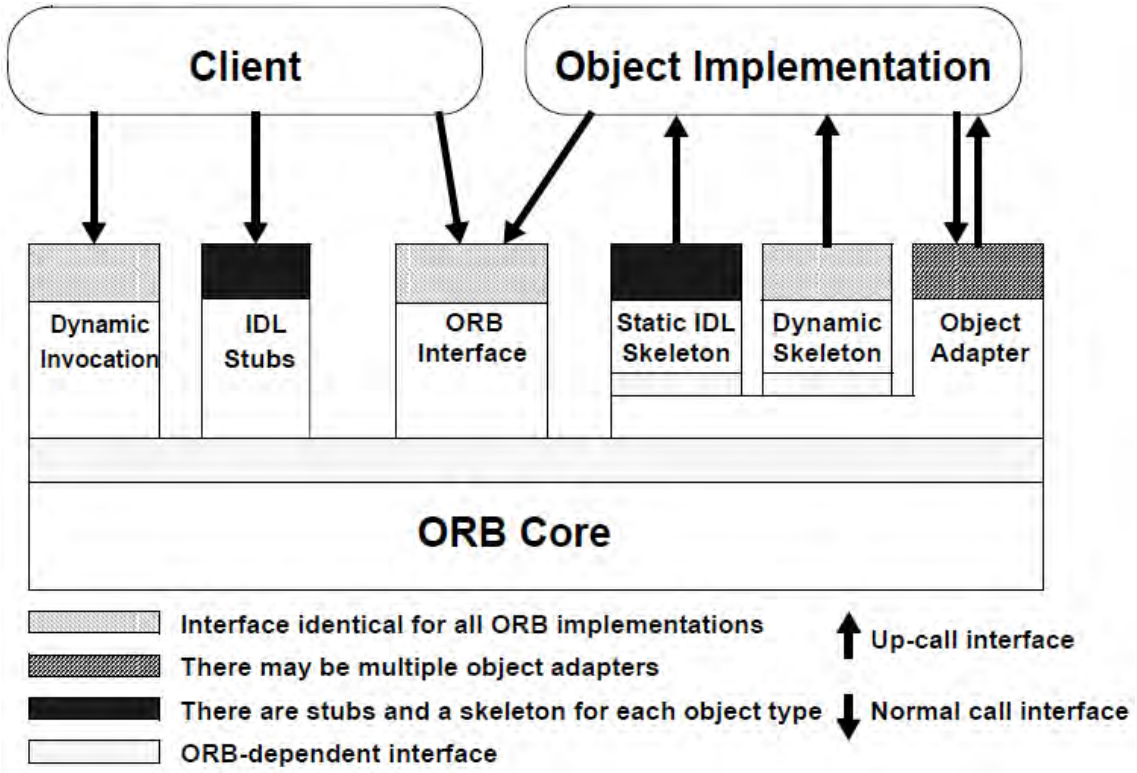


Figura 2.1 Arquitetura do CORBA.

Fonte: OMG. (2011b)

A linguagem de interfaces usada pelo CORBA chama-se *Interface Definition Language* - IDL. (OMG, 2011b) A IDL é puramente declarativa e a sua sintaxe é praticamente igual à da linguagem de programação ANSI/ISO C++, só que a IDL apenas define as interfaces e não a sua codificação, sendo independente da linguagem de programação. A IDL possibilita a interoperabilidade entre os diversos sistemas, devido à separação entre a definição da interface e a execução. A interface de cada objeto é definida de forma bastante específica, enquanto a sua execução permanece oculta para o resto do sistema. (VINOSKI, 1997)

O protocolo responsável pela administração de objetos é o *Object Request Broker* (ORB) que é visto como um middleware da especificação CORBA. Trata-se de um conjunto de módulos de software que gerenciam a comunicação entre os objetos, sendo

apresentado na literatura do CORBA como sendo o *barramento de objetos* que liga os componentes. O grande potencial desse *barramento* está em permitir que os objetos façam requisições, de forma transparente, a outros objetos que podem estar localizados local ou remotamente. Essa transparência assegura que o requisitante não tenha conhecimento dos mecanismos utilizados para comunicação com o objeto desejado.

O serviço de nomes do CORBA é o mecanismo chamado de *Dynamic Invocation Interface* (DII) (OMG, 2011b), que evita a utilização de definição de interfaces pré-compiladas. A invocação dinâmica caracteriza-se por permitir que os requisitantes possam dinamicamente construir e invocar requisições a objetos. O DII é uma interface que permite um acesso direto aos mecanismos providos por um ORB, descobre ou constrói dinamicamente uma invocação a um objeto, permitindo a inclusão do mesmo pelo cliente, sem que seja necessário um conhecimento prévio de quais objetos ou métodos estão disponibilizados. Todas as informações necessárias para que o requisitante possa realizar o processo de invocação, são encontradas no Repositório de Interfaces. Estes repositórios são pesquisados para localizar as interfaces que interessam ao requisitante. Em uma rede local a interoperabilidade entre clientes é transparente, pois o ORB, instalado em uma máquina, é capaz de descobrir todos os outros ORBs, instalados nas demais máquinas. O protocolo chamado *Inter-ORB Protocol* (IIOP), que está baseado sobre o TCP/IP, é usado para a interoperabilidade em redes remotas.

O CORBA permite que o middleware seja programado na linguagem mais conveniente, C, C++, Pascal, etc., e que interaja com qualquer cliente CORBA. As arquiteturas são genéricas para aplicações distribuídas, mas a comunicação entre os objetos é complexa, pois suporta a comunicação direta entre os objetos instanciados em plataformas diferentes. Como uma arquitetura de padrão aberto não está restrita a uma única linguagem, ela pode servir para integrar *sistemas legados* com sistemas novos que estejam implementadas em outras linguagens.

2.2.5 Arquitetura orientada a modelos

O conceito de arquitetura orientada a modelos também foi usado nesta dissertação. A arquitetura conhecida como arquitetura dirigida a modelos, do Inglês *Model Driven Architecture* (MDA) (KLEPPER et al., 2003; OMG, 2011d), proposta pelo Object Management Group (OMG, 2011a), é uma especificação para apoiar o desenvolvimento dirigido a modelos. A sua arquitetura compreende três passos principais:

a) *Platform Independent Model* (PIM): declaração do modelo com um alto grau de abstração, independente de qualquer plataforma tecnológica;

b) *Platform Specific Models* (PSM): geração das declarações do PIM para uma determinada plataforma tecnológica;

c) geração de código: o código fonte gerado não só declaras as estruturas básicas declaradas no PIM e transformadas em PSM, mas deve ser o mais próximo possível da solução definitiva do modelo.

A Figura 2.2 representa os passos da arquitetura MDA.

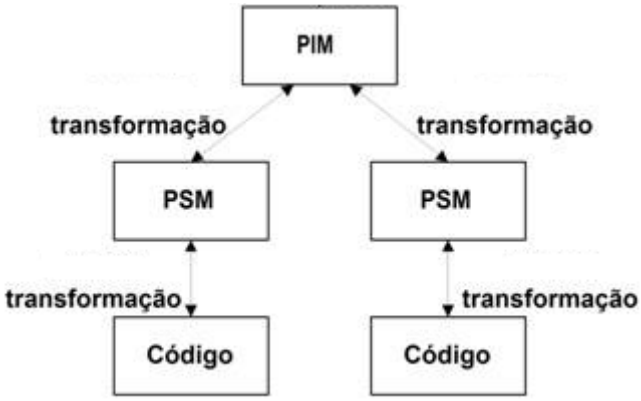


Figura 2.2 Arquitetura MDA.

A MDA define as transformações entre os modelos. Uma transformação é uma geração automática de um modelo em outro, conforme uma especificação de transformação, que abrange um conjunto de regras de transformação que descrevem como um modelo pode

ser transformado em outro. Um modelo descrito em PIM pode ser transformado em um ou mais modelos em PSMs, que por sua vez, são transformados em código.

2.3 Simulação

Simulação, segundo (SHANNON, 1975), é "*o processo de projetar um modelo de um sistema real e conduzir experimentos com tal modelo com o propósito de entender o comportamento do sistema ou avaliar várias estratégias, dentro dos limites impostos por um critério ou conjuntos de critérios, para a operação de um sistema*".

"*Um modelo é uma representação de um objeto, sistema ou ideia de alguma outra forma que não o da própria entidade*" (GORDON, 1969), os modelos podem ser confeccionados de várias formas (computacional, matemáticos, probabilísticos e estatísticos). Quando se trata de modelos algumas regras no desenvolvimento devem ser avaliadas, como o grau de fidelidade, o nível de agregação e a relevância em relação ao ambiente em que serão simulados.

Um simulador, geralmente, é um sistema computacional (incluindo algoritmo, um processador único ou uma rede de processadores) capaz de executar um (ou mais) modelo para reproduzir o seu(s) comportamento(s). (RAINEY, 2004) No simulador os modelos são agregados e interagem ou possuem interdependência para atingir entre si certos objetivos. Todo simulador deve ter uma interface simples e intuitiva, para que um condutor da simulação tenha como principal atenção os modelos simulados.

A simulação permite fazer muitas experiências, sem os riscos e os custos de uma experiência real, pois tudo ocorre no mundo virtual e de forma controlada. Como afirma Gavira (2003), nos últimos anos a simulação computacional vem assumindo uma importância cada vez maior como ferramenta de aquisição de conhecimento. Pois na medida em que a complexidade dos problemas cresce, aumenta a necessidade de se utilizar uma abordagem mais sistêmica e generalista.

2.3.1 Arquiteturas para simuladores

Existem hoje muitas arquiteturas e padrões para simulação, dentre os quais, segundo Souza e Trivelato (2008), citado por Reis (2009), as mais relevantes são:

- *Advanced Distributed Simulation* (ADS);
- *Distributed Interactive Simulation* (DIS);
- *High Level Architecture* (HLA);
- *Test and Training Enabling Architecture* (TENA).

As subseções a seguir trazem um resumo das principais características das arquiteturas citadas acima.

2.3.1.1 Advanced Distributed Simulation – ADS

A Simulação Disitribuída Avançada, do Inglês *Advanced Distributed Simulation* (ADS) é a denominação definida pelo Departamento de Defesa Americano, do Inglês, *Department of Defense* (DoD), para descrever a utilização cooperativa de simulações fisicamente distribuídas com um objetivo comum. O protocolo *Aggregate Level Simulation Protocol* (ALSP) define os mecanismos para a integração de simulações existentes segundo esta arquitetura. A principal motivação para a criação da ADS foi possibilitar a realização de exercícios de cenário tático. (WEATHERLY et al., 1996)

2.3.1.2 Distributed Interactive Simulation - DIS

A Simulação Distribuída Interativa, do Inglês *Distributed Interactive Simulation* (DIS), é um padrão para conduzir uma simulação em tempo real sobre uma grade de computadores ligados em rede de comunicação. A definição deste padrão foi patrocinada pelo Ministério da Defesa dos EUA (DMSO, 1998) e hoje é um padrão. (IEEE, 1998) A simulação distribuída interativa faz uso de um protocolo específico para sua implementação. O protocolo permite a interoperabilidade entre ambientes de simulação heterogêneos, distribuídos geograficamente para trabalharem juntos e

interagirem em tempo real, provendo um único simulador integrado. O protocolo baseia-se em padrões abertos de desenvolvimento e foi definido com a participação de governo, indústria, institutos de pesquisa, etc., com apoio da Organização de padrões da simulação e da interoperabilidade (SISO, 2011), estando a sua especificação na versão 7.

Nesta arquitetura não há um servidor central, a arquitetura é descentralizada. Cada máquina controla e manuseia alguns modelos da simulação específicos. As simulações são autônomas, ou seja, a responsabilidade de manter a situação das entidades e objetos que estão sendo simulados é do ambiente de simulação.

2.3.1.3 High Level Architecture - HLA

O Ministério da Defesa dos EUA há muitos anos tem investido na definição de uma arquitetura de propósito geral, distribuída, para sistemas de simulação.

O maior avanço na tecnologia de simulação interativa distribuída foi dado no plano denominado "*Modeling and Simulation Master Plan*" publicado em 1995, que descrevia seis objetivos, identificava as necessidades chaves e as respectivas ações para criação de um simulador. Os seis objetivos são: (i) desenvolver um arcabouço técnico comum para modelagem e simulação, (ii) prover oportunamente representações de autoridades do ambiente natural, (iii) prover representações de autoridades de sistemas, (iv) prover representações de autoridades do comportamento humano, (v) estabelecer um arcabouço de modelagem e simulação para satisfazer as necessidades do desenvolvedor e dos usuários finais, (vi) compartilhar os benefícios de modelagem e simulação. (DMSO, 1995) Um quadro com tais objetivos é mostrado na Figura 2.3.

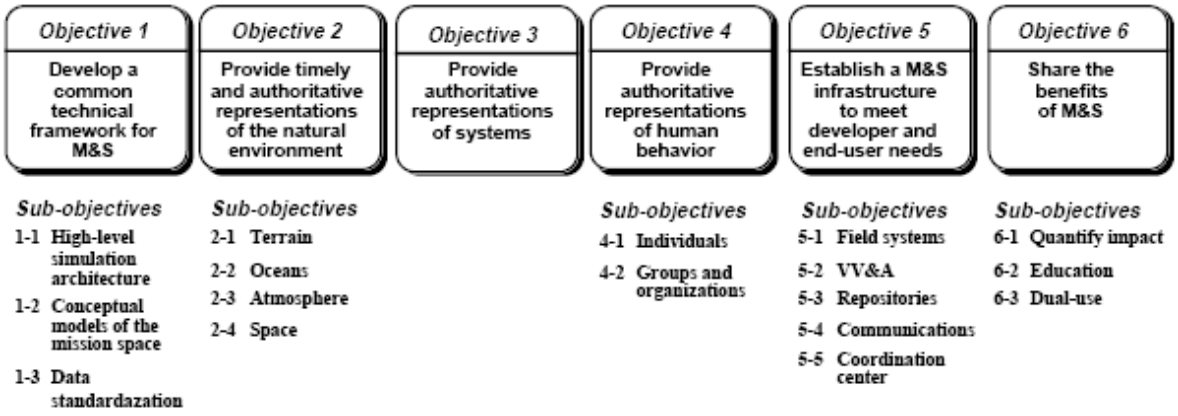


Figura 2.3 Objetivos do Modeling and Simulation (M&S) Master Plan.

Fonte: DMSO (1995)

HLA (DMSO, 2011) é uma arquitetura de propósito geral para reutilização e interoperabilidade de simulações. “A HLA permite o desenvolvimento de simulação distribuída baseada em componentes visando interoperabilidade e o reuso” (KUHL et al., 1999). Existem várias implementações da HLA: algumas delas primam pelo desempenho em detrimento ao suporte à distribuição do processamento; outras primam pelo suporte à distribuição do processamento com algum compromisso com relação ao desempenho (REIS, 2009).

A HLA tornou-se o padrão, do IEEE (*Institute of Electrical and Eletronic Engineers*) a partir do ano 2000, descrito nas normas IEEE 1516 (IEEE, 2000a; IEEE 2000b; IEEE 2000c).

A arquitetura da HLA, Figura 2.4, não traz informações sobre a construção e o conteúdo dos objetos modelados, mas requer que toda a simulação siga um processo de desenvolvimento em conformidade com a documentação do padrão HLA. (JUDITH et al., 1997) A HLA é confirmada como padrão industrial, usa processo aberto de desenvolvimento competitivo, porque provê uma arquitetura única que cobre tanto simulações analíticas como simulações de ambientes virtuais, permitindo o seu uso em várias áreas.

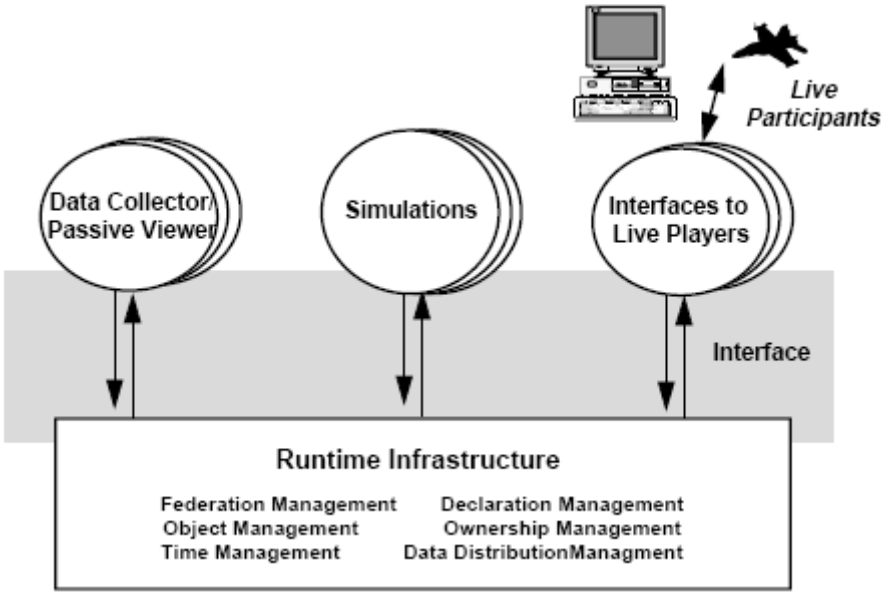


Figura 2.4 Arquitetura HLA uma visão funcional.

Fonte: JUDITH. (1998)

2.3.1.4 Test and Training Enabling Architecture – TENA

A arquitetura de treinamento e teste, do Inglês *Test and Training Enabling Architecture* (TENA), é uma arquitetura que tem sido progressivamente introduzida nas forças armadas norte-americanas para possibilitar a integração de sensores, redes e outros sistemas de hardware e software (ADSO, 2004).

Esta arquitetura possibilita teste e simulação integrados através da aplicação de um conceito denominado "*Ambiente Virtual*", onde entidades reais como aeronaves, tanques e radares podem interagir com entidades simuladas, não importando em que parte do mundo elas realmente estão. (REIS, 2009)

2.3.2 Arquiteturas de simulação vs. protocolos de interoperabilidade

As arquiteturas de simulação apresentadas possuem em suas especificações um mecanismo de interoperabilidade que permite desenvolver sistemas de simulação de

forma distribuída e integrada. Esse mecanismo faz com que o aproveitamento de modelos de simulação legados seja uma tarefa viável. Entretanto, uma desvantagem é que alguns desses mecanismos usam protocolos padronizados com características próprias para fazer a interoperabilidade e a integração de seus componentes, e que, em alguns casos, não é possível usar um componente de uma arquitetura para outra sem algum tipo de intervenção no código fonte do componente.

A Tabela 2.1 mostra a relação entre as arquiteturas de simulação citadas acima e o protocolo de interoperabilidade que elas usam. A coluna a direita representa os protocolos que são aplicados em sistemas de simulação.

Tabela 2.1 Arquiteturas de Simulação vs. Protocolos de Interoperabilidade.

Arquitetura de simulação	***Protocolos de Interoperabilidade***
ADS	ALSP
DIS	IEEE 1278
HLA	IEEE 1516 2010
TENA	CORBA-Like

2.3.3 Ferramentas para simulação de sistemas espaciais

Os simuladores de satélites são sistemas com altos custos de desenvolvimento. Na maioria dos casos estes sistemas são desenvolvidos com tecnologia proprietária, o que torna proibitiva a possibilidade de se obter uma cópia para estudos e análise. Os trabalhos sobre simuladores de satélites encontrados na literatura descrevem as ferramentas, proprietárias ou de código livre, que apóiam o desenvolvimento ou trazem resultados obtidos da aplicação de modelos específicos.

Nesta seção é apresentado um resumo de algumas ferramentas que servem de apoio ao desenvolvimento de simuladores de satélites ou apóiam a análise parcial de uma missão espacial.

2.3.3.1 Simulation Infrastructure for the Modeling of SATellites - SIMSAT

A ferramenta Simulation Infrastructure for the Modeling of SATellites (EGOS, 2007) define uma infra-estrutura de simulação capaz de simular um satélite, através de seus modelos, e simular uma interação com o sistema de controle de satélites do segmento solo de uma misão espacial. A fim de definir os modelos simulados, sem referência direta a qualquer infra-estrutura de uma simulação particular ou um ambiente de hardware, a interface do modelo de simulação, do Inglês *Simulation Model Interface* (SMI), definição de descrição de modelos padronizadas pelo SMP em sua versão 1, foi introduzida no SIMSAT. A interface SMI é usada para auxiliar a portabilidade de modelos de simulação desenvolvidos e apresenta um conjunto de orientações para o desenvolvimento de modelos de portáteis.

O SIMSAT também possuí uma interface-usuário chamada de interface homem-máquina, do Inglês *Man-Machine Interface* (MMI), que oferece uma interface gráfica, amigável, configurável para qualquer simulação. Ela permite ao usuário monitorar a simulação, uma vez que exibe informações sobre o estado do simulador - incluindo as mensagens de log, o tempo e o modo do simulador, o cronograma e os dados de simulação do modelo. O usuário pode controlar o simulador por meio de comandos de execução ou scripts que atuam na simulação.

No momento, somente a versão do SIMSAT para Windows, com tecnologia Microsoft chamade de *Component Object Model* (COM) está disponível, apesar do projeto estar atualmente sendo portado para Linux e CORBA.

Em suas versões anteriores a 4.0 o SIMSAT usava as interfaces padrões do Windows para integrar seus componentes, mas essa abordagem tornava a ferramenta totalmente dependente da plataforma Windows. A partir de versão 4.0 os componentes gerados pela ferramenta exportam uma série de interfaces que podem ser usadas por outros simuladores. O SIMSAT não possuí uma forma de integração e interoperabilidade para aproveitamento de modelos legados de simulação.

2.3.3.2 EuroSIM

O EuroSim (VRIES, 2003) é um produto formado por um conjunto de ferramentas e várias bibliotecas que formam uma aplicação que é executada em uma determinada plataforma.

A proposta principal do EuroSim é ser um ambiente de apoio a preparação , execução em tempo real e a análise dos resultados de uma simulação. O EuroSim permite definir várias configurações para um ambiente de simulação conforme apresentado em NLR (2006).

O EuroSim é uma ferramenta integrada para o projeto de simuladores de satélite que possam ser desenvolvidos ao longo do ciclo de vida de uma missão, incluindo as fases de viabilidade, engenharia, prototipagem, produção e testes, e operação e treinamento. Ele também incorpora um conjunto de ferramentas em SMP que permitem o aproveitamento de modelos legados, mas não possuí nas ferramentas a possibilidade de interoperabilidade entre os modelos.

2.3.3.3 Open-SESSAME Framework

A ferramenta Open-SESSAME (TURNER, 2003) consta de um arcabouço que serve de base para a modelagem e análise por especialistas no desenvolvimento de satélite. Esta ferramenta possuí os seguintes requisitos:

- uma coleção de bibliotecas que possam ser usadas para ajudar na análise e operação de satélites;
- combinação das bibliotecas e de ferramentas para representar uma simulação completa de um satélite em vôo;
- apoio ao desenvolvimento de sistemas em simulação;
- adição de extensão, dos modelos e dos recursos de simulação, na ferramenta.

A ferramenta tem um utilitário que faz a ligação via rede de comunicação entre as aplicações de simulação e componentes de hardware por um servidor de aplicações em Open-SESSAME. Por um lado, não específica um padrão que permita o aproveitamento de modelos legados em simulação de outra plataforma. Por outro lado, por ser uma ferramenta de código aberto permite ao desenvolvedor programar um padrão de interoperabilidade que desejar.

Turner (2003) descreve em seu trabalho como usar a Open-SESSAME como um modelo de extensão no desenvolvimento de um ambiente de modelagem e simulação de satélites. Além disso, gerou o software que estende esse arcabouço, fornecendo a estudantes e pesquisadores uma ferramenta com os modelos mais comuns em simulador de satélites, como dinâmica de atitude, dinâmica orbital e métodos numéricos (Euler, Runge-Kutta, etc.).

Este trabalho levantou a questão do uso de software livre em aplicações espaciais. Em Paul (2007) encontra-se uma discussão das vantagens do uso de ferramentas de software livre para simulação de satélites e um apanhado do estado atual e o potencial uso de software livre no âmbito da área espacial, com ênfase especial para os desenvolvedores de pequenos satélites. Os benefícios que o software de código aberto tem para os pequenos satélites o de aumentar a confiabilidade, diminuir o custo e de ter maior flexibilidade em relação ao software de código fechado ou software desenvolvido exclusivamente in-house, são apresentados.

2.3.3.4 Satellite Toolkit (STK)

O STK (TURNER, 2003) é um pacote desenvolvido pela empresa Analytical Graphics Inc. (AGI). O STK foi inicialmente usado pela comunidade aeroespacial por possuir um modelo de análise de órbita e um sistema de controle e treinamento para o segmento solo. O software foi expandido com módulos que incluem a capacidade de realizar cálculos para o sistema de comunicação, radar, missões interplanetárias e evitar colisões de satélites em órbita.

O núcleo do STK foi projetado para determinar a posição, a dinâmica e a atitude de ativos espaciais. Ele foi originalmente criado para resolver os problemas que envolvem os satélites que orbitam a Terra, e é usado nos setores aeroespacial e de defesa.

A versão original do STK foi desenvolvida originalmente para a Sun Microsystems e mais tarde migrou para PCs em plataforma Windows. Tem sido desenvolvido ao longo de 20 anos como uma ferramenta *Commercial Off-The Shelf* (COTS).

O STK não descreve um mecanismo que forneça aos componentes desenvolvidos pela ferramenta a integração ou interoperabilidade com outras ferramentas ou simuladores. Não descreve em sua documentação formas que permitam a um comprador do produto STK fazer integração de algum modelo legado.

Uma revisão sobre simuladores de satélites comerciais e não comerciais pode ser obtida no trabalho do Turner (2003).

2.4 Simuladores de satélites no INPE

Um simulador de satélite pode ser usado para prover muitos cenários que poderão existir durante o seu ciclo de vida. Alguns dos cenários em que um simulador pode ser aplicado são:

a) validação dos dispositivos aclopados fisicamente (processadores, placas, etc.) e logicamente (modelos discretos e contínuos) em um subsistema do satélite ou um subsistema de rastreio e operação (em solo);

b) teste do Sistema de Controle do Satélite;

c) treinamento de operadores de satélite;

d) validação dos procedimentos operacionais de voo;

e) análise de comportamento de subsistemas em diferentes fases da missão;

f) verificação e validação dos programas de computadores a bordo.

O INPE tem uma história de desenvolvimento de simuladores nas mais váriadas áreas de trabalho. O primeiro simulador documentado foi a ligação do equipamento EAI-640 no computador B6700 da Borroughs (MARANHÃO, 1977), passando por vários simuladores de análise ambientais, climático, matemático, computacional e espacial no desenvolvimento de satélites.

Ambrosio et al. (2006) apresentam um resumo dos simuladores operacionais desenvolvidos e utilizados pelo INPE para apoio as atividades do Centro de Controle de Satélites do INPE. Muitos outros trabalhos de simulação foram feitos durante o desenvolvimento dos satélites, mas possuiam como principal objetivo o apoio aos engenheiros em seus trabalhos, teses, dissertações e a validação dos vários cenários que cada grupo ou departamento necessitava durante as fases de desenvolvimento.

Três diferentes simuladores operacionais de satélites foram desenvolvidos pelo INPE, são eles.

a) *Simuladores da família SCD* (Satélite de Coleta de Dados)

   O desenvolvimento do primeiro simulador de satélite, o SIMS, foi iniciado em 1991 para o primeiro satélite de coleta de dados (SCD1), o qual foi lançado em 1993 e ainda está em operação. O SIMS foi desenvolvido totalmente pelo pessoal técnico do INPE seguindo o paradigma clássico de desenvolvimento em cascata. O projeto do software foi baseado nos conceitos de análise estruturada e a linguagem de programação usada foi o Fortran.

   Embora o SIMS tenha sido o primeiro simulador desenvolvido pelo INPE, demonstrou ter uma alta fidelidade em muitos dos subsistemas do satélite e tido um satisfatório acoplamento com os requisitos dos stackeholders. Um requisito a ser ressaltado foi a alta fidelidade com o ambiente espacial, estação terrena (equipamentos e interfaces com o sistema de controle do satélite), e a simulação dos subsistemas do satélite. Este software sofreu modificações para

atender as necessidades do próximo satélite de coleta de dados o SCD2 e SCD2A.

b) *Simulador para o satélite CBERS* (China-Brazil Earth Resources Satellite)

O segundo simulador de satélite usado no treinamento da equipe de operação foi o SIMC. O simulador fazia parte da cooperação entre o Brasil e a China no desenvolvimento do primeiro satélite de recursos da terra o CBERS-1, que foi colocado em órbita em 1998 pelo lançador Chinês denominado de Long March 4B.

A particulariedade deste simulador é que foi desenvolvido em cooperação entre o Centro de Controle de Satélite Brasileiro (CCSB)/INPE-Brasil e o Centro de Controle de Satélite de Xian (XSCC)/China. As negociações se iniciaram em 1997 e a última versão foi liberada pelo XSCC ao Brasil em 1999. A equipe de brasileiros produziu os requisitos e participou das revisões formais do projeto, enquanto a equipe de chineses projetou e codificou o simulador.

O SIMC é um simulador de média fidelidade. Foi escrito em linguagem C++ e é executado na plataforma PC/Windows NT.

No lado brasileiro, o principal benefício obtido deste simulador, além do suporte ao treinamento dos operadores, foi o de estreitar a lacuna de conhecimentos que havia entre as equipes de operação dos dois países; e assim foi possível melhorar as informações relacionadas aos detalhes de operação, como são os campos e seus significados em telecomandos do AOCS, contador de mensagens, a posição dos bits no quadros de telecomando e telemetria puderam ser verificados.

c) *Simulador para o micro-satélite científico Franco-Brasileiro*

O terceiro simulador de satélite, o FBMSIM foi projetado para o treinamento da equipe de operação do micro-satélite Franco-Brasileiro (FBM). Ficou com o

Brasil, acordado na cooperação do projeto com CNES/France, o desenvolvimento do Sistema de Solo. Em 2002, foi decidido que seria adquirida uma ferramenta para o desenvolvimento de um simulador de satélite para ganhar tempo na criação do simulador do FBM, o motivo desta aquisição era o reduzido grupo de pessoas do INPE para o desenvolvimento. A contratação do desenvolvimento do simulador completo estava fora de questão, pois haviam informações confidencias sobre o OBDH. A opção foi a aquisição da ferramenta comercial, isto é, produto com as funções mais comuns de um simulador de satélite. Este solução parecia, a primeira vista, ser muito interessante, pois poderia acelerar o desenvolvimento sem a necessidade de abrir as informações proprietárias para terceiros. Infelizmente, a missão FBM foi descontinuada e o simulador completo não foi desenvolvido.

A Tabela 2.2 apresenta as características dos simuladores desenvolvidos para os satélites feitos pelo INPE.

Tabela 2.2 Característica dos simuladores de satélites do INPE.

	Simuladores de Satélites		
	SIMS	SIMC	FBMSIM
Características			
Satélite	SCD1 SCD2 SCD2A	CBERS1 CBERS2	FBM EQUARS
Comunicação e protocolos	Formatos TM e TC ESA SDID e X.25	Formatos TM e TC ESA SDID TCP/IP e X.25	CCSDS E TCP/IP
Grau de fidelidade	Alto	Médio	Médio
Plataforma	VAX-VMS Alpha-Open VMS	MS-Windows/NT	MS-Windows/NT
Estilo de projeto	Análise Estruturada	Orientado a objetos	Orientado a objetos
Lingaguem de Programação	Fortran	C++	C++
Linhas de código programado	50.000 linhas	60.063 linhas	12.900 linhas,

Continua

Tabela 2.2- Conclusão

	aproximadamente		exceto as incorporadas das classes do CTOS
Tempo de desenvolvimento	2 anos	1.5 anos	4 anos
Número de desenvolvedores	8 pessoas	6 pessoas (China) 3 pessoas (Brasil) na especificação	0.8 pessoas/ano (muitas mudanças de pessoas)
Tempo de uso	6 anos aproximadamente	Em atividade, mas com pouco uso	Somente um protótipo foi terminado
Grau de satisfação	Alto	Médio	Baixo
Grau de reúso	Alto para a familia de satélites, mas baixo para outros satélites	Sem reúso	Não conhecido

Fonte: AMBROSIO. (2006)

2.5 Resumo

Este Capítulo apresentou os principais conceitos relacionados a esta dissertação. Tais conceitos permitiram compreender a interoperabilidade de sistemas, os sistemas legados e a relação dos modelos legados em simuladores.

CAPÍTULO 3

SIMULATION MODEL PORTABILITY

"... por distinção, um enganar por palavras é comumente chamado de mentira e, enganar por palavras, gestos e comportamento é chamado de simulação..."

(ROBERT SOUTH)

3.1 Princípios sobre a arquitetura padrão do SMP

Nesta seção é apresentada a arquitetura padrão SMP usada como base de desenvolvimento desse trabalho.

Os modelos gerados para os simuladores são executados para os mais variados objetivos, usando diferentes linguagens de programação, executados em diversos sistemas computacionais, ambientes operacionais e produzidos por uma variedade de empresas contratadas para o seu projeto e desenvolvimento (ESOC, 2005a).

Com o objetivo de incentivar a padronização em simuladores de satélites, a Agência Espacial Européia (ESA), juntamente com a indústria européia, definiu o *Simulation Model Portability* (SMP). A proposta principal do SMP é promover a portabilidade e o reúso dos modelos em simulação. Em 1999, iniciaram-se as atividades de documentação das especificações para um padrão em simulação e modelagem. A primeira liberação do SMP foi em 2001 em sua versão 1. Essa versão vem sendo usada a partir desta data em projetos de diferentes ambientes de simulação, propiciando um alto grau de maturidade nas aplicações desenvolvidas. Visando aperfeiçoar esse padrão com a inclusão de novas tecnologias, sendo uma a interoperabilidade, uma nova especificação do SMP em sua versão 2 foi liberada em 2005 com a sigla SMP2. O padrão ECSS (2011a), liberado em 25 de Janeiro de 2011, descreve em forma de memorando técnico as especificações e os requisitos para o SMP. O SMP não possuí um documento único que apresenta as suas características, mas um conjunto de especificações e requisitos para a codificação, uso da interface e a geração da documentação. São 4 (quatro) os seus objetivos principais:

a) minimizar as interações dos modelos com o ambiente;

b) padronizar as interfaces usadas pelos modelos;

c) criar modelos com interfaces simples;

d) criar modelos facilmente entendidos pelos desenvolvedores.

O resultado destes objetivos é promover o desenvolvimento em uma plataforma independente, interoperável e reutilizável de modelos de simulação, baseado em padrões abertos de projetos de componentes. (OMG, 2011d)

O *Simulation Model Portabilty* (SMP) em sua versão 2 é um guia com:

a) os formatos padronizados das interfaces que os modelos devem possuir;

b) o modo de troca de mensagens entre os modelos e com o núcleo de simulação;

c) promoção de portabilidade e reúso dos programas e de desenvolvimento sobre uma plataforma independente;

d) definição do modo de utilização de uma arquitetura única de interoperabilidade.

O SMP utiliza os padrões abertos como o *Unified Modeling Language* (UML) (BOOCH et al., 2005) e o *Extensible Markup Language* (XML) (RAY, 2001), e formula dois importantes requisitos (ESOC, 2005a):

a) *Common Concepts*: estabelece que o desenvolvimento do modelo será feito em um nível de abstração essencial para ser independente de plataforma e ser reutilizável;

b) *Common Type System*: define que diferentes modelos tenham em comum: uma base sintática e semântica, o que é essencial para a interoperabilidade dos diferentes modelos.

O primeiro requisito, específica que todos os modelos são derivados de um único conceito fundamental, comum e conhecidos; e o segundo, que a criação de um modelo tem uma base comum e um conjunto de elementos comuns como classes, atributos e métodos que interagem entre si. Assim, os modelos convivem entre os dois requisitos como mostrado na Figura 3.1.

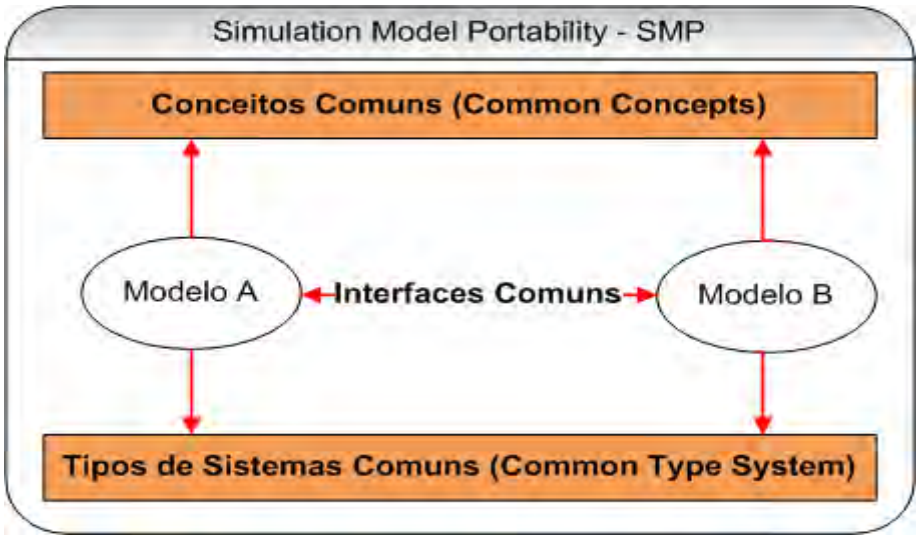


Figura 3.1 Princípios que compõem as camadas do SMP.

Fonte: Adaptada ESOC. (2005a)

A arquitetura do SMP separa bem os elementos de uma plataforma específica (como o computador, o sistema operacional, a linguagem de programação, etc.) e os que são independentes da plataforma (interfaces, descrições, atributos, etc.).

Na fase de projeto e desenvolvimento os modelos são descritos em termos de interfaces, atributos e características. Já na fase de execução é descrito como estes modelos serão instanciados para a execução na simulação. Com estes conceitos podem-se definir quais elementos podem ser reusados e como eles são integrados na execução do simulador. Os

elementos que fazem parte do tempo de projeto e o de execução no SMP são mostrados na Figura 3.2.

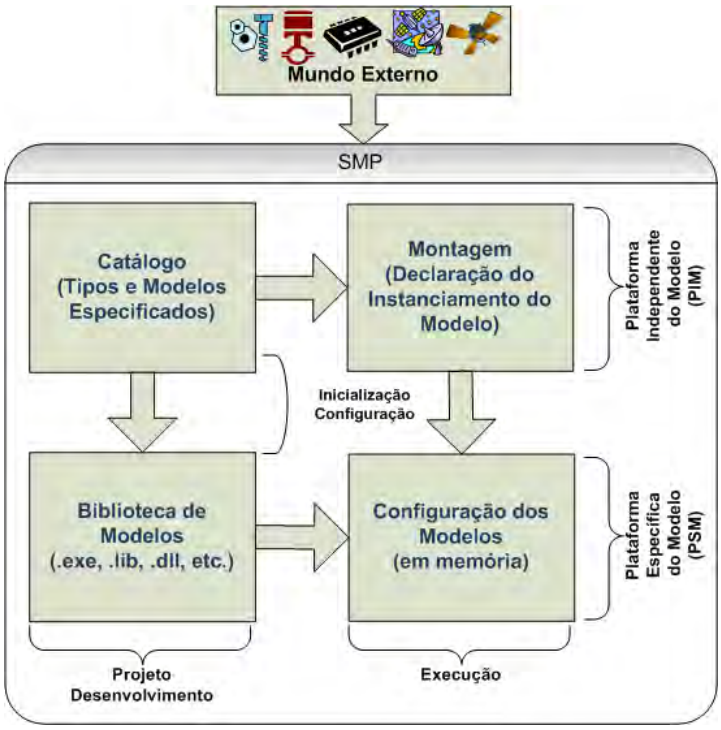


Figura 3.2 Componentes do SMP.

3.2 Conceitos sobre o SMP

A arquitetura conceitual do SMP, ilustrada na Figura 3.3, possuí as camadas que correspondem aos diferentes níveis de abstração em conformidade com o conceito de arquitetura orientada a modelos (MDA).

A camada superior representa o mundo real e as necessidades dos modelos a serem usados no simulador. O SMP não orienta o desenvolvimento dos modelos, embora reconheça que o uso de algum padrão de desenvolvimento é importante.

A segunda camada representa a Modelagem Independente da Plataforma (PIM) do sistema a ser simulado. O objetivo principal desta camada é definir as declarações das especificações do modelo, quais sejam: (i) Catálogo (arquivo *Catalogue*), (ii) Montagem (arquivo *Assembly*) e (iii) Escalonador (arquivo *Schedule*). A linguagem

denominada *Simulation Model Definition Language* (SMDL), baseada em XML, é usada para declaração dos modelos nos arquivos usados na simulação. (ECSS, 2011b)

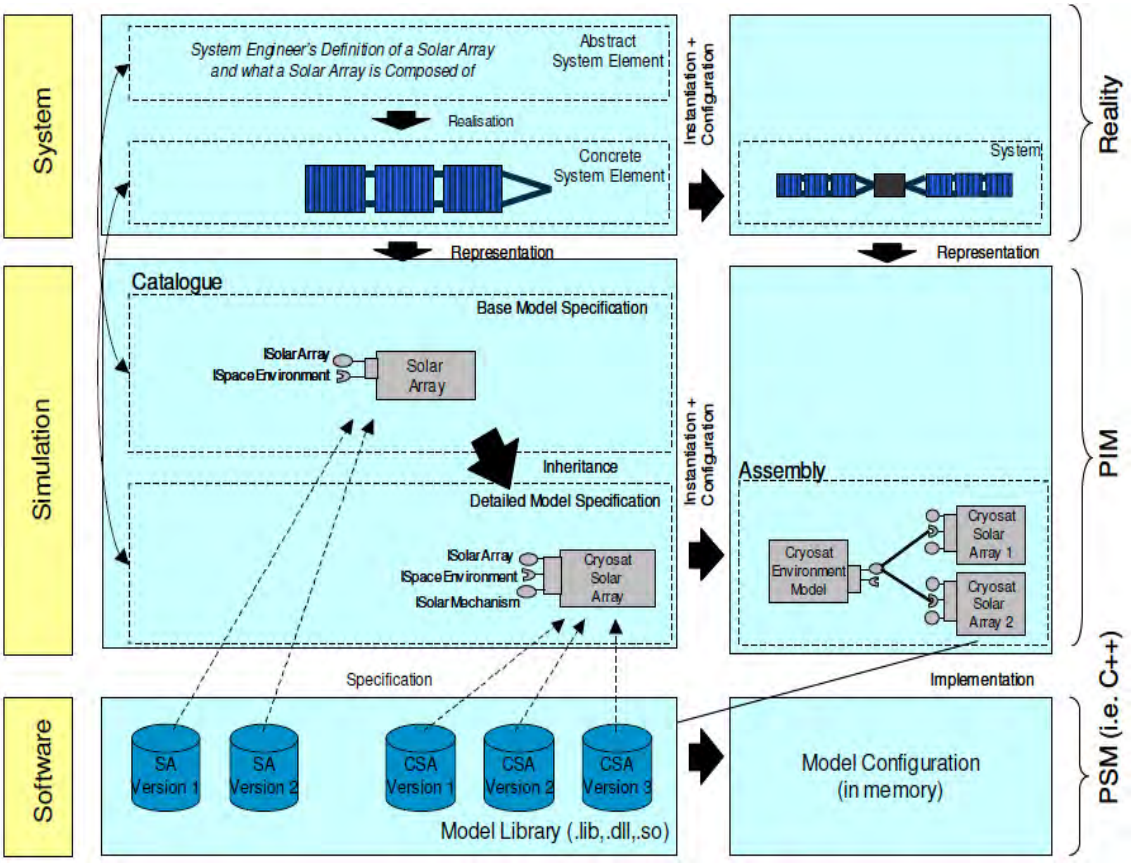


Figura 3.3 Separação dos componentes do SMP.

Fonte: ESOC. (2005a)

O arquivo *Catalogue* possuí a descrição das especificações das interfaces e dos atributos do modelo para o SMP. O arquivo *Assembly* possuí as declarações de como instanciar as interfaces e atributos que foram declarados no arquivo *Catalogue*.

A terceira camada representa o Modelo de Plataforma Específica (PSM) do sistema simulado, este é o nível de implementação física das declarações do modelo no SMP, atualmente feita em ANSI/ISO C++. Embora as declarações possam ser feitas manualmente, parte da implementação deve tratar da integração e também da geração das informações a serem guardadas no arquivo *Catalogue*. O suporte ao modelo de

componente do padrão SMP e o conjunto de serviços padronizados que fazem parte do PSM estão descritos em ECSS (2011c). Finalizando, as descrições dos metadados e dos componentes modelados devem ser mapeadas em ANSI/ISO C++ para depois serem compiladas e assim o simulador torna-se executável para padrão SMP. (ECSS, 2011d)

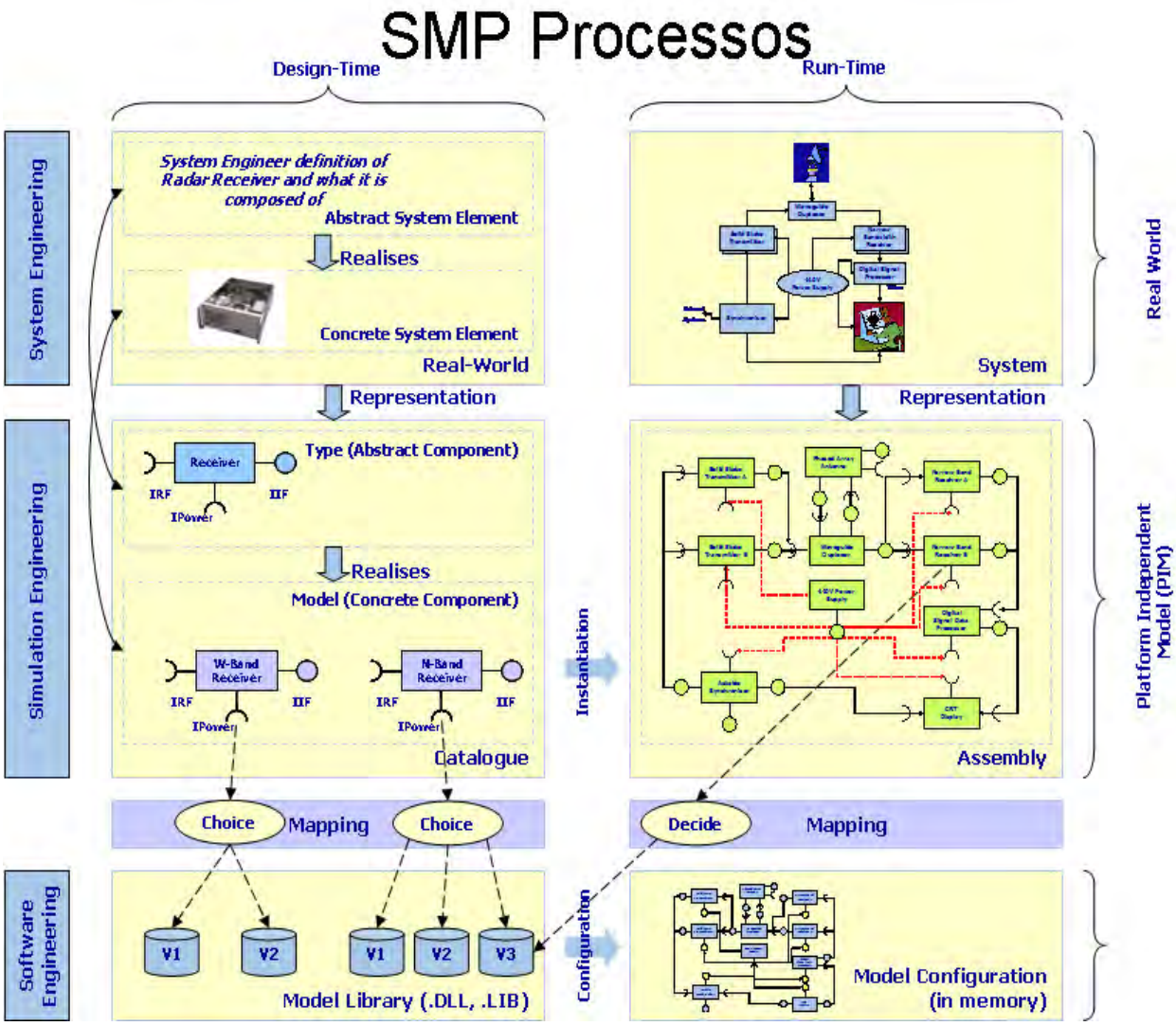


Figura 3.4 Visão expandida dos processos do SMP.

Fonte: NASA; ESA. (2004)

A Figura 3.3 apresenta os conceitos das camadas, como são definidas no MDA, enquanto a Figura 3.4 detalha quais processos podem ser aplicados em cada uma das camadas especificadas. A camada *System* da Figura 3.3 tem uma correspondência com a

camada chamada *System Engineering* da Figura 3.4 Esta camada mostra um elemento físico ou lógico do mundo real, que deverá ser modelado para uma simulação. A camada *Simulation* da Figura 3.3 possuí a base das especificações do modelo, tradução do mundo real, em sua representação para uma linguagem de modelagem como o SysML. Na camada *Simulation Engineering* da Figura 3.4 que declara os componentes e as suas interfaces que podem ser abstratos ou concretos de um modelo, esta camada tem a denominação de *Platform Independent Model* (PIM). A camada Software da Figura 3.3 que representa a biblioteca dos modelos no ambiente lógico, ou seja, a codificação em alguma plataforma computacional, o *Software Engineering* da Figura 3.4, ou Engenharia de Software (PRESSMAN, 1995), que oferece um conjunto de metodologias e técnicas para o desenvolvimento de software, denominada de *Platform Specific Model* (PSM).

Os componentes são desenvolvidos pelos processos associados às fases de Projeto (*Design-Time*) e Execução (*Run-Time*), e cada um tem as suas próprias especificações. O *Design-Time* fornece um ambiente de controle sobre as especificações do modelo com a utilização de ferramentas para engenharia, como exemplo o SYSML. Enquanto o *Run-Time* executa os modelos em uma plataforma computacional sob a administração de um ambiente de simulação.

O fluxo de informações entre Projeto e Execução é feito via arquivos criados no Design-Time e usados no *Run-Time*. Em *Run-Time*, os vários modelos declarados no arquivo *Catalogue* são instanciados. O arquivo *Catalogue* possuí as definições abstratas de cada modelo de um ambiente de simulação e no arquivo *Assembly* as declarações de como os modelos serão instanciados para a sua execução, de acordo com as referências ao seu catálogo, podendo ter várias referências de modelos. Cada modelo pode ainda estar associado a um arquivo *Catalogue* diferente.

3.3 Arquitetura do SMP

O SMP adota os conceitos XML para as declarações dos modelos e o IDL/CORBA para definição da *Platform Independent Model* (PIM) que específica os modelos e para a

Platform Specific Model (PSM) que específica os modelos de componentes, ou seja, a programação das interfaces. A *Interface Specifications* descreve a exposição das interfaces entre os modelos e/ou simulador através do mapeamento do PIM para o PSM usando a plataforma ANSI/ISSO C++.

3.3.1 Platform Independent Model (PIM)

As definições dos elementos que compõem o PIM do SMP são definidas no formato XML, através de uma descrição não ambígua das especificações dos modelos a serem simulados, chamado de Metamodelo. O IDL/CORBA é a linguagem para descrição das interfaces, as quais especificam a implementação física dos componentes de simulação, o chamado *Component Model*.

A linguagem de definição do modelo de simulação, do Inglês *Simulation Model Definition Language* (SMDL) (ECSS, 2011b), representa uma linguagem em Metamodelo que providencia um mecanismo independente da plataforma para desenhar e descrever as declarações sobre os modelos (*Catalogue*), integrar as instâncias dos modelos (*Assembly*) e o enfileiramento dos mesmos (*Schedule*). O SMDL suporta o projeto e integração das técnicas baseadas nos modelos em classe de objetos, interfaces, componentes e eventos.

As declarações das especificações dos modelos, descritos em SMDL, são geradas no formato XML (RAY, 2001) e podem ser validadas pelos esquemas gerados automaticamente da declaração em SDML. Este mecanismo garante a facilidade na troca de informações da modelagem entre os interessados.

O Metamodelo possuí três partes principais para definir um modelo: o *Catalogue*, o *Assembly* e o *Schedule*. O *Catalogue* possuí informações das declarações das interfaces, quais os tipos de variáveis usadas e a estrutura dos modelos, enquanto o *Assembly* descreve como as instâncias dos modelos estão interconectadas em uma configuração específica, baseada nos tipos de informações definidas no arquivo *Catalogue*. Finalmente, o *Schedule* possuí informações adicionais de enfileiramento durante a

montagem dos modelos para a simulação. Existem dois arquivos adicionais, não obrigatórios, o Package que define um agrupamento de *Catalogue* e *Assembly* e o *Workspace* que faz referência a um conjunto de documentos de uma determinada simulação. O esquema do Metamodelo para o SMDL é apresentado na Figura 3.5 enquanto a estrutura do Metamodelo para o SMP é mostrado na Figura 3.6. As figuras mostram o metamodelo na linguagem de modelagem UML.

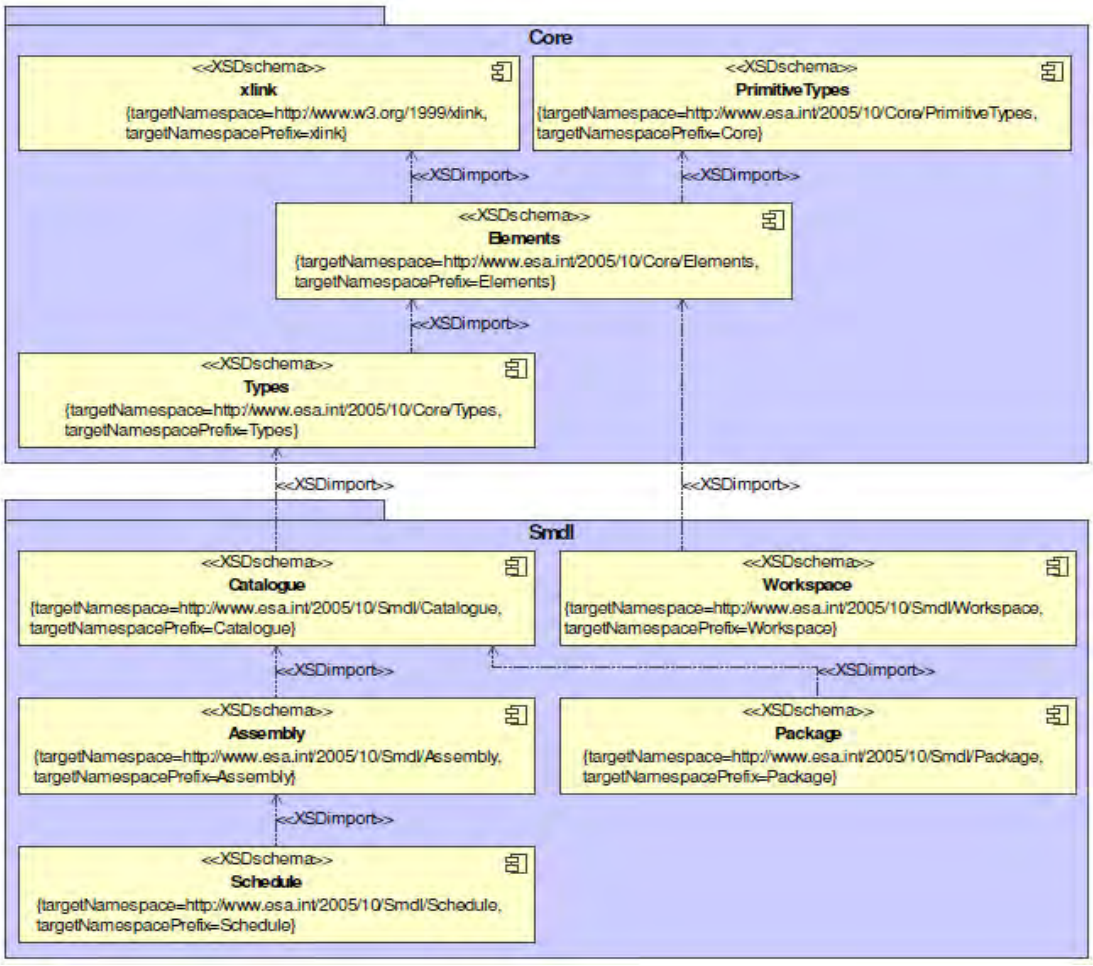


Figura 3.5 Esquema do Metamodelo em SMDL.

Fonte: ECSS. (2011b)

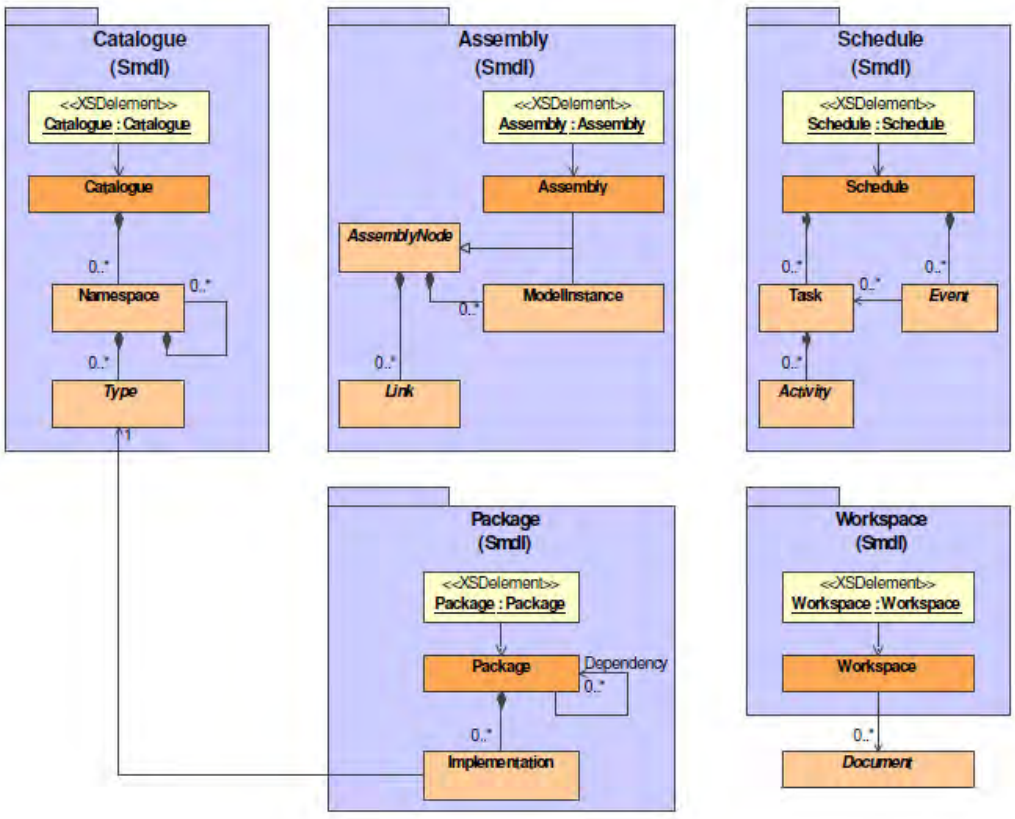


Figura 3.6 Estrutura do Metamodelo para o SMP.

Fonte: ECSS. (2011b)

Em ECSS (2011b) é descrito os requisitos necessários para o desenvolvimento das ferramentas necessárias a geração dos arquivos metamodelos no padrão SMDL. Tais ferramentas são:

a) editor - arquivos *Catalogue*, *Assembly* e *Schedule*;

b) validador - compila os arquivos XML dos esquemas gerados pelo editor;

c) gerador de código - gera o código fonte para uma plataforma específica usando as informações declaradas no *Catalogue*;

d) gerador de documentos - produz a documentação dos modelos que foram descritos no *Catalogue*;

e) conversor - traduz um modelo descrito no *Catalogue* em UML ou vice-versa.

O Metamodelo, chamado de SMDL, é desenvolvido em UML usando a ferramenta Magic Draw 9.5. A saída gerada pela ferramenta é um arquivo com a estrutura nativa em *XML Metadata Interchange* (XMI) (KOVSE; HÄRDER, 2002), que é usada como entrada no programa *XML Schema Converter* (PENN et al., 2003) , versão 2.0, e gerou os esquemas *XML Schema Definition* (XSD) (BEX et al., 2007), o qual é o mecanismo básico de controle de consistência dos arquivos em SMDL.

Como a base de definição do SMDL é desenvolvida em uma linguagem de modelagem, como descrita acima, uma ferramenta de modelagem em UML pode ser usada para descrever as especificações dos modelos no formato reconhecido pelo SMP. Em ESOC (2005c) há uma descrição de como usar as ferramentas de modelagem em UML para gerar os arquivos no padrão SMDL.

No SMP três são as partes principais que compõem uma simulação: (i) o próprio ambiente de execução da simulação, (ii) os serviços e (iii) os modelos.

Em ECSS (2011c) é apresentado uma proposta de forma não ambígua de como definir os componentes a serem desenvolvidos conforme a padronização do SMP, explicitando quais as partes da simulação, os componentes opcionais e os mecanismos que estarão disponíveis para os modelos. No documento estão descritas as interfaces obrigatórias no formato de plataforma independente. O mesmo documento não tem a intenção de descrever em detalhes a programação das interfaces em linguagem ANSI/ISO C++. O mapeamento da *Platform Independent Model* (PIM) para dentro da *Platform Specific Model* (PSM) fica a cargo dos desenvolvedores da simulação a sua programação.

A linguagem de definição usada no SMP, para descrever as especificações das interfaces de forma independente é a linguagem *Interface Definition Language* (IDL) do CORBA. Desta forma, todas as especificações e características associadas ao IDL são absorvidas nas especificações de ECSS (2011c), tornando-as independentes da plataforma.

A programação do *Component Model* (ECSS, 2011c) representa um kernel em tempo de execução (*Run-time*) do ambiente para a execução da simulação e deve estar conforme os padrões do SMP, com relação aos tipos de dados, as interfaces e aos serviços. Um resumo de ECSS (2011c) é descrito a seguir:

a) *Tipos de dados*: os padrões definidos são o boolean, integer, float, date e time, string, universal unique identifier (UUID) e elementos do tipo collections;

b) *Interfaces e serviços mandatórios*: a especificação do SMP distingue entre as interfaces mandatórias e as opcionais. As principais interfaces e serviços que são obrigatórios no *Run-time* são:

- **Interfaces**:

  ISimulator: é a principal interface que fornece acesso aos modelos e aos serviços. Ela é responsável pela transição de estados da simulação para isso deve possuir os métodos de transição bem definidos conforme mostrado na Figura 3.7.

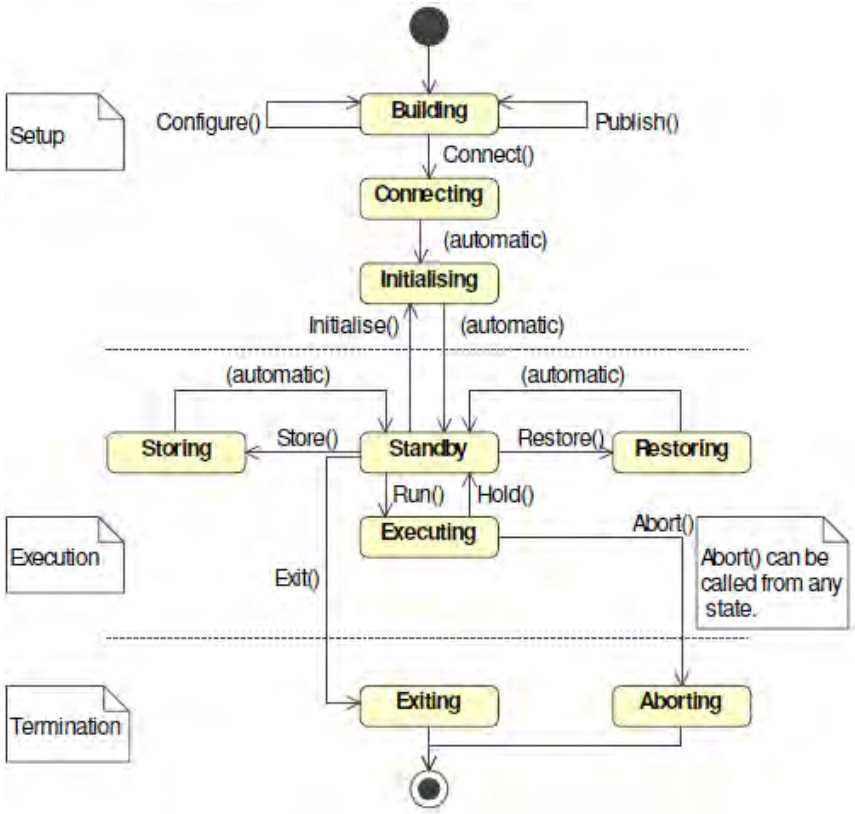


Figura 3.7 Estado de Transição da Simulação.

Fonte: ESOC. (2005c)

IDynamicSimulator: esta interface estende a interface ISimulator e adiciona métodos que podem criar componentes dinamicamente, são tipicamente os modelos de uma fábrica de componentes descritos na interface *IFactory*;

IPublication: possuí a funcionalidade de publicar membros, incluindo os campos e operações. Na inicialização, cada modelo precisa ser acessado pelo publicador para receber os campos e operações que foram publicados.

Existem outras 28 (vinte e oito) interfaces especificadas no ECSS (2011c).

- **Serviços**:

Logger: este serviço providencia um método para enviar uma mensagem para o arquivo de log do simulador;

Time Keeper: gerencia e controla o tempo de simulação conhecido como serviço "Simulation Time", o SMP suporta quatro tipos diferentes de tempo;

Scheduler: este serviço é baseado nos pontos de entrada (entry point) dos modelos e os eventos são disparados (trigger) por um dos quatros tipos de tempo (Time Keeper);

Event Manager: providencia um mecanismo global de notificação.

Para as declarações em PIM são necessários que se descrevam os tipos usados pelos modelos como as suas interfaces, estruturas, enumeração, variáveis, etc. como mostrado na Figura 3.8.

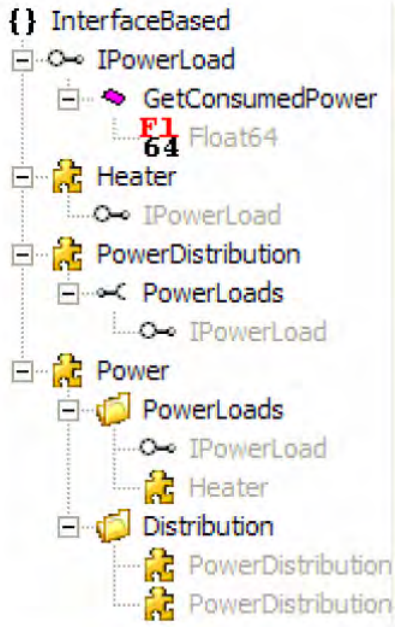


Figura 3.8 Exemplo de elementos SMP em uma ferramenta automatizada.

Fonte: ESOC. (2005c)

As atividades de administração das declarações de um modelo não estão cobertas nas especificações do SMP, mas somente as ferramentas necessárias para tal:

a) *Editor de Catalogue*: este editor cria e edita o documento Catalogue do SDML, o qual define os tipos e modelos que serão usados com o simulador SMP.

b) *Validador do Catalogue*: valida o conteúdo do documento com o esquema do *Catalogue* do SDML. Não é uma validação sintática sobre o XML Schema definido para o *Catalogue*, mas uma validação semântica.

c) *Editor do Assembly*: editor que cria e edita o documento *Assembly* do SDML, o qual define a forma de integração dos modelos definidos no *Catalogue* do SDML.

d) *Validador do Assembly*: valida o conteúdo do documento com o esquema do *Assembly* do SDML. Não é uma validação sintática do XML Schema definido para o *Assembly*, mas uma validação semântica.

e) *Gerador de código*: geração de um esqueleto em código fonte em ANSI/ISO C++ dos modelos contidos no arquivo *Catalogue* do SDML. Este código garante que a geração de alguma biblioteca esteja sincronizada com a correspondente descrição do modelo no arquivo *Catalogue*.

3.3.2 Platform Specific Model (PSM)

O PSM é uma especificação de sistema, de acordo com (PRESSMAN, 1995) e representa a implementação física do processo. No caso do SMP as declarações do modelo, no nível mais próximo do código fonte é o produto da integração de um PIM. A especificação PSM traduz a ideia principal de aproveitamento do código fonte de uma plataforma específica através de padrões de transformação.

Nas especificações do SMP a ligação existente entre o PIM e PSM de um modelo é que a partir das declarações feitas do modelo no catálogo, é possível gerar o código fonte

dos atributos e das interfaces e usar este código no programa do modelo fazendo a ligação do simulador à biblioteca de modelos (fontes, .exe, .dll, .lib, etc.) conforme apresentada na Figura 3.9 abaixo.

3.3.3 Interfaces de comunicação

O SMP **não** descreve as interfaces de comunicação, a interoperabilidade, entre os modelos descritos no PIM. O formato usado é a troca de mensagens através da estrutura de interfaces que são declaradas usando a linguagem ANSI/ISSO C++. Com uso deste formato, as interfaces somente poderão ser programadas na linguagem ANSI/ISO C++ e dentro do próprio código da simulação e dos modelos.

3.4 Processo de desenvolvimento de um simulador no padrão SMP

Esta seção discute a forma de integração dos vários elementos que apóiam a definição de atributos e declarações dos modelos em um ambiente integrado de **Projeto** e **Desenvolvimento** de um simulador no padrão SMP.

O SMP está em conformidade com a abordagem *Model Driven Arquitecture* (MDA), pois recomenda especificar as declarações dos modelos de forma independente (PIM) e utiliza uma linguagem específica de domínio, a chamada SDML. Por um processo de engenharia aplicado no código fonte do modelo, a descrição em PIM será integrada a uma plataforma específica (PSM), e para isso é necessária uma ferramenta automatizada que forneça suporte às transformações e mapeamentos dos modelos, segundo o padrão SMP.

A referência ECSS (2011a) apresenta uma forma de controle e automatização de processos para a criação das declarações e integração de modelos em SMP.

A Figura 3.9 mostra os principais elementos e o fluxo de informações desde a concepção do Projeto até a obtenção do código executável, no Desenvolvimento de um simulador.

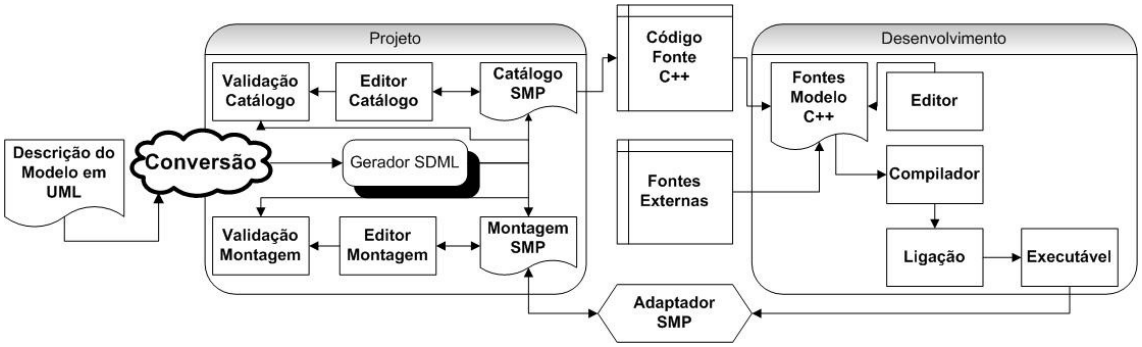


Figura 3.9 Do Projeto ao Desenvolvimento do Código executável, de acordo como SMP.

As especificações do SMP não têm referência a um ambiente integrado de Desenvolvimento ou uma ferramenta que automatize o processo de criação, desenvolvimento e manutenção das declarações dos modelos durante o ciclo de vida de um Projeto.

Um Projeto de um simulador no padrão SMP, pode ter a criação das declarações de um modelo feita de duas formas possíveis:

a) a primeira usa-se alguma linguagem de modelagem, como por exemplo o UML, para definir os atributos e interfaces de um modelo. Esta tarefa exige o uso de um Editor do artefato em UML. Como os arquivos gerados por este Editor não estarão em conformidade com o formato da linguagem SMDL, especificada pelo SMP, é necessário fazer uma Conversão das informações dos arquivos UML para o formato lido pelo Gerador SDML. No documento ESOC (2005c) encontra-se uma descrição de como transformar um modelo em UML em um modelo em SMDL.

b) a segunda forma é usar uma ferramenta específica para gerar os arquivos que descrevem o comportamento do modelo diretamente no formato SMDL. Esta ferramenta deve ter as funcionalidades para criar, desenvolver e manter os arquivos das declarações e das interfaces do modelo a ser simulado.

A tarefa de codificação do modelo em alguma linguagem de programação pode ser feita através de um ambiente integrado de desenvolvimento como o Microsoft Visual Studio, o Borland C Builder, entre outras ferramentas que facilitam o processo de desenvolvimento de software. O SMP não contempla uma metodologia ou ferramenta para o desenvolvimento dos modelos novos ou modificação em modelos legados. Mecanismos ou ferramentas para o desenvolvimento de um modelo são deixados a critério do desenvolvedor.

A principal ligação entre o Projeto e o Desenvolvimento é a geração dos códigos fontes, a partir das especificações dos atributos e das interfaces dos modelos que foram declarados nos arquivos *Catalogue* em SDML. Estes arquivos gerados são integrados ao código fonte do modelo para a geração do programa executável. Desta forma, o programa do modelo pode ser instanciado pelo simulador através de um mecanismo de adaptação do SMP no ambiente do simulador.

O SMP não possuí orientações para que modelos possam ser executados remotamente, ou seja, não inclui soluções que permitam a interoperabilidade.

Resumidamente, o processo de desenvolvimento de um simulador no padrão SMP consiste na transformação das definições, em alto nível do modelo (arquivo em PIM) para uma plataforma específica (formato PSM) e em seguida a transformação do formato PSM para o código executável, como ilustrado na Figura 3.10.

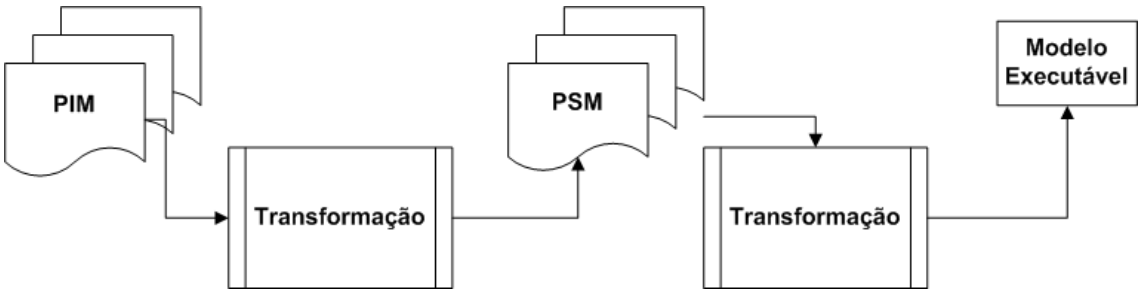


Figura 3.10 Resumo do processo de desenvolvimento de um simulador em SMP.

3.5 Resumo

Este Capítulo mostrou os principais conceitos do padrão SMP e apontou as limitações para criação de um simulador que inclua modelos legados. Com base nos estudos dos documentos de padronização escritos pela ESA, sobre o SMP, é apresentada uma forma para o desenvolvimento de um simulador no padrão SMP.

CAPÍTULO 4

ARQUITETURA SMP COM MECANISMOS PARA COMUNICAÇÃO DE MODELOS LEGADOS

Este Capítulo descreve algumas soluções para o uso de modelos legados em simuladores de satélites, visando permitir que os modelos legados convivam com modelos em novas tecnologias em um simulador conforme o padrão SMP.

A solução proposta para que modelos legados integrem um simulador de satélites no padrão SMP consta de incluir uma facilidade no SMP para aceitação e manipulação de modelos executados remotamente. Assim tais modelos não precisam ser projetados de acordo com o padrão SMP para fazerem parte de um simulador no padrão SMP. Dado que as especificações do SMP não incluem orientações para declarações que permitam que os modelos legados sejam executados remotamente, esta dissertação apresenta uma solução.

4.1 Contexto de simulação

Um simulador compreende um conjunto de elementos que se interligam e cooperam para realizar a simulação. No texto desta dissertação, o termo **contexto da simulação** foi usado como o conjunto dos elementos que formam o simulador. Desta forma, o **contexto da simulação** possuí todos os elementos usados ou referenciados no desenvolvimento e no uso de um simulador, como os modelos, os interresados externos, etc..

Mais precisamente, o contexto da simulação, como apresentado na Figura 4.1 compreende:

a) os modelos a serem simulados, os quais podem ser desenvolvidos (modelos novos) ou já existirem (os modelos legados);

b) um módulo de ligação entre os vários elementos que fazem parte do contexto;

c) o ambiente de simulação ou simplesmente chamado de simulador, o qual permite que os elementos externos e internos atuem sobre a execução do simulador;

d) os interessados externos, ilustrados na Figura 4.1, como Externo 1 e Externo 2. Os interessados externos podem ser os condutores da simulação, os analistas dos resultados da simulação ou ainda outro elemento físico ou lógico que interage em algum grau com o contexto da simulação,

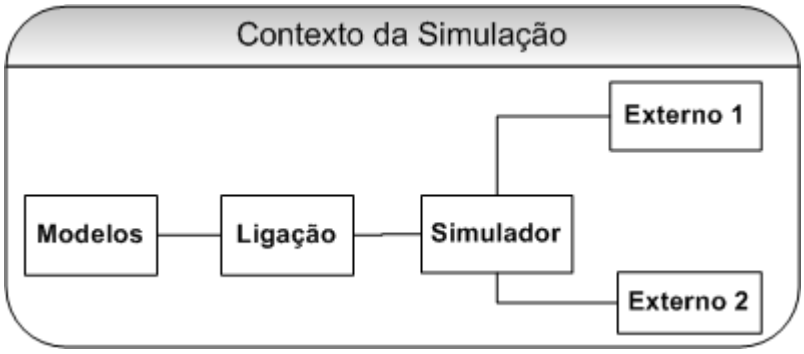


Figura 4.1 Esquema mostrando os elementos que interagem com ou fazem parte de um simulador.

4.2 Abordagens para inclusão de modelos legados em simuladores

A inclusão de modelos legados em um simulador pode causar problemas ligados à evolução das simulações, pois, sendo essenciais ao contexto da simulação, não são recriados com facilidade para atender os novos requisitos tecnológicos. A integração dos modelos criados com novas tecnologias podem não se adaptar aos modelos desenvolvidos com uma realidade tecnológica que foi tornada obsoleta pela fantástica evolução que a computação teve nos últimos anos.

A seguir são apresentadas 3 (três) abordagens para reúso de modelos legados em simuladores que seguem o padrão SMP.

Abordagem 1 - engenharia reversa

O aproveitamento direto de um modelo legado em um simulador de satélite é possível quando o código do modelo pode ser reusado em outra plataforma, para isso pode-se usar a técnica de engenharia reversa (CANHOTA et al., 2005). O resultado do trabalho da engenharia reserva é uma mudança do programa para uma linguagem que possua os requisitos da nova plataforma computacional e da tecnologia de orientação a objeto (CAGNIN; PENTEADO, 2011). Ao ser feita a mudança do código do modelo para a plataforma orientada a objeto pode-se usar a arquitetura especificada no padrão SMP. A Figura 4.2 ilustra, de forma esquemática, um Simulador A, do padrão SMP, composto de 3 modelos A, B e C e adaptável a plataforma A; e um Simulador B, também no padrão SMP composto dos modelos A, B, C, D e E. Os modelos A, B e C são de uso comum aos simuladores A e B, isto é possível, pois os modelos são desenvolvidos usando as especificações de portabilidade e reúso do SMP. Observa-se o aproveitamento dos modelos entre simuladores diferentes.

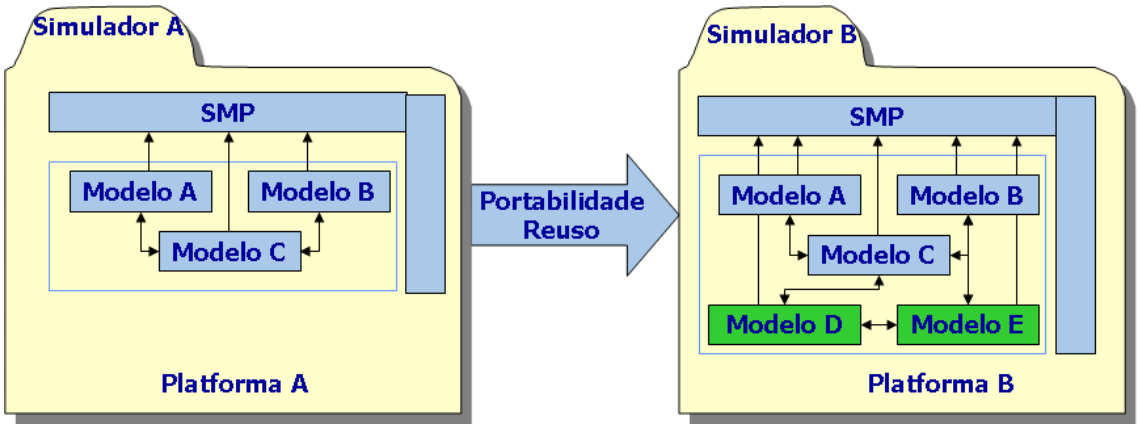


Figura 4.2 Esquema ilustrativo de portabilidade e reúso de modelos em simuladores no padrão SMP.

Esta abordagem permite o aproveitamento dos modelos entre as várias simulações que seguem as especificações do SMP. As especificações do SMP são muito úteis e vantajosas quando os programas, sejam os modelos e o contexto do simulador, estão sobre uma mesma plataforma. Aqui, todo o código a ser usado no contexto do simulador

deve estar ligado diretamente ao módulo principal da simulação e esta restrição está imposta nas especificações do SMP.

A aplicação desta abordagem, mostrada na Figura 4.2, pode ser usada em algum contexto de simulação, mas não permite que modelos legados desenvolvidos em outras plataformas, sejam instanciados local ou remotamente durante a execução da simulação.

Abordagem 2 – inclusão de chamadas a modelos remotos

A abordagem 2 trata da situação em que modelos legados são usados em um simulador que tenha que atender requisitos de interoperabilidade e execução remota de modelos. Neste caso, novas especificações são necessárias, inclusive a definição de um protocolo para realização da comunicação entre os modelos legados e o ambiente de simulação, uma vez que o SMP não contempla soluções para a comunicação via rede.

No padrão SMP, a ligação para *Montagem* e *Execução* do programa é baseada nas informações contidas no arquivo *Catalogue*. O *Catalogue* do SMP não possuí um marcador que informe em qual máquina o modelo será alojado e executado, nem qual o tipo de protocolo que será usado para comunicação entre os modelos e o ambiente de simulação. Neste cenário, o SMP não prevê facilidades para atender o requisito "que um modelo possa ser executado de forma remota via rede". Não existe na especificação do SMP uma forma de declarar que um modelo será executado remotamente. O SMP espera que as instâncias de execução do modelo estejam dentro do código do ambiente de simulação.

Nesta abordagem, o interessado cria as chamadas, no código principal do ambiente de simulação, a modelos remotos. Isso implica em modificar o código do ambiente de simulação toda vez que houver necessidade de incluir um novo modelo.

Abordagem 3 – interoperabilidade e distribuição dos modelos no padrão SMP

A terceira abordagem consta de adicionar um mecanismo de comunicação a todos os modelos que compõem o simulador, mantendo toda a estrutura e filosofia do padrão

SMP. Esta abordagem aproveita as características de manipulação das declarações dos modelos e do contexto de simulação do SMP. Esta abordagem é parte da proposta desta dissertação e é descrita com detalhes na seção a seguir.

Cabe observar que as declarações dos modelos de definição das camadas PIM e PSM para os modelos legados, em cada abordagem, são escritos manualmente nos arquivos *Catalogue* e *Assembly* para serem usadas por um simulador no padrão SMP.

4.3 Modelos legados em ambiente computacional distribuído

A Figura 4.3 mostra um contexto da simulação no qual o simulador é composto por modelos distribuídos remotamente em uma rede. Cada modelo possuí uma interface no padrão SMP, ilustrada como Interface SMP na figura. Dado que o padrão SMP não contempla informações sobre modelos remotos, resta saber como declarar as chamadas a tais modelos sem ter que alterar o código do simulador, como descrito na abordagem 3 (três). Os sinais de interrogação (?) no contexto da simulação indicam a ausência de especificação da plataforma de comunicação que permitirá a execução do simulador.

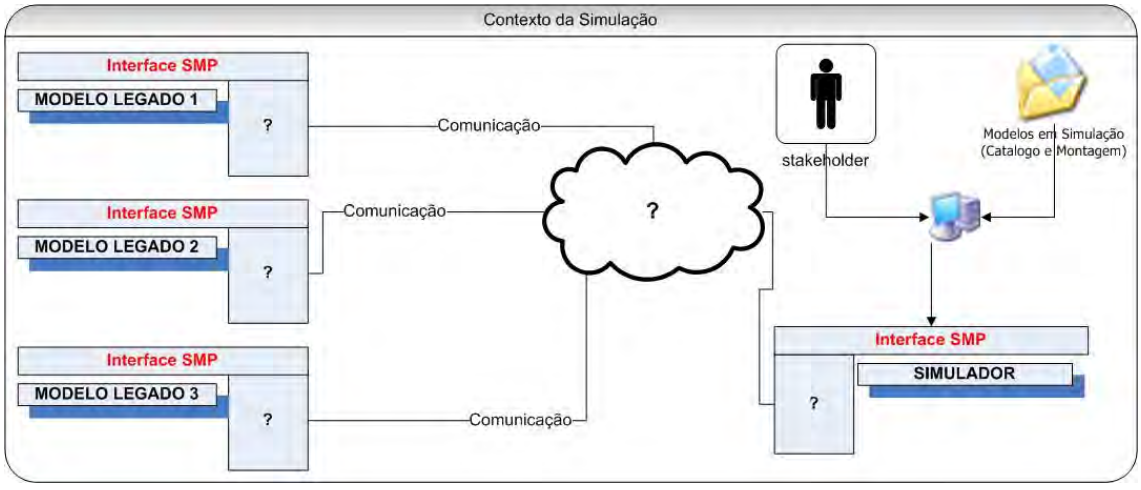


Figura 4.3 Comunicação com os modelos legados distribuídos para o padrão SMP.

Uma vez que os modelos são executados fisicamente em máquinas distintas, é proposta uma forma de trocar dados e objetos entre os modelos distribuídos em plataformas heterogêneas. A forma escolhida é baseada na plataforma de interoperabilidade

CORBA. A Figura 4.4 mostra o uso do CORBA no mesmo contexto de simulação ilustrado na Figura 4.3.

O motivo que levou ao uso da plataforma CORBA na abordagem proposta foi a necessidade de satisfazer o requisito de interoperabilidade do SMP e ao mesmo tempo integrar um modelo legado (rodando em sua plataforma original) em um simulador do padrão SMP. Nesta abordagem, o simulador tem seus modelos distribuídos, executados em máquinas distintas.

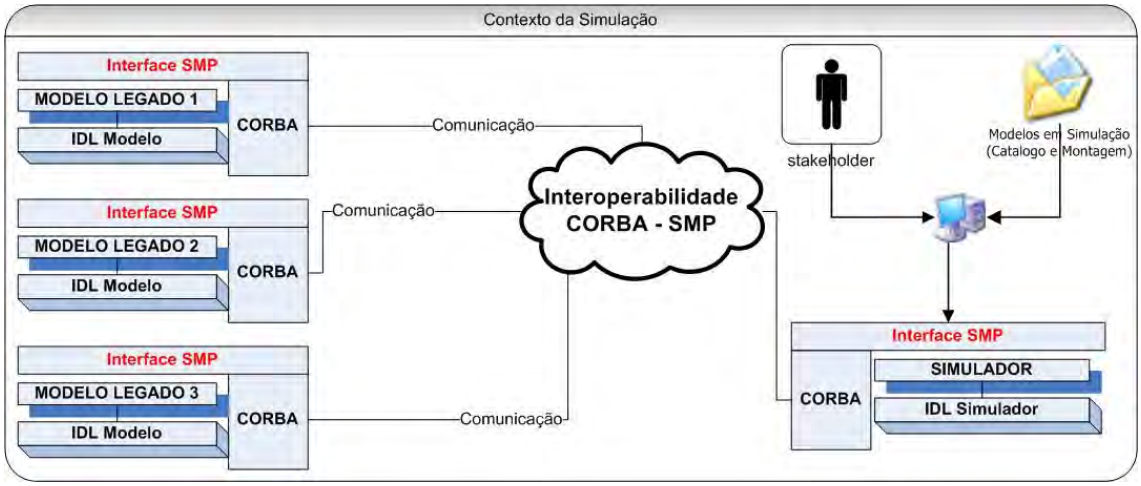


Figura 4.4 Interoperabilidade via a plataforma CORBA para o padrão SMP.

O mecanismo de comunicação entre os processos remotos via CORBA permite integrar os modelos legados independente da plataforma e da linguagem de programação.

Caso um modelo legado seja programado em uma linguagem que não permita especificações de comunicação em CORBA, deve-se criar um programa empacotador (do Inglês, "*Wrapper*") que servirá de ponte entre os processos que possuem o CORBA e aqueles que não possuem. Entretanto, esta técnica exige que o sistema operacional contenha mecanismos para troca de mensagens entre os processos.

Nesta abordagem a técnica de reengenharia também deve ser aplicada nos programas do modelo legado a fim de se obter as informações necessárias as interfaces públicas usadas para comunicação com outros modelos do simulador. Uma vez obtidos os

atributos e as características das interfaces, estas devem ser adicionadas ao mecanismo de comunicação dos modelos. A partir daí, pode-se gerar os arquivos SMDL e as definições PIM exigidas no padrão SMP. Assim, os modelos podem ser instanciados por meio de uma arquitetura distribuída, cujos objetos são executados de forma independente da plataforma de desenvolvimento do programa, do meio físico de comunicação usado para a troca de mensagens entre os processos e do sistema operacional onde será executado. Para o aproveitamento das declarações dos modelos no formato das especificações em SMP é necessário que o ambiente de simulação possua uma entidade que leia as informações sobre quais modelos serão usados. No contexto da simulação, estas informações são obtidas a partir dos arquivos *Catalogue* e *Assembly* que foram criados pela ferramenta geradora dos arquivos SDML conforme mostrado na Figura 4.5.

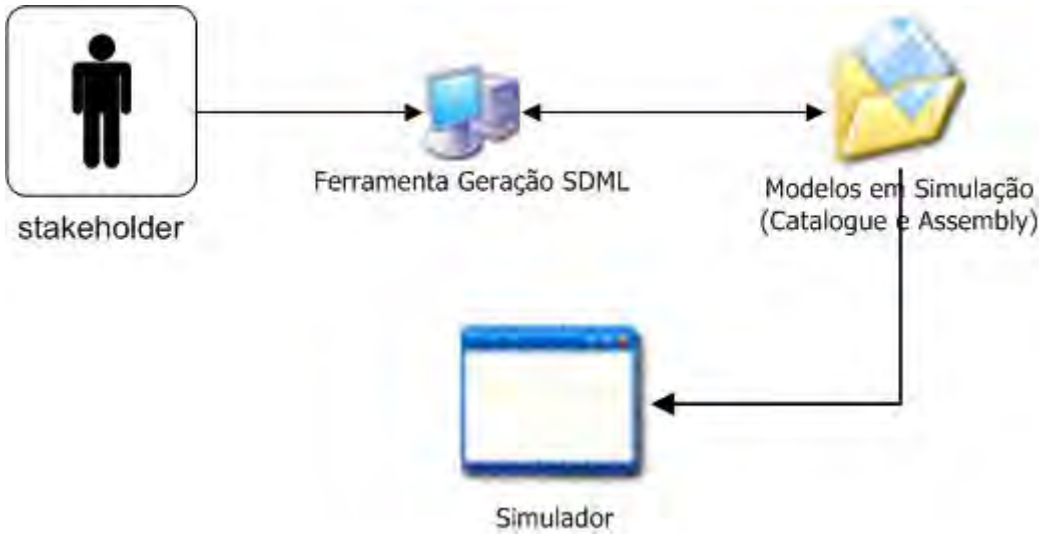


Figura 4.5 Geração dos arquivos de Catalogue e Assembly em SDML.

Para permitir que modelos legados façam parte do **contexto de simulação**, mantendo-se o padrão SMP, foram criados métodos especiais para os modelos legados e as funcionalidades padronizadas para a interoperabilidade no SMP, conforme mostrados na Figura 4.6.

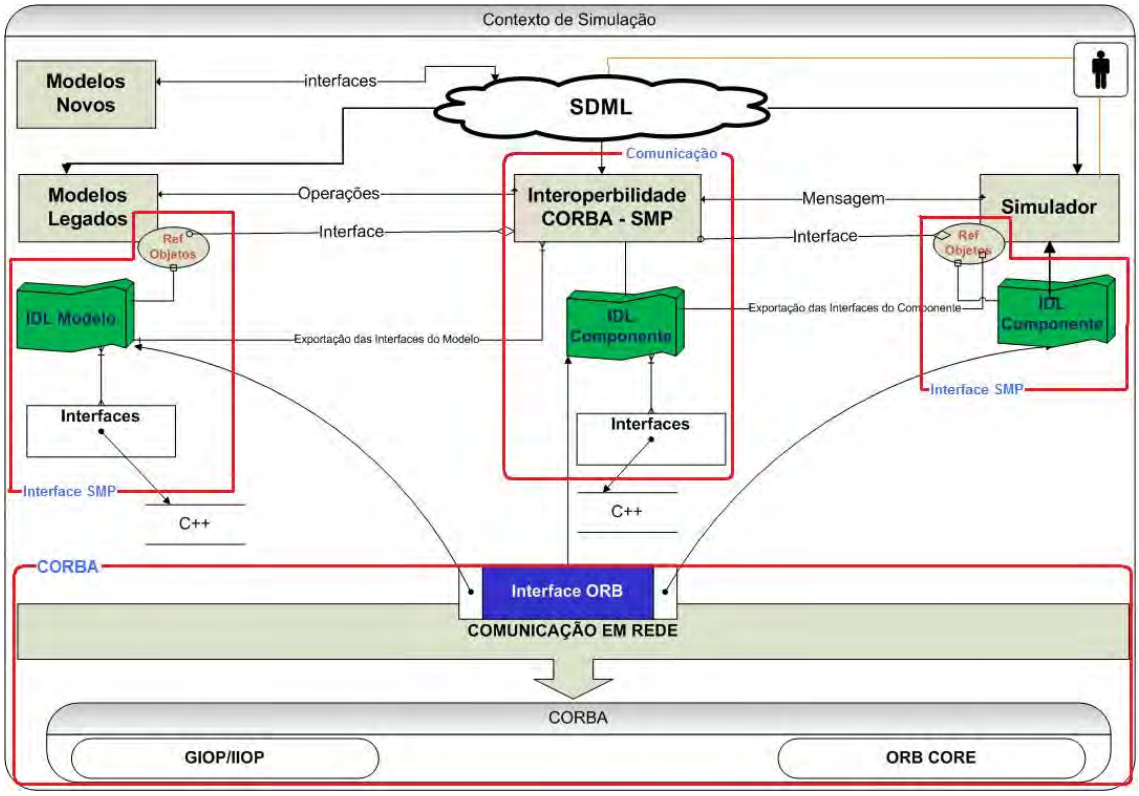


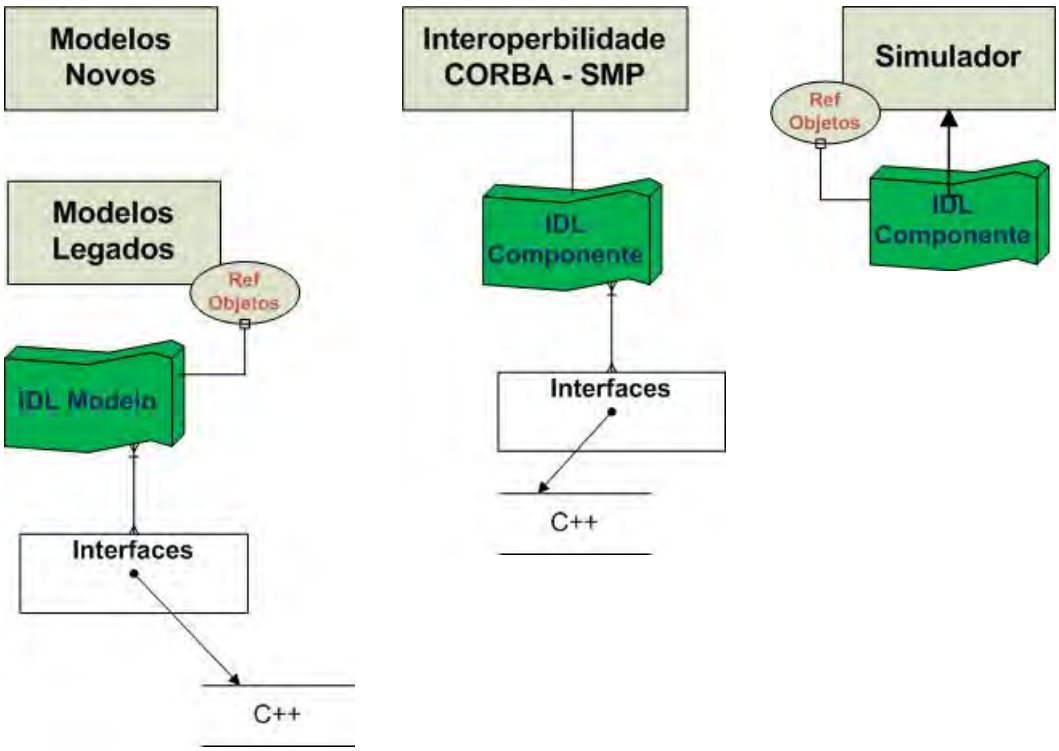
Figura 4.6 Esquema mostrando a interoperabilidade com uso de CORBA em um simulador do padrão SMP.

No esquema da Figura 4.6, existem quatro elementos com funcionalidades distintas: (i) os modelos novos, (ii) os modelos legados e suas interfaces exportadas, (iii) o mecanismo de interoperabilidade que permiti a ligação entre os modelos legados e o simulador e (iv) o simulador propriamente dito, cuja função é instanciar e gerenciar a execução da simulação.

A Figura 4.7 detalha estes elementos. O modelo legado e suas interfaces que são exportados para um arquivo no formato IDL (IDL Modelo), são mostrados na Figura 4.7 (A). A ligação entre os modelos legados e o simulador, detalhada na Figura 4.7 (B), usa tanto o arquivo IDL (IDL Componente) padronizado pelas especificações SMP (ESOC, 2005b), como a interoperabilidade CORBA - SMP que possuí os requisitos de comunicação entre os modelos instanciados para o simulador. O simulador, mostrado na Figura 4.7 (C), instancia e gerencia os elementos para realização da simulação e faz uso

do arquivo com as interfaces exportadas na IDL (IDL Componente) do modelo de componentes. (ESOC, 2005b).

O elemento Interoperabilidade CORBA – SMP da Figura 4.7 (B), foi definido a partir da integração das especificações SMP 2.0 C++ model development kit (ESOC, 2005b) e Simulation modeling plafatform - C++ Mapping (ECSS, 2011d) .Este elemento pode ser visto como um novo sistema independente que não estava declarado ou especificado nos memorandos técnicos do SMP atual.



(A) (B) (C)

Figura 4.7 Detalhes do esquema da interoperabilidade no padrão SMP.

ECSS (2011c) e ECSS (2011d) possuem um conjunto de declarações e especificações que descrevem como implementar as funcionalidades do SMP na programação dos modelos e do simulador.

Para manter os requisitos de interoperabilidade dos modelos legados e do simulador no contexto distribuído, foi necessário criar novos elementos de Gerenciamento da Execução dos modelos. Tais elementos incluem:

a) um leitor dos arquivos de configuração em SDML,

b) um novo atributo nas interfaces de comunicação dos modelos,

c) modificações no código e nas interfaces do "IDL Componente" (Figura 4.7 (B)).

As modificações descritas permitem a ligação entre os modelos legados e a simulação dentro de um contexto de simulação distribuída ainda em conformidade com o padrão SMP.

A Figura 4.8 ilustra um esquema de portabilidade e reúso de modelos em simuladores no padrão SMP, agora incluindo Interoperabilidade entre modelos legados distribuídos. A figura representa a solução da abordagem 3, a qual faz uso da abordagem 1 ilustrada na Figura 4.2. Os modelos A, B e C são compartilhados entre dois simuladores no padrão SMP usando a funcionalidade de comunicação incluída no SMP. Esse compartilhamento dos modelos em uma arquitetura distribuída só ocorre com a inclusão da interoperabilidade nas implementações físicas do SMP nos modelos e no simulador.

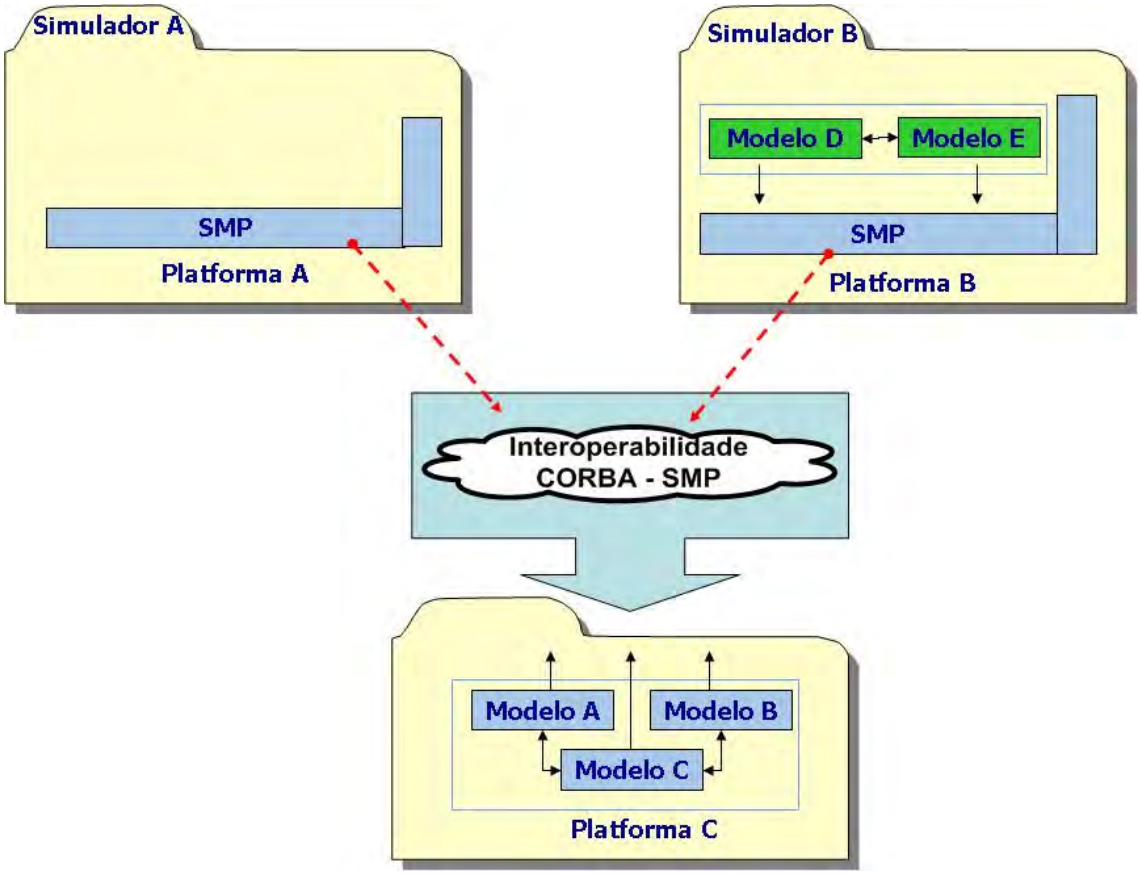


Figura 4.8 Esquema ilustrativo de portabilidade e reúso de modelos em simuladores no padrão SMP com Interoperabilidade entre modelos legados distribuídos.

4.4 Resumo

Neste Capítulo foram apresentadas as alterações na arquitetura do SMP necessárias para permitir a integração de sistemas legados aos simuladores que seguem o padrão SMP. Estas alterações adicionam à arquitetura a propriedade de interoperabilidade em um ambiente distribuído para simulação, permitindo a inclusão de modelos legados em um simulador de satélites.

CAPÍTULO 5

ESTUDO DE CASO: IMPLEMENTAÇÃO E AVALIAÇÃO

Esse Capítulo apresenta o desenvolvimento de um simulador simplificado usando os conceitos do padrão SMP e as soluções da abordagem 3 (três) para inclusão de modelos legados a um simulador de satélites. O simulador faz uso do elemento Interoperabilidade CORBA - SMP apresentado no Capítulo anterior para integração de modelos legados. O objetivo é demonstrar a viabilidade da solução proposta. De forma complementar, este simulador, um protótipo, serviu também para exercitar e melhor compreender as especificações do padrão SMP.

5.1.1 Descrição do estudo de caso

O desenvolvimento do simulador, segundo o padrão SMP, requer o uso de algumas ferramentas para criação dos programas e dos arquivos *Catalogue* e *Assembly*. Alguns desses programas e seus respectivos documentos, arquivos fontes, executáveis referentes ao SMP foram obtidos do sítio http://www.portal.vega.de/smp, com registro de usuário, enquanto, outros foram desenvolvidos pelo autor desta dissertação. Neste sítio foram obtidos vários exemplos de uso. No grupo de discussão do mesmo sítio, foram obtidas respostas sobre dúvidas sobre o SMP. Entretanto, depois de novembro de 2011, este sítio foi desativado por razões desconhecidas pelo autor desta dissertação.

Dois modelos legados foram usados neste estudo de caso, são eles: o modelo de órbita e o modelo de geração de energia do satélite, chamado de SAG. O primeiro modelo calcula o posicionamento do satélite enquanto que o segundo calcula os valores de geração de energia dos painéis solares e o seu consumo pelos equipamentos do satélite.

5.1.2 Modelo de Órbita

O modelo de posicionamento do satélite calcula a posição de um satélite em órbita e informa se uma referência geográfica está em sua visada. Com as informações de posicionamento das estações de rastreio em solo e das localizações das Plataformas de Coleta de Dados (PCD) este modelo informa se a estação e as PCDs estão em visada para comunicação com o satélite.

O modelo é baseado em uma biblioteca pública escrita em linguagem Pascal, chamada SGP4 (KELSO, 2000). Para permitir a visualização dos dados, este modelo foi encapsulado no formato Activex. O modelo obtém, a partir do nome do satélite, várias informações sobre o satélite entre elas a sua posição em solo e os pontos visíveis em sua visada. A linguagem usada para encapsulamento do modelo legado foi o Delphi 6 com compilador para o IDL/CORBA.

A declaração do modelo de órbita é feita de forma gráfica usando-se a ferramenta xSDML, como mostra a Figura 5.1. Na declaração, o modelo é visto como uma classe e possuí os atributos fLatitudeEstaçao, fLongitudeEstaçao, fAltitudeEstaçao e pSatelite e as interfaces que trocam informações com o simulador como exemplo a get_p_dados_pcd_visivel que recupera as PCDs visíveis em determinada passagem do satélite. A declaração completa do modelo em SDML (formatado em XML) é apresentada no Apêndice - A.1 Catalogue do Modelo de Órbita.

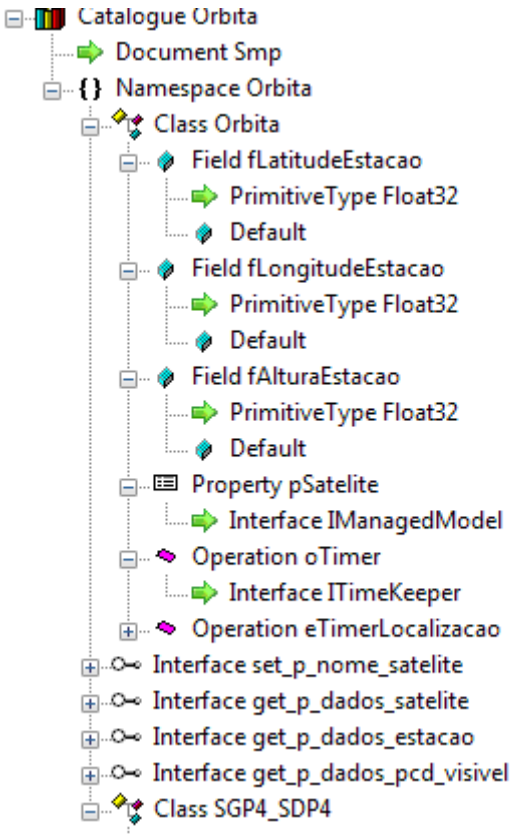


Figura 5.1 Trecho da declaração do modelo de órbita na ferramenta xSDML.

5.1.3 Modelo SAG

O modelo de geração de energia (SAG) informa a quantidade de energia gerada via um painel solar e a quantidade de energia consumida pelos equipamentos do satélite em um determinado tempo.

Este modelo legado recebe um conjunto de vinte e quatros parâmetros em um arquivo de entrada e fornece como saída o valor gerado e o valor consumido de energia do satélite através de uma interface no modelo.

A modificação deste modelo legado envolveu duas etapas. A primeira foi a compilação do código que calcula os dados geração de energia e de consumo. Este programa está escrito em Fortran99, e que não possuí uma interface de comunicação com outros modelos. A segunda foi a programação de um código em Delphi 6 que criou uma

interface de comunicação para exportar os dados obtidos do modelo SAG. Aqui foi usado o compilador IDL/CORBA e uma função em pipe entre o Delphi e este programa.

A declaração do modelo SAG, em SMDL usando a ferramenta xSDML, é mostrada na Figura 5.2. Uma declaração completa do modelo em SDML (em formato XML) é apresentada no Apêndice - A.2 Catalogue do Modelo SAG.

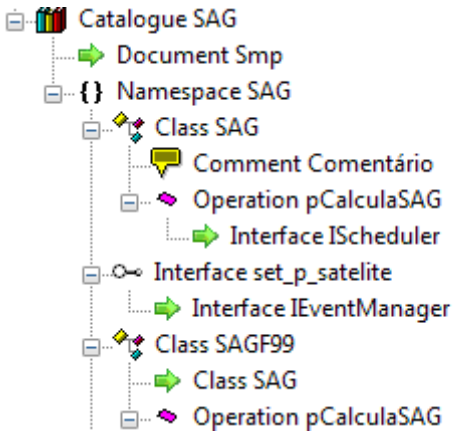


Figura 5.2 Trecho da declaração do modelo SAG em xSMDL.

Com as definições feitas em conformidade com o *Catalogue* do SMP foi possível integrar os modelos no formato *Assembly* do SMP e implementar as declarações dos modelos descritos no arquivo *Catalogue*.

Os modelos apresentados são executados como um processo independente e assim permitem o desenvolvimento do simulador como uma aplicação distribuída.

5.1.4 Ferramenta para a geração dos arquivos em SMDL

A criação do simulador no padrão SMP seguiu o ciclo de vida das declarações de um modelo em SMDL, como mostra a Figura 5.3.

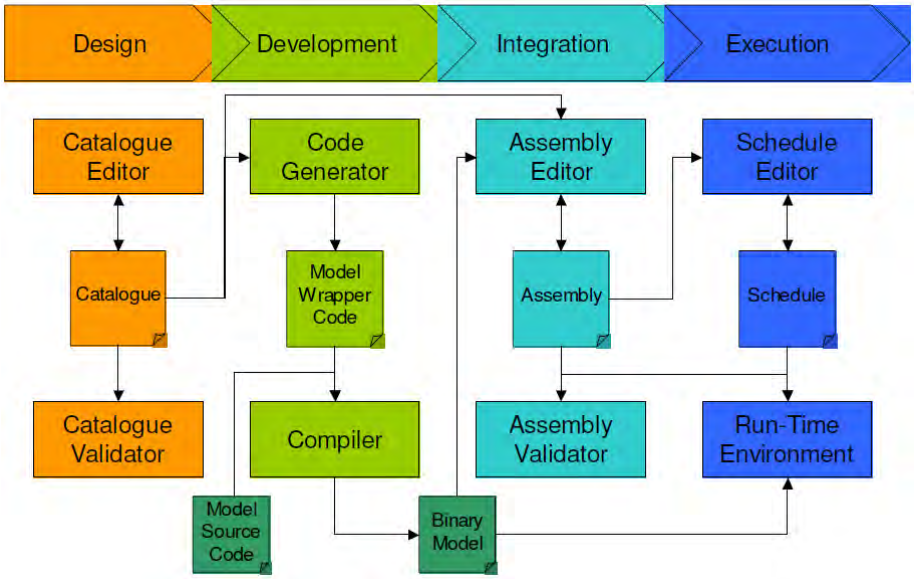


Figura 5.3 Ciclo de vida das declarações dos modelos em xSDML.

Fonte: ECSS. (2011b)

As declarações SMDL dos modelos usados no estudo de caso foram geradas com a ferramenta xSMDL. A ferramenta xSMDL é, de fato, um protótipo desenvolvido para esta dissertação a partir das especificações do metamodelo definido em ECSS (2011b).

A ferramenta gera as descrições, atributos e interfaces, dos modelos de forma independente da plataforma, usando os arquivos *Catalogue* e *Assembly* em SMDL de acordo com os requisitos do PIM do SMP. Os arquivos em SMDL serão lidos pelo simulador quando de sua execução. A ferramenta (xSMDL) que gerencia o ciclo de vida das declarações SMDL do modelo é mostrada na Figura 5.4, foi desenvolvida em Java, usando a plataforma Eclipse, podendo ser executada em qualquer sistema operacional que possua a máquina virtual Java.

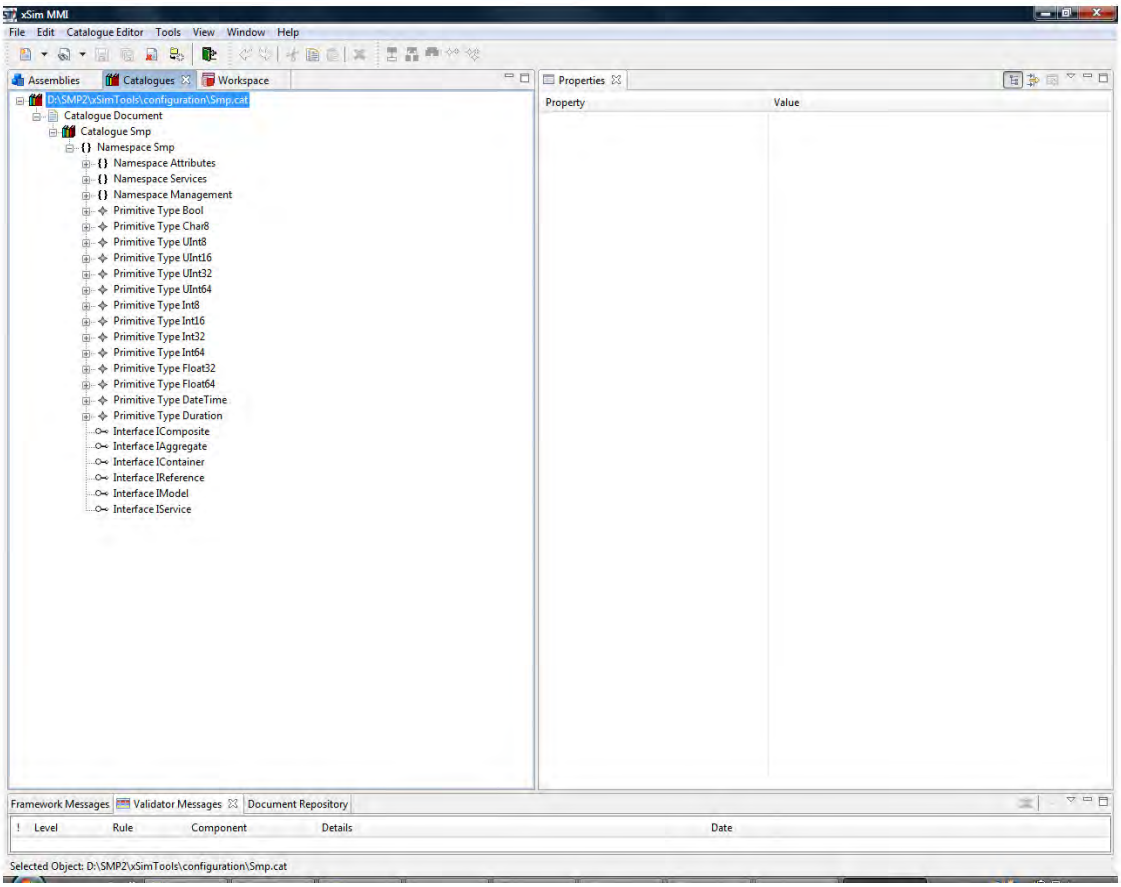


Figura 5.4 Tela da ferramenta xSDML.

5.1.5 Descrição do desenvolvimento do protótipo

O simulador simplificado usado no estudo de caso foi desenvolvido em linguagem Delphi com diretivas de compilação para IDL/CORBA. O principal requisito deste simulador é se comunicar com o elemento Interoperabilidade CORBA - SMP, a partir das interfaces padronizadas. A criação seguiu a abordagem de orientação a objeto, já que não existe um método especificado em SMP para criação de simuladores.

A Figura 5.5 ilustra uma instanciação da interoperabilidade em SMP para o simulador simplificado que integra os modelos de órbita e SAG.

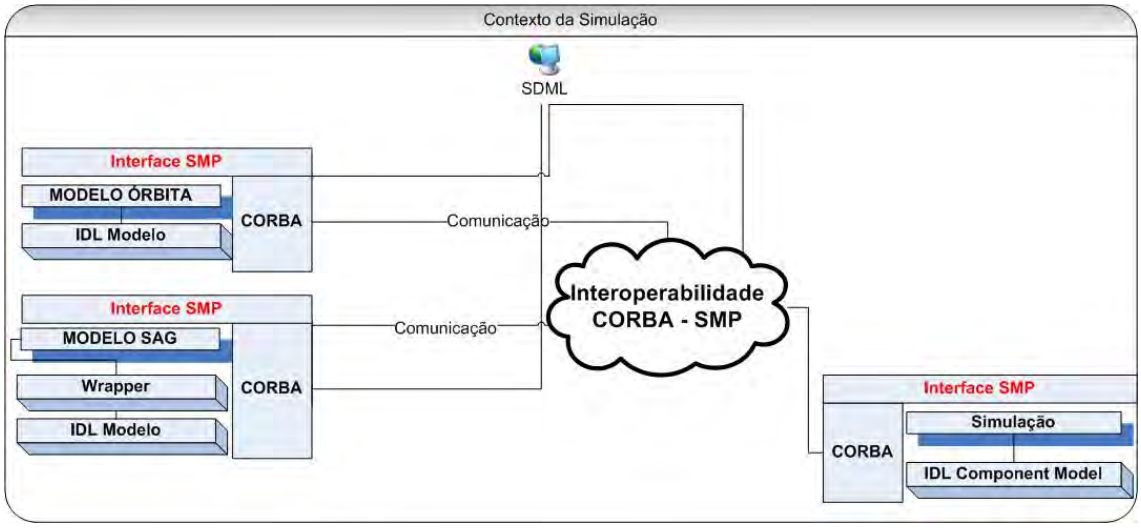


Figura 5.5 Instanciação da interoperabilidade em SMP para o simulador simplificado.

Em ambos os modelos foi aplicada a abordagem 3 (três) para sua integração, descrita na seção 4.2. No modelo de Órbita as interfaces são exportadas pelo programa, pois ele usa a tecnologia de orientação a objeto. Enquanto que o modelo SAG, por não possuir tecnologia de orientação a objeto, foi criado um "Wrapper" que empacota as informações e as envia para o elemento Interoperabilidade CORBA - SMP.

O SMP permite que um mesmo modelo possa ser instanciado para diferentes simuladores graças à separação das definições em PIM e PSM. No protótipo desenvolvido, os modelos foram configurados para os satélites SCD1, SCD2 e do CBERS2B. A arquitetura física é apresentada na Figura 5.6.

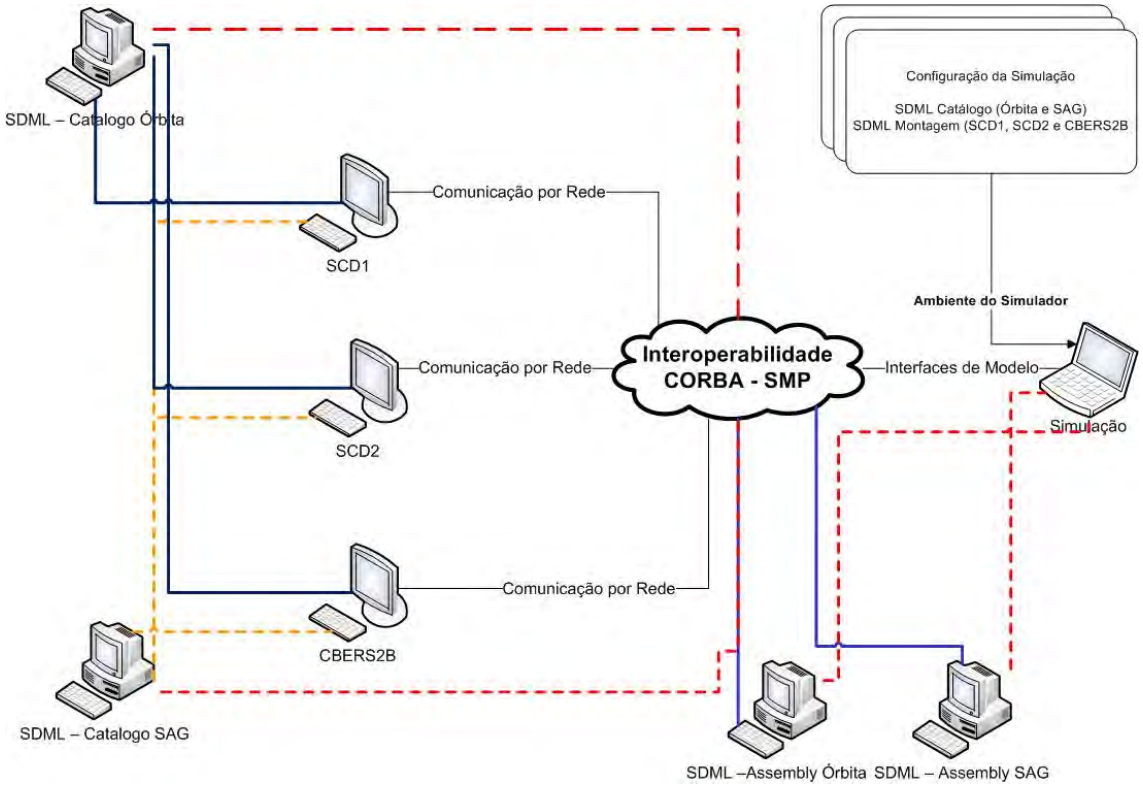


Figura 5.6 Arquitetura física do estudo de caso.

5.1.6 Órbita

Foram aplicadas as especificações do arquivo *Catalogue* do modelo Órbita aos três satélites descritos acima, SCD1, SCD2 e CBERS2B . As figuras a seguir mostram como foi aplicado o formato do arquivo *Assembly* do padrão SMP para cada um dos satélites.

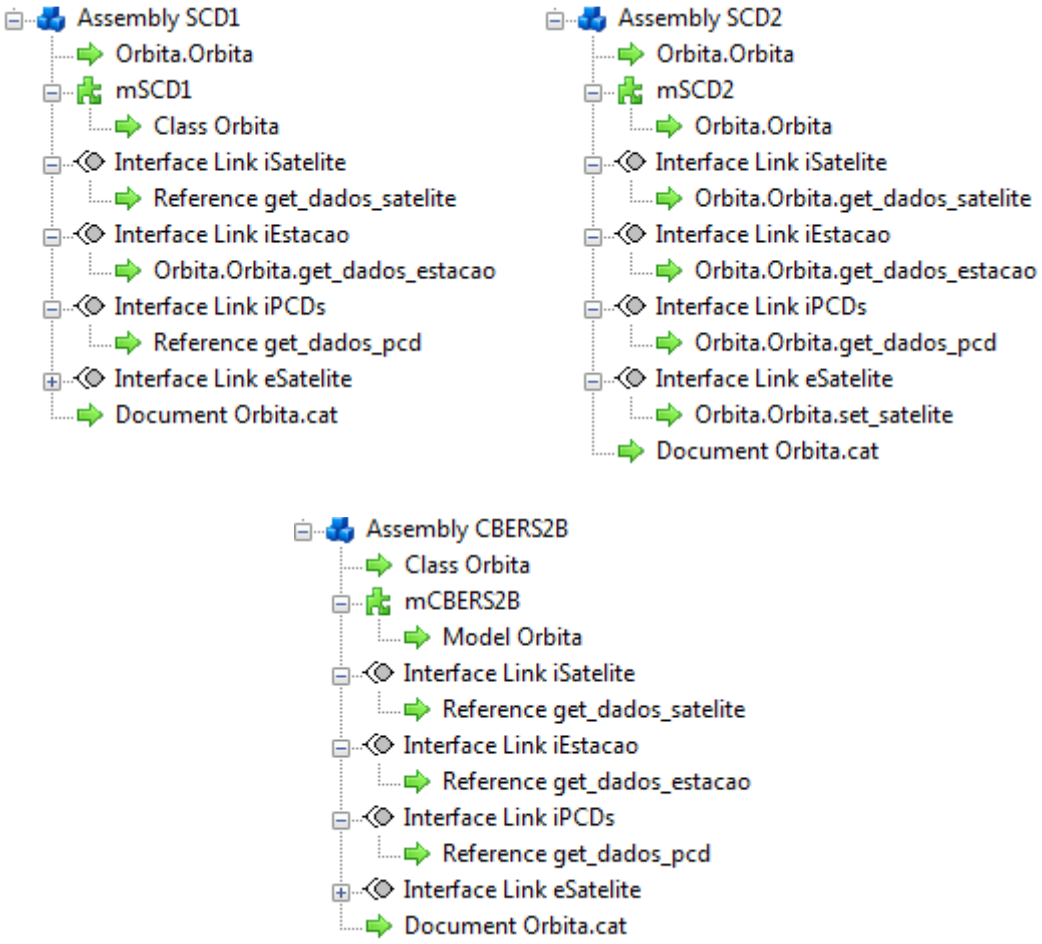


Figura 5.7 Modelo de Órbita instanciado para diferentes satélites no arquivo Assembly.

Na Figura 5.7 é mostrado um trecho de cada instância da Órbita dos modelos de satélite em SDML. Mais informações sobre o conteúdo dos arquivos *Assembly* estão no Apêndice A, seção A.3 Assembly do Modelo de Órbita. Cada satélite que é instanciado tem associado um conjunto de atributos próprios e únicos que o identifica no modelo de órbita.

5.1.7 SAG

No modelo SAG foram feitas as mesmas aplicações que no modelo Órbita, ou seja, o modelo SAG foi instanciado para os três satélites SCD1, SCD2 e CBERS2B. O resultado nos arquivos *Assembly* é mostrado não Figura 5.8.

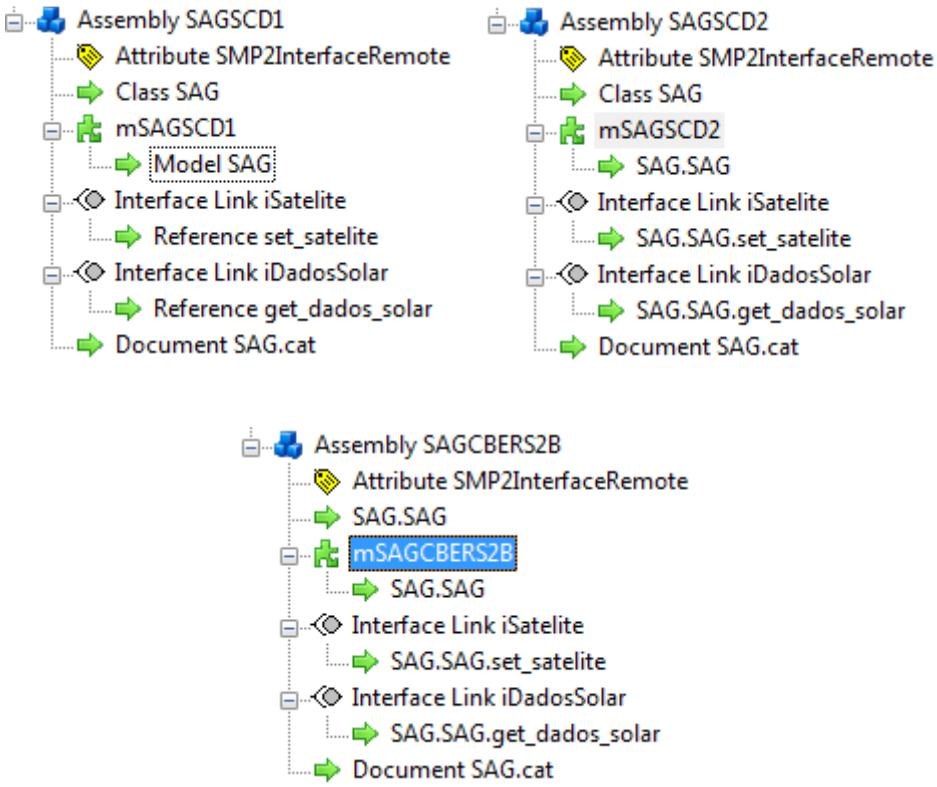


Figura 5.8 Modelo SAG instanciado para diferentes satélites no arquivo Assembly.

Informações sobre o conteúdo dos arquivos *Assembly* estão no Apêndice A em A.4 Assembly do Modelo SAG. Cada satélite que é instanciado tem associado um conjunto informações sobre os painéis solares, o consumo de energia e uma referência que o identifica no modelo do SAG.

A Figura 5.9 ilustra um trecho de código extraído da especificação do SMP a fim de simplificar e explicar como a abordagem 1 da seção 4.2 é aplicada em um simulador que usa o SMP.

A classe SmpSample é instanciada dentro do próprio código do simulador e depois o controle é passado ao gerenciador de interfaces do SMP o qual assume o controle sobre a instância na mesma plataforma, essa forma representa a aplicação da abordagem 1 (um) descrita na seção 4.2.

```
int main(int argc, char* argv())
{
   (void)argc;     // Avoid compiler warning
   (void)argv;     // Avoid compiler warning
   // Test SmpSample
   {
      // Create an instance of the sample model
      SmpSample* smpSample = new SmpSample("SmpSample", "This is an Smp
sample model", NULL);
      TestSample(smpSample, "SmpSample");
      delete smpSample;
   }
   // Test SmpManagedSample
   {
      // Build up a small tree of managed components
      Examples::SmpManagedSample* smpManagedSample;
      Examples::SmpSample* smpChild1;
      Examples::SmpSample* smpChild2;
      smpManagedSample = new
Examples::SmpManagedSample("SmpManagedSample", "This is an Smp sample
model", NULL);
      smpChild1 = new Examples::SmpSample("SmpChild1", "This is an Smp sample
child", smpManagedSample);
      smpChild2 = new Examples::SmpSample("SmpChild2", "This is an Smp sample
child", smpManagedSample);

      smpManagedSample->MyManagedContainer->AddComponent(smpChild1);
      smpManagedSample->MyManagedContainer->AddComponent(smpChild2);

      TestManagedSample(smpManagedSample, "SmpManagedSample");

      delete smpManagedSample;
      delete smpChild1;
      delete smpChild2;
   }
```

Figura 5.9 Exemplo do SMP usando a abordagem 1 da seção 4.2.

Um modelo declarado no arquivo *Assembly* deve ser instanciado na mesma plataforma onde se encontra o simulador. Como o SMP não permite processamento distribuído, para contornar esta deficiência foi criado um marcador que informa em qual máquina deverá ser instanciado o modelo e qual o tipo de protocolo de comunicação deverá ser usado entre os modelos e o simulador.

Para efetuar uma mudança nas especificações do SMP deve-se modificar as especificações e os esquemas do padrão SMP dos arquivos UML. Como estes arquivos não são disponibilizados nos fontes documentais da especificação, as mudanças na base das especificações e depois a sua propagação nos documentos gerados por esta base são inviabilizadas. Entretanto, dado que as especificações geradas para os modelos estão em XML é possível criar um marcador pré-definido para conter as informações necessárias para comunicação entre os modelos e também para a simulação.

Nesta dissertação foi definido que na especificação em XML do SMP, o marcador com a denominação de '*Attribute'* e seu campo denominado de '*Description'* será inferido um texto com informações de como o modelo definido no arquivo Assembly deverá ser instanciado de forma remota. Neste campo, um texto informará à máquina que irá instanciar o modelo, qual o tipo de comunicação será usado e a forma como o modelo será instanciado, Figura 5.10. Com estas informações fornecidas pelo arquivo SMDL do *Assembly*, Figura 5.11, o gerenciador de modelos em SMP poderá administrar os modelos remotos sem a necessidade de nova compilação a cada modelo que for adicionado a simulação.

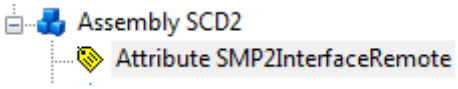


Figura 5.10 Exemplo do atributo de interface de comunicação.

```
<Metadata xsi:type="types:Attribute" Id="SMP2InterfaceRemote"
Name="SMP2InterfaceRemote">
   <Description>Host='150.163.17.174' Protocol='Corba' Kind='Static'</Description>
 </Metadata>
```

Figura 5.11 Trecho em XML da especificação do atributo de comunicação.

A regra de criação deve ser exatamente a seguinte: o tipo deve ser definido como '*Attribute'*, o nome e o Id deve ter o conteúdo '*SMP2InterfaceRemote*'. No marcador '*Description'* o texto deve ser fornecido na seguinte sequência: (1) 'Host=' endereço da máquina onde será instanciado o modelo, (2) 'Protocol=' tipo do protocolo de comunicação (atualmente somente o tipo CORBA) e (3) 'Kind=' tipo de instancia do modelo que pode ser 'Static' todas as informações são obtidas de um arquivo IDL do CORBA ou 'Dynamic' que recupera as informações do modelo em tempo de execução.

5.1.8 Componente de modelos e ambiente de execução

Esta seção apresenta a definição e o desenvolvimento do módulo que fornece ao SMP a interoperabilidade.

ECSS (2011c) apresenta um arquivo no formato IDL/CORBA com as especificações em PIM das interfaces que administram as trocas de mensagens entre a simulação e os modelos simulados. Este documento providencia os conhecimentos básicos para a implementação física dos componentes que fornecem suporte ao padrão SMP. Uma das melhorias apresentadas neste documento foi a de padronizar o conceito de projeto baseado em componentes para a simulação. Três tipos de componentes fazem parte da execução da simulação: o ambiente de execução da simulação, a simulação de serviços e os modelos.

ESOC (2005b) descreve cada interface e o ECSS (2011d) mostra a sua programação, ou seja, o PSM, que utiliza a linguagem de programação ANSI/ISO C++. A proposta dos documentos é fornecer suporte técnico ao desenvolvimento das especificações SMP em conformidade com a linguagem ANSI/ISO C++ criando uma biblioteca denominada de C++ MDK. O MDK não é parte integrante das especificações do SMP, mas providencia de maneira fácil a migração das implementações que definem as interfaces dos modelos para o desenvolvimento de um ambiente de execução da simulação padronizado pelo o SMP.

O MDK é gerado em formato de biblioteca, que só pode ser usado no ambiente de execução da simulação se for *linkado* em tempo da geração do executável do ambiente. Este procedimento tem a desvantagem de quando alguma modificação for feita no MDK, pois se o código programado da simulação for grande e complexo isto pode dificultar a sua manutenção. O MDK não possuí a funcionalidade de um leitor de arquivos com as definições dos modelos em SMDL, a tarefa para a leitura destes arquivos e a configuração dos modelos fica a cargo da programação do ambiente.

Nas especificações que descrevem a implementação do MDK existem os meios necessários a criação da comunicação entre modelos legados e o simulador, pois as declarações sobre o modelo em SMP são independentes de qualquer código associado a uma simulação e de execução própria no sistema operacional. As especificações das interfaces descritas em ESOC (2005b) podem ser integradas, transportando o conceito de separação entre as interfaces e a execução do código do modelo, para isso é necessário compilar com as diretivas de compilação que usam o IDL/CORBA e o ANSI/ISO C++. O sistema gerado para executar a interoperabilidade poderá ser executado em qualquer computador que esteja em alguma rede e disponibilizar a qualquer simulação as interfaces que executam as funcionalidades de um modelo.

Para ser uma aplicação independente de qualquer código de simulação e executar em rede é necessária a adição de dois requisitos importantes: a interface que declara quais arquivos SMDL serão usados na interoperabilidade e o leitor dos arquivos SDML escritos em XML.

Para a criação de uma nova interface simplesmente basta incluir novas declarações no arquivo IDL/CORBA descrito em ESOC (2005b), Figura 5.12:

```
Module Smp
{
    Interface ISDML : IObject
    {
        exception InvalidNameSDML
        {
            String8 fileName;
        };
        IComponent CreateObjectModel(in String8 Name, in UInt16);
        IObject ObjectModel();
    };
};
```

Figura 5.12 Declarações para a interoperabilidade no SMP.

Com as declarações qualquer programa que execute uma simulação pode chamar o processo de Interoperabilidade CORBA - SMP, que estará sendo executado em algum computador na rede e instanciar e configurar um modelo a partir dos arquivos SMDL criados para um determinado simulador. Quando o modelo for instanciado suas interfaces serão exportadas pela rede. Para se ter acesso a estas interfaces é necessário apenas compilar o ambiente de execução da simulação com o arquivo IDL/CORBA descrito em ESOC (2005b).

Para leitura dos arquivos em SDML é necessário incluir um trecho de código que leia as "*tags*" que estão formatadas em XML e configurar as várias interfaces que serão necessárias para a execução dos modelos. O instanciamento dos modelos declarados pode ser de forma distribuída conforme as informações.

5.1.9 Conversão ANSI/ISO C++ para IDL/CORBA

A ferramenta xSDML permite gerar, a partir das declarações do modelo do arquivo *Catalogue*, o código em ANSI/ISSO C++ das interfaces declaradas do modelo. Mas este código só pode ser ligado a modelo desenvolvido em linguagem ANSI/ISO C++.

Uma solução é portar as especificações que são feitas no SMP, que por característica própria possuem um vínculo com a linguagem ANSI/ISO C++, para outras linguagens de programação. O script CtoIDL, desenvolvido para este trabalho, permite que as declarações do arquivo .h, geradas pelo xSDML, sejam convertidas para um arquivo IDL/CORBA com as relações entre as interfaces em ANSI/ISO C++. Com os arquivos de declarações é possível compilar programas feitos em outra linguagem e que possuam diretivas de compilação que permitam o uso do CORBA.

5.1.10 Aplicação do estudo de caso

Para fazer as avaliações sobre o estudo de caso foram feitos programas de testes e protótipos em um ambiente de laboratório. As linguagens usadas foram o Delphi 6, versão 2001, para os modelos Órbita e SAG, o simulador e para a interoperabilidade CORBA - SMP utilizou-se o Microsoft Visual Studio C versão 6.0. Todo código executável foi gerado para a plataforma Windows. Para o ORB, que é o servidor de objetos do CORBA, foi usado o Visbroker for C++ versão 04.00.02.C1.16 na plataforma Windows Vista.

Durante o processo de desenvolvimento dos programas foram estudadas as várias situações técnicas encontradas na sua implementação, analisadas as vantagens e desvantagens da aplicação desenvolvida e o incremento funcional das novas características aplicadas ao SMP. Os resultados obtidos do laboratório, em forma resumida, estão divididos em dois grupos: o *desenvolvimento* (modelos, simulador e a interoperabilidade) e a *execução* (ORB, rede de comunicação e sistema operacional) dos programas.

Os comentários nessa seção devem servir como referência aos desenvolvedores que desejarem replicar este trabalho em suas simulações novas ou legadas e para antecipar os requisitos necessários às mudanças dos programas que compõem o simulador.

5.1.11 Orientações para o desenvolvimento de um simulador distribuído em consonância com o padrão SMP

Esta seção apresenta observações e orientações importantes para o desenvolvimento do simulador segundo padrão SMP:

- o desenvolvimento deve ser feito em linguagem orientada a objetos. Não é possível aplicar este trabalho em simuladores não orientados a objetos;

- para permitir os aspectos de distribuição dos modelos é preciso ter um sistema operacional com funcionalidades da plataforma CORBA, caso contrário não será possível aplicar este trabalho;

- a capacidade de processamento (CPU, unidade de disco, memória, etc.) do computador afeta diretamente o desempenho do simulador;

Com algumas modificações e a inclusão de objetos instanciados é possível programar as funcionalidades necessárias para o uso desse trabalho em simuladores já existentes e que usam a orientação a objetos.

Com relação aos modelos pode-se dizer que:

- independente de uma linguagem de programação, os modelos que não foram desenvolvidos usando orientação a objetos podem ser usados na simulação via um programa denominado de "*Wrapper*" que empacota as informações do modelo que não foi desenvolvido com a tecnologia de orientação a objetos e as transmite para o elemento Interoperabilidade CORBA - SMP desse para o simulador.

- para o desenvolvimento em orientação a objetos do "*Wrapper*" é necessário que o sistema operacional possua os mecanismos de comunicação entre processos, pois só assim um programa orientado a objetos pode se comunicar com outro que não seja;

- o aproveitamento dos modelos legados é feito com a programação das interfaces necessárias a interoperabilidade, declaradas em SDML nos arquivos *Catalogue* e *Assembly*;

- como as interfaces estão declaradas no arquivo *Catalogue* em SDML e são exportadas em IDL/CORBA pela ferramenta que mantém o catálogo, é necessário incluir este arquivo exportado na compilação do modelo;

- um modelo é totalmente dependente das plataformas que possuam a arquitetura CORBA;

- um modelo depende da capacidade de processamento do computador e da velocidade da rede de comunicação.

Com relação à comunicação e a interoperabilidade com os modelos legados pode-se dizer que esta solução:

- é totalmente dependente do sistema operacional Windows;

- somente executado em plataformas que possuam a arquitetura CORBA;

- depende da capacidade de processamento do computador, da velocidade da rede de comunicação e do volume de dados transmitidos pela rede;

- para o atendimento a uma requisição de simulação é necessário que o processo principal possa criar processos filhos que atendam individualmente cada conexão solicitada pelo simulador.

5.1.12 Sobre a execução do protótipo

O simulador simplificado foi executado em um computador ACER Aspire 5100 com o processador AMD Turion™ 64 X2 Mobile Tecnology TL-56 1.80 GHz, sistema operacional Windows Vista™ Home Premium em 32 bits com Service Pack 2 e o Visbroker for C++ versão 04.00.02.C1.16 com o servidor CORBA.

Os modelos foram executados em um computador DELL Latitude E6520 com processador Intel® Core™ i7-2720QM CPU @2.20 GHz, sistema operacional Windows 7 Profissional em 64 bits com o Service Pack 1 e o Visbroker for C++ versão 04.00.02.C1.16 com o servidor CORBA. Para que cada modelo fosse executado em máquinas distintas foram configuradas três máquinas virtuais e para a sua criação foi utiliza a ferramenta Oracle VM Virtualbox. As principais características destas máquinas foram: sistema operacional Windows XP com 1 Gytes de memória RAM e 100 Mbytes de disco.

Na Figura 5.13 é mostrado o formulário e os valores gerados pelo modelo que simula a órbita de um satélite, neste caso, configurado para o satélite SCD1.

Cabe observar que o mesmo modelo poderia ser configurado para simulação dos satélites SCD2 e CBERS2B.

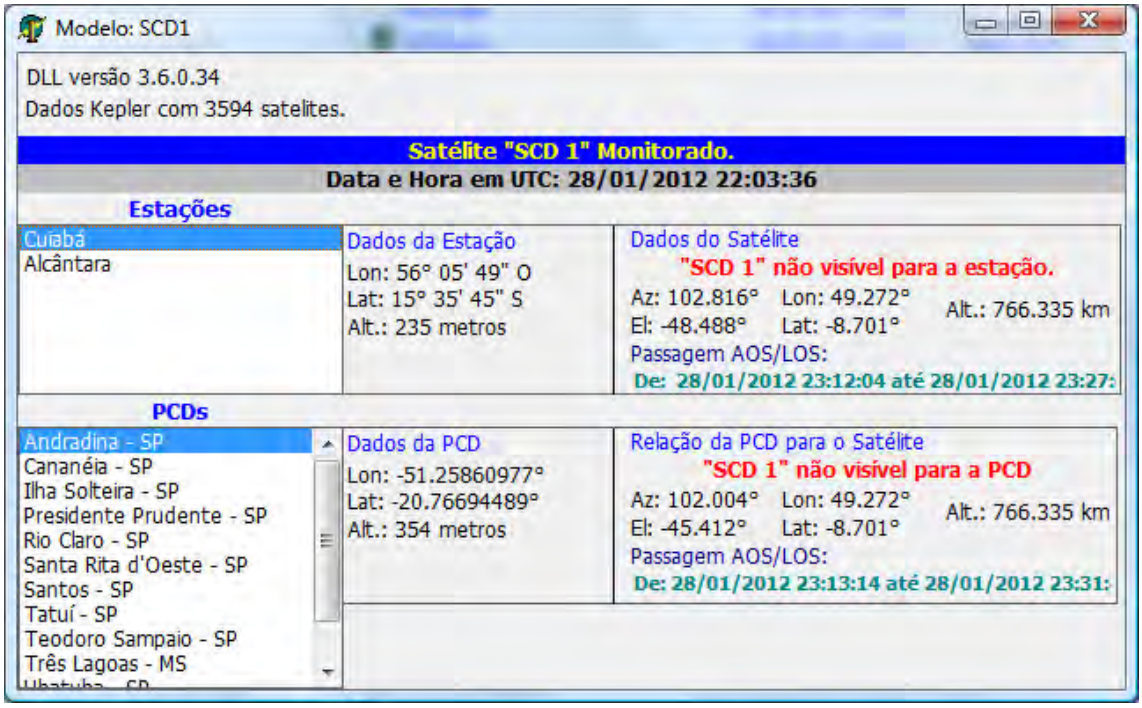


Figura 5.13 Programa do Modelo de Órbita.

Na Figura 5.14 é mostrado o modelo SAG. Os dados apresentados nesta figura são exibidos por um programa "*Wrapper*".

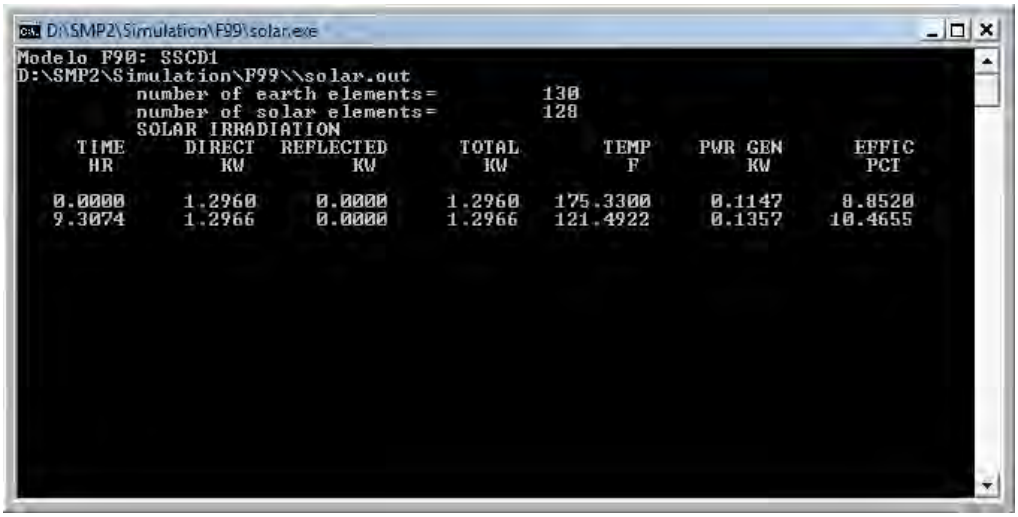


Figura 5.14 Programa do Modelo SAG.

Na Figura 5.15 é mostrada a tela do simulador quando os modelos ainda não estão sendo executados.

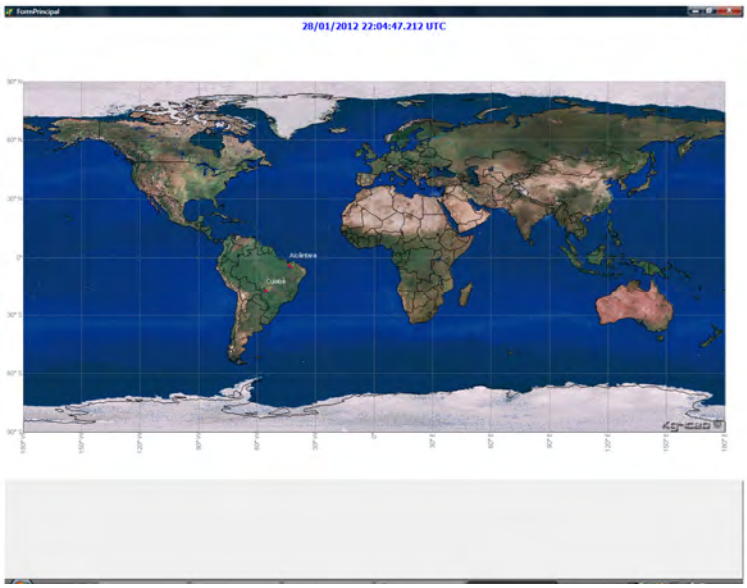


Figura 5.15 Tela do simulador sem os modelos.

Na Figura 5.16 é mostrada a tela do simulador com todos os modelos em execução.

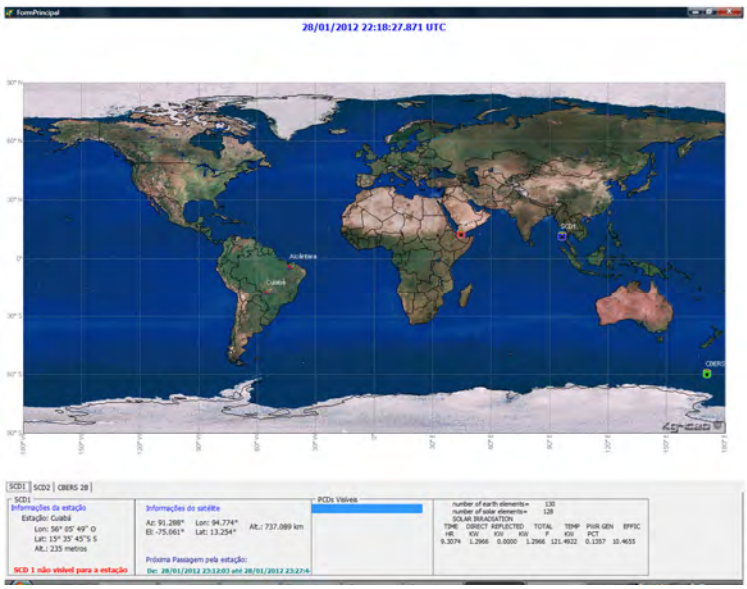


Figura 5.16 Tela do simulador com os modelos em execução.

A Figura 5.17 mostra o elemento Interoperabilidade CORBA –SMP, programa chamado de SMP.exe, e seus respectivos processos filhos que atendem as várias requisições dos simuladores.

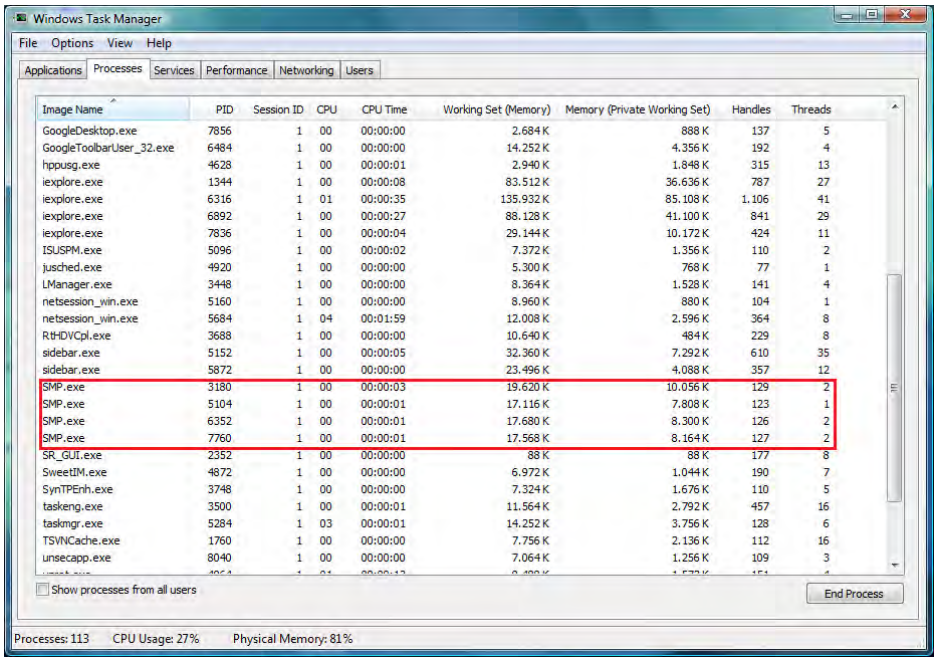


Figura 5.17 Interoperabilidade CORBA - SMP em execução.

Sobre a execução podemos dizer que:

- os modelos não são instanciados de modo automático, precisam ser instanciados manualmente em cada computador em que são executados;
- para que cada modelo ou simulador possa fazer comunicação, um ORB da CORBA precisa estar instalado no computador em que for executado;
- um gerenciamento de falhas e erros deve existir para que retardos de comunicação e problemas de execução dos programas possam ser recuperados ou regenerados durante o tempo de execução sem a perda da fidelidade da simulação.

5.1.13 Resumo

Este estudo de caso permitiu avaliar e demonstrar os usos dos processos e ferramentas que permitem a um simulador no padrão SMP se comunicar com modelos legados de simulação de forma distribuída.

CAPÍTULO 6

CONCLUSÕES, LIÇÕES APRENDIDAS E TRABALHOS FUTUROS

6.1 Conclusões

Nessa dissertação propôs-se um mecanismo de comunicação entre modelos legados para simuladores de satélites como uma extensão a característica de interoperabilidade do padrão de desenvolvimento de simuladores de satélites chamado SMP, proposto pela Agência Espacial Européia (ESA).

O mecanismo de comunicação proposto representa uma melhoria no padrão SMP de forma que não existe a necessidade de recriação do código fonte de modelos existentes. Um modelo legado projetado para uma dada plataforma pode ser reaproveitado para uma nova plataforma com a adição de interfaces de comunicação. Este mecanismo permite ao simulador incorporar as características de um programa distribuído e interoperável, integrando modelos legados.

O estudo de caso, que incluiu a implementação de um simulador simplificado, integrou dois modelos legados e permitiu demonstrar a aplicabilidade dos conceitos do SMP. Os modelos legados utilizados foram escritos em linguagens de programação distintas e em períodos diferentes. O estudo ilustrou a aplicabilidade dos mecanismos de comunicação aos modelos, quando na forma de programas executáveis.

A reutilização dos modelos legados pelo uso de tais mecanismos de comunicação possuí um papel de suporte aos projetos de simulação. No entanto, ainda existe muito trabalho a ser feito, notadamente quanto ao uso de padrões.

Novos padrões em simulações que requeiram um elevado grau de integração tendo como foco principal o estado da arte em tecnologia de interoperabilidade ainda estão indisponíveis, mas são promissores.

O trabalho desenvolvido nessa dissertação apresenta limitações quanto à flexibilidade para a utilização em outras plataformas computacionais que não tenham o protocolo CORBA bem como os modelos que não se tenha o código fonte.

Com o reúso e a interoperabilidade dos modelos legados em projetos de simulação de satélite terá um aumento de produtividade e redução de custos, pois o tempo será reduzido com o aproveitamento dos programas já desenvolvidos.

Como **vantagem** do mecanismo de comunicação aplicado a modelos legados de simulação tem-se que:

- permite que os modelos legados de simulação possam ser adicionados, atualizados ou removidos de um ambiente de simulação com pouco ou nenhum impacto para o simulador;
- propicia maior agilidade e segurança na troca de informação entre os modelos legados;
- permite o compartilhamento dos modelos legados entre várias simuladores;
- facilita o compartilhamento dos dados e informações entre os modelos legados e o simulador.

6.2 Lições aprendidas com o estudo de caso

O resultado final do estudo de caso foi a criação de um simulador, distribuído e conforme o SMP, que integrou dois modelos legados de simulação, o modelo de Órbita e o modelo SAG.

Para o desenvolvimento deste simulador, primeiramente foi desenvolvida uma ferramenta para gerar as declarações das interfaces em XML, limitada a criação das declarações nos arquivos *Catalogue* e da *Assembly*.

Em seguida, foi desenvolvida uma biblioteca de programas para leitura das especificações no arquivo *Catalogue*, no formato XML. O resultado desse trabalho foi a concepção de um módulo independente que pode ser usado por todos os programas que fazem parte da simulação, seja os modelos legados, seja as demais partes do simulador.

A execução dos modelos legados como parte de simuladores é feita a partir da união de: (i) uma biblioteca, (ii) um arquivo IDL que possuí as declarações das interfaces de comunicação do simulador com os modelos incluídos no código fonte do simulador. Assim, o simulador fica dependente das mudanças em alguma de suas partes internas (adicionadas ao código).

Para resolver este problema foi criada a Interoperabilidade CORBA - SMP com as especificações técnicas do SMP independente da ligação com o código fonte do simulador. Com isso a comunicação entre os modelos legados e o simulador é feita a partir de algum computador na rede.

Graças aos programas e bibliotecas desenvolvidos no contexto desta dissertação, a necessidade de intervenção nos códigos fontes dos modelos legados e no resto do simulador é limitada a exportação das interfaces de comunicação e o instanciamento dos modelos em tempo de execução. Esta solução faz a integração dos modelos legados ao simulador em SMP, sem necessidade de grande intervenção nos programas dos modelos e do simulador.

A proposta do mecanismo de comunicação entre sistemas legados de simulação mostrou-se satisfatória quando aplicada em um caso de estudo incluindo dois modelos legados. Com o aumento da facilidade de integração e a disponibilidade de padrões e ferramentas, os sistemas legados de simulação tendem a transformar-se em objetos funcionais distribuídos pelas redes de computadores.

Entretanto, a transformação de um sistema completo de simulação, desenvolvido para ser executado de forma sequencial, para uma execução interoperável possuí algum grau de dificuldade para ser considerado, pois existem muitos fatores a serem levados em

conta como os modelos que não são entidades independentes ou desenvolvidos em linguagem funcional. Simuladores desenvolvidos por ferramentas que não possuem qualquer grau de interoperabilidade. E plataformas computacionais que não tem qualquer tipo de interoperabilidade em suas funcionalidades operacionais.

6.3 Trabalhos Futuros

Algumas propostas para a realização de trabalhos futuros relacionados aos assuntos abordados neste trabalho são:

- avaliar os requisitos de temporização e sincronismo em um simulador distribuído e conforme padrão SMP;
- portar a interoperabilidade CORBA - SMP proposta deste trabalho da linguagem ANSI/ISO C++ para a linguagem JAVA, a fim de torná-lo realmente independente da plataforma;
- propor uma solução para um simulador genérico que possa ser executado em qualquer simulação sem a necessidade de re-geração de código;
- propor uma solução para integrar em um único ambiente todas as ferramentas necessárias ao projeto e desenvolvimento de um simulador no SMP;
- portar toda a plataforma definida em CORBA para uma tecnologia mais nova como o RMI ou SOAP;
- converter ou estender os padrões SMP para a plataforma com suporte a WEB;
- estender o padrão SMP ou propor uma solução para simuladores que integrem modelos legados ou não, que faça uso da tecnologia de computação em nuvem.

REFERÊNCIAS BIBLIOGRÁFICAS

AMBROSIO, A. M.; CARDOSO, P. E.; BIANCHI NETO, J. Brazilian satellite simulators: previous solutions trade-off and new perspectives for the CBERS program. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS, 9, 2006, Rome, Italy. **Proceedings...** 2006. p. 7. CD-ROM. (INPE-14068-PRE/9237). Disponível em: <http://urlib.net/sid.inpe.br/mtc-m16@80/2006/08.21.15.01>. Acesso em: 05 abr. 2011.

AUSTRALIAN DEFENCE SIMULATION OFFICE (ADSO). **Distributed simulation guide.** Canberra, ACT, Australia, 2004.

BEX, G. J.; NEVEN, F.; VANSUMMEREN, S. Inferring XML schema definitions from XML data. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, 33rd, 2007, Vienna, Austria. **Proceedings…** University of Vienna: ACM, 2007. p. 998-1009. ISBN (978-1-59593-649-3).

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **The Unified modeling language user guide**. Addison Wesley, 2005. 496 p. IBSN (0-321-26797-4).

CAGNIN; M. I.; PENTEADO, R. A. D. Avaliação das vantagens quanto à facilidade de manutenção e expansão de sistemas legados sujeitos à engenharia reversa e segmentação. In: CONGRESSO NACIONAL DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, XX, 2000, Curitiba. **Anais...** Curitiba: Editora Universitária Campagnat, 2000, v. 1. p. 32.

CANHOTA, A. J. S.; SOUZA, D. A.; MOUTINHO, D. S.; LOHNEFINK, F. P. **Engenharia reversa**. UFF – Universidade Federal Fluminense. Rio de Janeiro, 2005. Disponível em: < http://www.ic.uff.br/~otton/graduacao/informaticaI/apresentacoes/eng_reversa.pdf>. Acesso em: 15 jun. 2011.

CARRARA, V. **Implementações de modelos atmosféricos para uso em propagadores de orbita e atitude**. São José dos Campos: INPE, 1990, 629p.

DEFENSE MODELING AND SIMULATION OFFICE (DMSO). **Modeling and simulation (M&S) master plan**. Alexandria, VA, USA, 1995. (DoD 5000.59-P).

DEFENSE MODELING AND SIMULATION OFFICE (DMSO). **DoD modeling and simulation (M&S) glossary**. Washington, DC, USA, 1998. (DoD 50009.59-M).

DEFENSE MODELING AND SIMULATION OFFICE (DMSO). **High Level Architecture**. Disponível em: <https://www.dmso.mil/public/transition/hla>. Acesso em: 10 jan. 2011.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **Space engineering**. System modeling and simulation. Noordwijk: ESA-ESTEC Requirements and Standard Division, 2010, (ECSS-E-TM-10-21A).

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **Space engineering**. System modelling platform. Volume 1: Principles and requirements. Noordwijk: ESA-ESTEC Requirements and Standard Division, 2011a. (ECSS-E-TM-40-07 Volume 1A).

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **Space engineering.** Simulation modeling platform – Volume 2: Metamodel. Noordwijk: ESA-ESTEC Requirements and Standard Division, 2011b. (ECSS-E-TM-40-07 Volume 2A).

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **Space engineering.** Simulation modeling platform – Volume 3: Component model. Noordwijk: ESA-ESTEC Requirements and Standard Division, 2011c. (ECSS-E-TM-40-07 Volume 3A).

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **Space engineering.** Simulation modeling platform – Volume 4: C++ Mapping. Noordwijk:

ESA-ESTEC Requirements and Standard Division, 2011d. (ECSS-E-TM-40-07 Volume 4A).

EUROPEAN SPACE OPERATION CENTER (ESOC). **SMP 2.0 Handbook.** Issue 1, Revision 2. ESA Directorate of Operations and Infrastructure, 2005a. (EGOS-SIM-GEN-TN-0099).

EUROPEAN SPACE OPERATION CENTER (ESOC). **SMP 2.0 C++ model development kit.** Issue 1, Revision 2. ESA Directorate of Operations and Infrastructure, 2005b. (EGOS-SIM-GEN-TN-1001).

EUROPEAN SPACE OPERATION CENTER (ESOC). **SMP 2.0 model and tool integration.** Issue 1, Revision 2. ESA Directorate of Operations and Infrastructure, 2005c. (EGOS-SIM-GEN-TN-1002).

ESA GROUND OPERATION SYSTEM (EGOS). **SIMSAT 4.0 Designers´s user manual**. [S.1.], 2007. (EGOS-SIM-SIM-SUM-1001).

GANDIN, S. J. **Migração de sistema legados.** 2003. 68 fl:il. Dissertação (Mestrado em Ciência da Computação). Universidade Federal do Rio Grande do Sul, Porto Alegre, 2003. Disponível em: < http://www.lume.ufrgs.br/bitstream/handle/10183/3861/000405106.pdf?sequence=1>. Acesso em: 15 out. 2011.

GENTINA, J. **Desenvolvimento e simulação da lógica embarcada para manobras de rendezvous e docking da plataforma orbital recuperável SARA**. 2010. 128 p. (sid.inpe.br/mtc-m19@80/2010/02.13.15.13-TDI). Dissertação (Mestrado em Mecânica Espacial e Controle) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2010. Disponível em: <http://urlib.net/8JMKD3MGP7W/36UCDES>. Acesso em: 10 mar. 2012.

GAVIRA, M. O. **Simulação computacional como uma ferramenta de aquisição de conhecimento.** 2003. 150 p. Dissertação (Mestrado em Engenharia de Produção) –

Universidade de São Paulo, São Carlos, 2003. Disponível em: <http://www.teses.usp.br/teses/disponiveis/18/18140/tde-20052003-004345/pt-br.php>. Acesso em: 29 nov. 2011.

GORDON, M. G. **System simulation**. Englewwod Cliffs, N.J.: Prentice Hall, 1969. 336 p. ISBN (978-81-203-0140-5).

HOFFMANN, L. T.; PERONDI, L. F. Mecanismo de um escalonador de tarefas baseado em SMP2. In: WORKSHOP EM ENGENHARIA E TECNOLOGIA ESPACIAIS, 1. (WETE), 2010, São José dos Campos. **Anais...** São José dos Campos: INPE, 2010. v. IWETE2010-1044_2. DVD. ISSN 2177-3114. Disponível em: <http://urlib.net/J8LNKAN8RW/37NR3LE>. Acesso em: 15 mai. 2011.

INSTITUTE OF ELECTRICAL AND ELETRONICS ENGINEERS (IEEE). **IEEE standard for distributed interactive simulation application protocols.** IEEE Computer Society, 1998. (IEEE 1278.1A-1998).

INSTITUTE OF ELECTRICAL AND ELETRONICS ENGINEERS (IEEE). **Standard for modeling and simulation high level architecture** - Framework and Rules. IEEE Standards Board, 2000a. (IEEE 1516-2000).

INSTITUTE OF ELECTRICAL AND ELETRONICS ENGINEERS (IEEE). **Standard for modeling and simulation high level architecture** - Federate Interface Specification. IEEE Standards Board, 2000b. (IEEE 1516.1-2000).

INSTITUTE OF ELECTRICAL AND ELETRONICS ENGINEERS (IEEE). **Standard for modeling and simulation high level architecture** - Object Model Template (OMT) Specification. IEEE Standards Board, 2000c. (IEEE 1516.2-2000).

JUDITH S. D.; RICHARD M. F.; RICHARD M. W. The Department DoD High Level Architecture. In: CONFERENCE WSC97 WINTER SIMULATION CONFERENCE, 29, 1997, Atlanta, Georgia, United States. **Proceedings…** Washington, DC, USA: IEEE Computer Society. P. 142-149. ISBN (0-7803-4278-X).

KELSO, T. S. **Satellite traking software**. [S.1.], 2000. Disponível em < http://celestrak.com/software/tskelso-sw.asp>. Acesso em: 10 jan. 2011.

KLEPPE, A.; WARMER, J.; BAST, W. **MDA explained:** the model driven architecture: practice and promise. Boston: Pearson Education, 2003. 170 p. ISBN (0-31-19442-X).

KOVSE, J.; HÄRDER, T. Generic XMI-Based UML model transformations. In: INTERNATIONAL CONFERENCE, OBJECT-ORIENTED, 8th, 2002, v. 2425, Montpellier: France. **Proceedings…** Montpellier: Springer, 2002. p. 192-198. ISBN (3-540-4480-9).

KUGA, H. K.; CARRARA, V.; RAO, K. R. **Satélites artificiais** - movimento orbital**.** São José dos Campos: INPE, 2011. 103 p. (sid.inpe.br/mtc-m19/2011/11.22.18.25-PUD). Disponível em: <http://urlib.net/8JMKD3MGP7W/3ARJ3NH>. Acesso em: 10 mar. 2012.

KUHL, F.; WEATHERLY, R.; DAHMANN, J**. Creating computer simulation systems**: an introduction to the High Level Architecture. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1999. 212 p. ISBN (0-13-022511-8).

MARANHÃO, A. F. **Um simulador para o sistema digital EAI-640.** 1977. 207 p. (INPE-1033-TPT/053). Dissertação (Mestrado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 1976. Disponível em: <http://urlib.net/sid.inpe.br/mtc-m17@80/2007/10.09.18.21>. Acesso em: 07 jun. 2011.

MIRSHAMS, M.; VAHID, M. A. D.; TAEI, H. A Systems Engineering Tool for Satellite Simulator Design. In: BIENNIAL CONFERENCE ON ENGINEERING SYSTEMS DESIGN AND ANALYSIS (ESDA2010), 10TH, 2010, v.5, Istanbul: Turkey, **Proceedings…** Istanbul: International, 2010. p. .475-483. ISBN (978-0-7918-4919-4). Disponível em <http://dx.doi.org/10.1115/ESDA2010-25341>. Acessado em: 27 dez. 2011.

NATIONAL AEROSPACE LABORATORY (NLR). **EuroSim Mk 4.0 ower´s manual.** Leiden: Ducth Space BV, 2006. (FSS-EFO-TN-530). Disponível em: <http://www.eurosim.nl/support/manuals/manual_4_0/pdf/OM.pdf>. Acesso em: 20 nov. 2011.

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION (NASA) AND EUROPEAN SPACE AGENCY (ESA). Simulation model portability 2 – overview of the SMP2 standard. In: NASA/ESA WORKSHOP ON PRODUCT DATA EXCHANGE, 6., 2004, Friedrischshafen, Germany. **Proceedings…** Friedrischshafen: VEGA, 2004.

NEVES, C. E. B.; SAMIOS, E. M. B. **Niklas Luhmann:** a nova teoria dos sistemas. Porto Alegre: Editora da universidade/UFRGS: Goethe-Institut/ICBA, 1997. 111 p. ISBN (8570254237).

OBJECT MANAGEMENT GROPU (OMG). **About OMG®.** [S.1.], 2011a. Disponível em: <http://www.omg.org/gettingstarted/gettingstartedindex.htm>. Acesso em: 10 jun. 2011.

OBJECT MANAGEMENT GROUP (OMG). **Common object request borker architecture (CORBA)**, Specification, Version 3.1.1. [S.1.], 2011b. (formal/2011-08-01).

OBJECT MANAGEMENT GROUP (OMG). **ORB Basics**. [S.1.], 2011c. Disponível em: <http://www.omg.org/gettingstarted/orb_basics.htm>. Acesso em: 15 jun. 2011.

OBJECT MANAGEMENT GROUP (OMG). MDA® **Specification.** [S.1.], 2011d. Disponível em: <http://www.omg.org/mda/specs.htm>. Acesso em: 25 jun. 2011.

PAUL, W.; BOSWEL, D.; STAKEM, P.; COWAN-SHARP, J. Open Source Software for Small Satellites. In: ANNUAL AIAA/USU CONFERENCE ON SMALL SATELLITES, 21., 2007, Logan, UT, USA. **Proceedings…** Logan: Utah State University Research Foundation, 2007. p. 1-4.

PAULSON, L. D. COBOL Skills Demand Strong. **Dice. The carrer hub for tech™**, 2006. Disponível em: <http://www.dice.com/content/seekert/COBOL_job/COBOL_job.html>. Acesso em: 05 jun. 2011.

PENN, D. L.; LEE, D.; MANI, M.; CHU, W. W. Schema Conversion Methods between XML and Relational Models. **Information and software tecnology**, v. 48, i. 1, p. 1-17, 2006.

PENTEADO, R. A. D. GERMANO, F. F. F. MASIEIRO, P. C. Engenharia reversa orientada a objetos do ambiente StatSim: método utilizado e resultados obtidos. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, SOCIEDADE BRASILEIRA DE ENGENHARIA DE SOFTWARE, 9, 1995, Recife, Brasil. **Anais...** Recife: Jaelson Castro, 1995. p. 345-362.

PINTO, H. L. M.; BRAGA, J. L. Sistemas Legados e as Novas Tecnologias: técnicas de integração e estudo de caso. **Biblioteca Digital Brasileira de Computação**, v.7, n.1, p.47-69, 2005.

PRESSMAN, R. S. **Engenharia de software**. McGraw-Hill, 2006. 720 p. ISBN (8566804576).

RAINEY, L. B. **Space modeling and simulation** – Roles and applications throughout the system life cycle. California: American Institute of Aeronautics and Astronautics/Aerospace Press, 2004. 1059 p. ISBN (1-884989-15-2).

RAY, E. T. **Aprendendo XML.** Rio de Janeiro: Editora Campus, 2001. 384 p. ISBN (8535208097).

REGGESTAD, V.; GUERRUCCI, D.; EMANUELLI, P. P.; VERRIER, D. Simulator development: the flexible approach applied to operational spacecraft simulators. In: CONFERENCE ON SPACE OPERATIONS (SPACEOPS2004), 6, 2004, Montreal, Canada. **Proceedings...** Montreal: AIAA, 2004. Papers on Disc (CD-ROM).

REIS, A. M. **Simulação distribuída em tempo-real de um sistema de controle de atitude e órbita para a plataforma multi-missão utilizando a arquitetura HLA**. 2009. 196 p. (INPE-15782-TDI/1525). Dissertação (Mestrado em Mecânica Espacial e Controle) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2009. Disponível em: <http://urlib.net/sid.inpe.br/mtc-m18@80/2009/03.02.22.17>. Acesso em: 08 abr. 2011.

SANTANDER, V. F. A.; SILVA, L. P. Uma proposta de evolução em sistemas legados. In: WORKSHOP ON REQUIREMENTS ENGINEERING. 7, 2004, v. 1. Tandil: Argentina. **Proceedings...** Buenos Aires: Grupo Editor Tercer Milenio S.A., 2004. p. 201-213.

SHANNON, R. E. **System simulation**: The art and science. Englewood Cliffs, N.J.: Prentice Hall, 1975. 368 p. ISBN (0-138-81839-8)

SIEGEL, J. **CORBA 3 Fundamentals and programing**. New York: John Wiley e Sons, 2000. 899 p. ISBN (0471295183).

SILVA, E. L.; MENEZES, E. M. **Metodologia da pesquisa e elaboração de dissertação.** Florianópolis: Laboratório de Ensino a Distancia da UFSC, 2001. 121p.

SILVA, F. M.. **SOA - Arquitetura Orientada a Serviços.** Universidade de São Paulo, São Paulo. [S.l.], 2006. Disponível em: <http://www.linux.ime.usp.br/~cef/mac499-06/monografias/filipemadeira/monografia.pdf>. Acesso em: 10 mai. 2011.

SIMULATION INTEROPERABILITY STANDARD ORGANIZATION (SISO). **Overview of SISO**: Who we are and what we do… [S.1.], 2011. Disponível em: <http://www.sisostds.org/AboutSISO/Overview.aspx>. Acesso em: 15 jul. 2011.

SOMMERVILLE, I. **Engenharia de software**. Pearson Addison–Wesley: Prentice Hall, 2007. 720 p. ISBN (978-85-88639-287).

The Important Of Being Earnest (TIOBE). **TIOBE programming community index for May 2011**. Eindhoven, 2011. Disponível em: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>. Acesso em: 05 jun. 2011.

SOUZA, M. L. O.; TRIVELATO, G. C. Simulation architectures and standards: their characteristics and applications to the simulation and control of aerospace vehicles. In: CONGRESSO E EXPOSIÇÃO INTERNACIONAIS DE TECNOLOGIA DA MOBILIDADE, 17., 2008, São Paulo, SP, Brasil. **Proceedings…** São Paulo: Sociedade de Engenheiros da Mobilidade (SAE Brasil), 2008. (SAE-2008-36-0271).

TOMINAGA, J. **Simulador de satélites para verificação de planos de operações em voo.** 2010. 174 p. (sid.inpe.br/mtc-m19@80/2010/05.24.18.55-TDI). Dissertação (Mestrado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2010. Disponível em: <http://urlib.net/8JMKD3MGP7W/37HL3J8>. Acesso em: 10 mar. 2012.

TURNER, A. J. **An Open-Source, extensible spacecraft simulation and modeling environment framework.** Dissertação (Master of Science in Aerospace Engineering) - Virginia Polytechnic Institute and State University, 2003. Disponível em: <http://scholar.lib.vt.edu/theses/available/etd-08242003-133909/unrestricted/turneretd.pdf >. Acesso em: 01 fev. 2011.

VINOSKI, S. CORBA: integrating diverse applications within distributed heterogeneous environments. **IEEE Communications Magazine**, v.14, 1997. p. 35-37. Disponível em: <http://www.cs.wustl.edu/~schmidt/PDF/vinoski.pdf>. Acesso em: 12 jun. 2011.

VRIES, R. H. EuroSim Simulation Framework. In: DASIA data system in aerospace, 2003, Prague, Czech Republic. **Proceedings…** Noordwijk, Netherlands: European Space Agency, 2003. p 4.1. (CDROM).

WEATHERLY, R.M.; WILSON, A. L.; CANOVA, B. S.; PAGE, E. H.; FISCHER, M. C. Advanced Distributed Simulation through the Aggregate Level Simulation Protocol. In: ANNUAL HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES. 29, 1996. Hawaii: USA, **Proceedings…** Hawaii: IEEE, 1996. p. 407-413. ISBN (0-8186-7327-3).

APÊNDICE A

Este apêndice apresenta conteúdos dos arquivos, escritos em SDML, que foram gerados pela ferramenta xSDML e servem com base de entrada em todos os programas que utilizam o SMP, o seu conteúdo representa as declarações das interfaces e atributos dos modelos legados usados no estudo de caso.

A.1.Catalogue do Modelo de Órbita

```
<?xml version="1.0" encoding="UTF-8"?>
<catalogue:Catalogue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:catalogue="http://www.esa.int/2005/02/Smdl/Catalogue"
xmlns:elements="http://www.esa.int/2005/02/Core/Elements"
xmlns:types="http://www.esa.int/2005/02/Core/Types"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" Id="Orbita" Name="Orbita"
Creator="teste" Date="2011-10-27T11:19:19.435Z" Version="1.0">
 <Description>Modelo de Ã³rbita em SMP2.</Description>
 <Namespace Id="Orbita" Name="Orbita">
  <Description>Area de definiÃ§Ã£o do modelo de Ã³rbita.</Description>
  <Type xsi:type="catalogue:Class" Id="Orbita.Orbita" Name="Orbita">
   <Description>Classe com as funcionalidades do modelo de Ã³rbita</Description>
   <Uuid>8ed1fc18-10be-40b6-8474-648bb805a1de</Uuid>
   <Field Id="Orbita.Orbita.fLatitudeEstacao" Name="fLatitudeEstacao"
Visibility="private">
    <Description>Latitude da estaÃ§Ã£o que monitora o satÃ©lite em
Ã³rbita</Description>
    <Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
    <Default xsi:type="types:SimpleValue">
     <Value xsi:type="xsd:float">1.0</Value>
    </Default>
   </Field>
   <Field Id="Orbita.Orbita.fLongitudeEstacao" Name="fLongitudeEstacao"
Visibility="private">
    <Description>Longitude da estaÃ§Ã£oq que monitora a Ã³rbita do
satÃ©lite</Description>
    <Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
    <Default xsi:type="types:SimpleValue">
     <Value xsi:type="xsd:float">1.0</Value>
```

```
      </Default>
    </Field>
    <Field Id="Orbita.Orbita.fAlturaEstacao" Name="fAlturaEstacao"
Visibility="private">
      <Description>Altura geogrÃ¡fica da localizaÃ§Ã£o da estaÃ§Ã£o que monitora a
Ã³rbita do satÃ©lite</Description>
      <Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
      <Default xsi:type="types:SimpleValue">
        <Value xsi:type="xsd:float">1.0</Value>
      </Default>
    </Field>
    <Property Id="Orbita.Orbita.pSatelite" Name="pSatelite">
      <Description>Nome do satÃ©lite a ter a sua Ã³rbita monitorada.</Description>
      <Type xlink:title="Interface IManagedModel"
xlink:href="http://www.esa.int/2005/02/Smp#Smp.Management.IManagedModel"/>
    </Property>
    <Operation Id="Orbita.Orbita.oTimer" Name="oTimer">
      <Description>RelÃ³gio que controla o tempo para a localizaÃ§Ã£o da Ã³rbita do
satÃ©lite</Description>
      <Type xlink:title="Interface ITimeKeeper"
xlink:href="http://www.esa.int/2005/02/Smp#Smp.Services.ITimeKeeper"/>
    </Operation>
    <Operation Id="Orbita.Orbita.eTimerLocalizacao" Name="eTimerLocalizacao">
      <Description>Evento temporizado para recuperar a Ã³rbita do satÃ©lite
selecionado</Description>
      <Type xlink:title="Interface ITimeKeeper"
xlink:href="http://www.esa.int/2005/02/Smp#Smp.Services.ITimeKeeper"/>
    </Operation>
  </Type>
  <Type xsi:type="catalogue:Interface" Id="Orbita.set_p_nome_satelite"
Name="set_p_nome_satelite">
    <Description>Nome do satÃ©lite a ter sua Ã³rbita monitorada</Description>
    <Uuid>2ae96cb7-2eca-46e5-8662-72c6e576ed82</Uuid>
    <Base xlink:title="Interface IEventManager"
xlink:href="http://www.esa.int/2005/02/Smp#Smp.Services.IEventManager"/>
  </Type>
  <Type xsi:type="catalogue:Interface" Id="Orbita.get_p_dados_satelite"
Name="get_p_dados_satelite">
    <Description>Recupera as informaÃ§Ãµes do satÃ©lite que estÃ¡ sendo
monitorado</Description>
    <Uuid>c5ce3c9d-62ad-40d0-a7f4-782cc1751f17</Uuid>
    <Base xlink:title="Interface IEventManager"
xlink:href="http://www.esa.int/2005/02/Smp#Smp.Services.IEventManager"/>
  </Type>
```

<Type xsi:type="catalogue:Interface" Id="Orbita.get_p_dados_estacao" Name="get_p_dados_estacao">
<Description>Recupera as informaÃ§Ãµes da estaÃ§Ã£o que estÃ¡ monitorando o Ã³rbita do satÃ©lite</Description>
<Uuid>75020ee1-2679-4ddd-b5ef-6d78ccf89114</Uuid>
<Base xlink:title="Interface IEventManager" xlink:href="http://www.esa.int/2005/02/Smp#Smp.Services.IEventManager"/>
</Type>
<Type xsi:type="catalogue:Interface" Id="Orbita.get_p_dados_pcd_visivel" Name="get_p_dados_pcd_visivel">
<Description>Recupera as informaÃ§Ãµes das pcds visÃ-veis pelo satÃ©lite monitorado.</Description>
<Uuid>bc31a947-7dcf-43a7-965f-b92767eb8dc4</Uuid>
<Base xlink:title="Interface IEventManager" xlink:href="http://www.esa.int/2005/02/Smp#Smp.Services.IEventManager"/>
</Type>
<Type xsi:type="catalogue:Class" Id="Orbita.SGP4_SDP4" Name="SGP4_SDP4">
<Description>Classe que determina a Ã³rbita de um determinado satÃ©lite</Description>
<Uuid>92c77b16-944d-4736-9ba1-8592f8a74737</Uuid>
<Property Id="Orbita.SGP4_SDP4.ObsLon" Name="ObsLon">
<Description>Longitude a ser observada pelo satÃ©lite em determinado tempo</Description>
<Type xlink:title="PrimitiveType Float32" xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
</Property>
<Property Id="Orbita.SGP4_SDP4.ObsLat" Name="ObsLat">
<Description>Latitude a ser observada pelo satÃ©lite em determinado tempo</Description>
<Type xlink:title="PrimitiveType Float32" xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
</Property>
<Property Id="Orbita.SGP4_SDP4.ObsHeigth" Name="ObsHeigth">
<Description>Altura a ser observada pelo satÃ©lite em determinado tempo</Description>
<Type xlink:title="PrimitiveType Float32" xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
</Property>
<Operation Id="Orbita.SGP4_SDP4.IsVisible" Name="IsVisible">
<Description>Recupea informaÃ§Ãµes da Ã³rbita e se a mesma estÃ¡ sobre a visada de uma estaÃ§Ã£o em solo e sobre as pcds</Description>
<Metadata xsi:type="elements:Comment" Id="Orbita.SGP4_SDP4.IsVisible.IsVisible" Name="IsVisible">
<Description>Retorna se uma determina localizaÃ§Ã£o informada na variÃ¡vies ObsLong, ObsLat e ObsHeight estÃ¡ visÃ-vel para o satÃ©lite. E recupera vÃ¡rias

informaÃ§Ãµe atravÃ©s dos parametros informados na chamada da
funÃ§Ã£o.</Description>
      </Metadata>
      <Type xlink:title="Enumeration OperatorKind"
xlink:href="http://www.esa.int/2005/02/Smp#Smp.Attributes.OperatorKind"/>
      <Parameter Id="Orbita.SGP4_SDP4.IsVisible.sat" Name="sat">
        <Description>Nome do satÃ©lite a ter a Ã³rbita calcula</Description>
        <Type xlink:title="PrimitiveType Char8"
xlink:href="http://www.esa.int/2005/02/Smp#Char8"/>
        <Default xsi:type="types:SimpleValue">
          <Value xsi:type="xsd:string"></Value>
        </Default>
      </Parameter>
      <Parameter Id="Orbita.SGP4_SDP4.IsVisible.jt" Name="jt">
        <Description>Data juliana para o cÃ¡culo da Ã³rbita</Description>
        <Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
        <Default xsi:type="types:SimpleValue">
          <Value xsi:type="xsd:float">1.0</Value>
        </Default>
      </Parameter>
      <Parameter Id="Orbita.SGP4_SDP4.IsVisible.ThresholdDeg"
Name="ThresholdDeg">
        <Description>Valor defindo em 0.5 graus</Description>
        <Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
        <Default xsi:type="types:SimpleValue">
          <Value xsi:type="xsd:float">1.0</Value>
        </Default>
      </Parameter>
      <Parameter Id="Orbita.SGP4_SDP4.IsVisible.southbound" Name="southbound">
        <Description>Retorna: Borda sul da satÃ©lite</Description>
        <Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
        <Default xsi:type="types:SimpleValue">
          <Value xsi:type="xsd:float">1.0</Value>
        </Default>
      </Parameter>
      <Parameter Id="Orbita.SGP4_SDP4.IsVisible.Azimute" Name="Azimute">
        <Description>Retorna: Valor do Azimute do SatÃ©lite</Description>
        <Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
        <Default xsi:type="types:SimpleValue">
          <Value xsi:type="xsd:float">1.0</Value>
        </Default>

```
      </Parameter>
      <Parameter Id="Orbita.SGP4_SDP4.IsVisible.Elevation" Name="Elevation">
        <Description>Retorna: A elevaÃ§Ã£o do satÃ©lite no tempo
informado</Description>
        <Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
        <Default xsi:type="types:SimpleValue">
          <Value xsi:type="xsd:float">1.0</Value>
        </Default>
      </Parameter>
      <Parameter Id="Orbita.SGP4_SDP4.IsVisible.Laltitude" Name="Laltitude">
        <Description>Retorna: A localizaÃ§Ã£o em laltiude do satÃ©lite em Ã³rbita
</Description>
        <Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
        <Default xsi:type="types:SimpleValue">
          <Value xsi:type="xsd:float">1.0</Value>
        </Default>
      </Parameter>
      <Parameter Id="Orbita.SGP4_SDP4.IsVisible.Longitude" Name="Longitude">
        <Description>Retorna: A localizaÃ§Ã£o em longitude do satÃ©lite em Ã³rbita
</Description>
        <Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
        <Default xsi:type="types:SimpleValue">
          <Value xsi:type="xsd:float">1.0</Value>
        </Default>
      </Parameter>
      <Parameter Id="Orbita.SGP4_SDP4.IsVisible.Altitude" Name="Altitude">
        <Description>Retorna: A altitude do satÃ©lite em Ã³rbita. </Description>
        <Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
        <Default xsi:type="types:SimpleValue">
          <Value xsi:type="xsd:float">1.0</Value>
        </Default>
      </Parameter>
    </Operation>
    <Operation Id="Orbita.SGP4_SDP4.FindAOS" Name="FindAOS">
      <Description>Recupera o tempo inicial da visada de ume localizaÃ§Ã£o de um
ponto de referÃªncia em solo</Description>
      <Type xlink:title="Interface IRegistry"
xlink:href="http://www.esa.int/2005/02/Smp#Smp.Services.IRegistry"/>
      <Parameter Id="Orbita.SGP4_SDP4.FindAOS.Sat" Name="Sat">
        <Description>Nome do SatÃ©lite </Description>
```

```
        <Type xlink:title="PrimitiveType Char8"
xlink:href="http://www.esa.int/2005/02/Smp#Char8"/>
        <Default xsi:type="types:SimpleValue">
          <Value xsi:type="xsd:string"></Value>
        </Default>
      </Parameter>
      <Parameter Id="Orbita.SGP4_SDP4.FindAOS.UTC" Name="UTC">
        <Description>Data em juliana da tempo a ser recuperado</Description>
        <Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
        <Default xsi:type="types:SimpleValue">
          <Value xsi:type="xsd:float">1.0</Value>
        </Default>
      </Parameter>
    </Operation>
    <Operation Id="Orbita.SGP4_SDP4.FindLOS" Name="FindLOS">
      <Description>Recupera a data final de visada de uma referÃªncia em
solo.</Description>
      <Type xlink:title="Interface IRegistry"
xlink:href="http://www.esa.int/2005/02/Smp#Smp.Services.IRegistry"/>
      <Parameter Id="Orbita.SGP4_SDP4.FindLOS.Sat" Name="Sat">
        <Description>Nome do satÃ©lite</Description>
        <Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
        <Default xsi:type="types:SimpleValue">
          <Value xsi:type="xsd:float">1.0</Value>
        </Default>
      </Parameter>
      <Parameter Id="Orbita.SGP4_SDP4.FindLOS.AOS" Name="AOS">
        <Description>Valor resultante da operaÃ§Ã£o FindAOS</Description>
        <Type xlink:title="Enumeration OperatorKind"
xlink:href="http://www.esa.int/2005/02/Smp#Smp.Attributes.OperatorKind"/>
        <Default xsi:type="types:SimpleValue">
          <Value xsi:type="xsd:int">0</Value>
        </Default>
      </Parameter>
    </Operation>
  </Type>
 </Namespace>
</catalogue:Catalogue>
```

A.2.Catalogue do Modelo SAG

```
<?xml version="1.0" encoding="UTF-8"?>
```

<catalogue:Catalogue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:catalogue="http://www.esa.int/2005/02/Smdl/Catalogue"
xmlns:elements="http://www.esa.int/2005/02/Core/Elements"
xmlns:types="http://www.esa.int/2005/02/Core/Types"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" Id="SAG" Name="SAG"
Creator="teste" Date="2011-10-27T12:27:19.676Z" Version="1.0">
 <Description>Modelo de geraÃ§Ã£o e controle de energia do
satÃ©lite</Description>
 <Namespace Id="SAG" Name="SAG">
  <Description>Nome principal da SAG</Description>
  <Type xsi:type="catalogue:Class" Id="SAG.SAG" Name="SAG">
   <Description>Classe prinicpal do modelo de SAG</Description>
   <Metadata xsi:type="elements:Comment" Id="SAG.SAG.ComentÃ¡rio"
Name="ComentÃ¡rio">
    <Description>Este modelo Ã© uma interface com um modelo de SAG
desenvolvido em F99.</Description>
   </Metadata>
   <Uuid>0fc716f4-ff30-4e6d-99ec-2c189fc60090</Uuid>
   <Operation Id="SAG.SAG.pCalculaSAG" Name="pCalculaSAG">
    <Description>Calcula o valor de geraÃ§Ã£o e consumo em determinado
horÃ¡rio</Description>
    <Type xlink:title="Interface IScheduler"
xlink:href="http://www.esa.int/2005/02/Smp#Smp.Services.IScheduler"/>
   </Operation>
  </Type>
  <Type xsi:type="catalogue:Interface" Id="SAG.set_p_satelite"
Name="set_p_satelite">
   <Description>Informa o satÃ©lite a ser monitorado pelo SAG</Description>
   <Uuid>6c408c1e-70cf-4df0-a456-58fb31573df6</Uuid>
   <Base xlink:title="Interface IEventManager"
xlink:href="http://www.esa.int/2005/02/Smp#Smp.Services.IEventManager"/>
  </Type>
  <Type xsi:type="catalogue:Class" Id="SAG.SAGF99" Name="SAGF99">
   <Description>Classe do Modelo SAG em Fortran 99</Description>
   <Uuid>82eb9231-c4b1-4fb3-864d-547c972c4ab3</Uuid>
   <Base xlink:title="Class SAG" xlink:href="#SAG.SAG"/>
   <Operation Id="SAG.SAGF99.pCalculaSAG" Name="pCalculaSAG">
    <Description>Calcula os valores de geraÃ§Ã£o e consumo de energia pelo
satÃ©lite</Description>
    <Type xlink:title="Interface IScheduler"
xlink:href="http://www.esa.int/2005/02/Smp#Smp.Services.IScheduler"/>
    <Parameter Id="SAG.SAGF99.pCalculaSAG.Temperatura"
Name="Temperatura">
     <Description>TEMPERATURE OF ARRAY (R)</Description>

```
      </Parameter>
      <Parameter Id="SAG.SAGF99.pCalculaSAG.Area" Name="Area">
        <Description>AREA OF ARRAY (FT2)</Description>
        <Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
        <Default xsi:type="types:SimpleValue">
          <Value xsi:type="xsd:float">1.0</Value>
        </Default>
      </Parameter>
      <Parameter Id="SAG.SAGF99.pCalculaSAG.EmissaoCelula"
Name="EmissaoCelula">
        <Description>EMITTANCE OF CELL SIDE</Description>
        <Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
        <Default xsi:type="types:SimpleValue">
          <Value xsi:type="xsd:float">1.0</Value>
        </Default>
      </Parameter>
      <Parameter Id="SAG.SAGF99.pCalculaSAG.EmissaoTraseira"
Name="EmissaoTraseira">
        <Description>EMITTANCE OF BACK SIDE</Description>
        <Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
        <Default xsi:type="types:SimpleValue">
          <Value xsi:type="xsd:float">1.0</Value>
        </Default>
      </Parameter>
      <Parameter Id="SAG.SAGF99.pCalculaSAG.AbsCelula" Name="AbsCelula">
        <Description>SOLAR ABSORPTANCE OF CELL SIDE</Description>
        <Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
        <Default xsi:type="types:SimpleValue">
          <Value xsi:type="xsd:float">1.0</Value>
        </Default>
      </Parameter>
      <Parameter Id="SAG.SAGF99.pCalculaSAG.AbsTraseira" Name="AbsTraseira">
        <Description>SOLAR ABSORPTANCE OF BACK SIDE</Description>
        <Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
        <Default xsi:type="types:SimpleValue">
          <Value xsi:type="xsd:float">1.0</Value>
        </Default>
      </Parameter>
      <Parameter Id="SAG.SAGF99.pCalculaSAG.PesoSilica" Name="PesoSilica">
        <Description>WEIGHT OF FUSED SILICA (LB)</Description>
```

<Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
<Default xsi:type="types:SimpleValue">
<Value xsi:type="xsd:float">1.0</Value>
</Default>
</Parameter>
<Parameter Id="SAG.SAGF99.pCalculaSAG.PesoPolimida"
Name="PesoPolimida">
<Description>WEIGHT OF POLYIMIDE SUBSTRATE (LB)</Description>
<Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
<Default xsi:type="types:SimpleValue">
<Value xsi:type="xsd:float">1.0</Value>
</Default>
</Parameter>
<Parameter Id="SAG.SAGF99.pCalculaSAG.PesoAdesivo"
Name="PesoAdesivo">
<Description>WEIGHT OF ADHESIVE (LB)</Description>
<Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
<Default xsi:type="types:SimpleValue">
<Value xsi:type="xsd:float">1.0</Value>
</Default>
</Parameter>
<Parameter Id="SAG.SAGF99.pCalculaSAG.Fracao" Name="Fracao">
<Description>FRACTION OF ARRAY DESIGN LIFE</Description>
<Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
<Default xsi:type="types:SimpleValue">
<Value xsi:type="xsd:float">1.0</Value>
</Default>
</Parameter>
<Parameter Id="SAG.SAGF99.pCalculaSAG.PLOAD" Name="PLOAD">
<Description>PLOAD=LOAD POWER LEVEL (KW)</Description>
<Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
<Default xsi:type="types:SimpleValue">
<Value xsi:type="xsd:float">1.0</Value>
</Default>
</Parameter>
<Parameter Id="SAG.SAGF99.pCalculaSAG.Horario" Name="Horario">
<Description>TIME (HR)</Description>
<Type xlink:title="PrimitiveType DateTime"
xlink:href="http://www.esa.int/2005/02/Smp#DateTime"/>
<Default xsi:type="types:SimpleValue">

```
          <Value xsi:type="xsd:dateTime">2005-02-23T23:00:00.0Z</Value>
        </Default>
      </Parameter>
      <Parameter Id="SAG.SAGF99.pCalculaSAG.Pontos" Name="Pontos">
        <Description>NUMBER OF POINTS</Description>
        <Type xlink:title="PrimitiveType UInt32"
xlink:href="http://www.esa.int/2005/02/Smp#UInt32"/>
        <Default xsi:type="types:SimpleValue">
          <Value xsi:type="xsd:unsignedInt">0</Value>
        </Default>
      </Parameter>
      <Parameter Id="SAG.SAGF99.pCalculaSAG.ANO" Name="ANO">
        <Description>DATE OF THE YEAR (U.T.)</Description>
        <Type xlink:title="PrimitiveType UInt16"
xlink:href="http://www.esa.int/2005/02/Smp#UInt16"/>
        <Default xsi:type="types:SimpleValue">
          <Value xsi:type="xsd:unsignedShort">0</Value>
        </Default>
      </Parameter>
      <Parameter Id="SAG.SAGF99.pCalculaSAG.SemiEixoMaior"
Name="SemiEixoMaior">
        <Description>SEMIMAJOR AXIS (FT)</Description>
        <Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
        <Default xsi:type="types:SimpleValue">
          <Value xsi:type="xsd:float">1.0</Value>
        </Default>
      </Parameter>
      <Parameter Id="SAG.SAGF99.pCalculaSAG.Exec" Name="Exec">
        <Description>ECCENTRICITY</Description>
        <Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
        <Default xsi:type="types:SimpleValue">
          <Value xsi:type="xsd:float">1.0</Value>
        </Default>
      </Parameter>
      <Parameter Id="SAG.SAGF99.pCalculaSAG.Asce" Name="Asce">
        <Description>RIGHT ASCENSION</Description>
        <Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
        <Default xsi:type="types:SimpleValue">
          <Value xsi:type="xsd:float">1.0</Value>
        </Default>
      </Parameter>
```

<Parameter Id="SAG.SAGF99.pCalculaSAG.InclinaÃ§Ã£o"
Name="InclinaÃ§Ã£o">
<Description>INCLINATION</Description>
<Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
<Default xsi:type="types:SimpleValue">
<Value xsi:type="xsd:float">1.0</Value>
</Default>
</Parameter>
<Parameter Id="SAG.SAGF99.pCalculaSAG.Perigeu" Name="Perigeu">
<Description>ARGUMENT OF PERIGEE</Description>
<Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
<Default xsi:type="types:SimpleValue">
<Value xsi:type="xsd:float">1.0</Value>
</Default>
</Parameter>
<Parameter Id="SAG.SAGF99.pCalculaSAG.Elementos" Name="Elementos">
<Description>NUMBER OF ARRAY ELEMENTS</Description>
<Type xlink:title="PrimitiveType Int32"
xlink:href="http://www.esa.int/2005/02/Smp#Int32"/>
<Default xsi:type="types:SimpleValue">
<Value xsi:type="xsd:int">0</Value>
</Default>
</Parameter>
<Parameter Id="SAG.SAGF99.pCalculaSAG.LatitudeTerra"
Name="LatitudeTerra">
<Description>NUMBER OF EARTH LATITUDE DIVISIONS</Description>
<Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
<Default xsi:type="types:SimpleValue">
<Value xsi:type="xsd:float">1.0</Value>
</Default>
</Parameter>
<Parameter Id="SAG.SAGF99.pCalculaSAG.LatitudeSol" Name="LatitudeSol">
<Description>NUMBER OF SOLAR LATITUDE DIVISIONS</Description>
<Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
<Default xsi:type="types:SimpleValue">
<Value xsi:type="xsd:float">1.0</Value>
</Default>
</Parameter>
<Parameter Id="SAG.SAGF99.pCalculaSAG.PassosOrbita"
Name="PassosOrbita">
<Description>NUMBER OF TIME STEPS PER ORBIT</Description>

```
        <Type xlink:title="PrimitiveType Float32"
xlink:href="http://www.esa.int/2005/02/Smp#Float32"/>
        <Default xsi:type="types:SimpleValue">
          <Value xsi:type="xsd:float">1.0</Value>
        </Default>
       </Parameter>
     </Operation>
   </Type>
   <Type xsi:type="catalogue:Interface" Id="SAG.get_p_dados_solar"
Name="get_p_dados_solar">
     <Description>Recupera as informaÃ§Ãµes sobre os dados do SAG</Description>
     <Uuid>ee8f774c-887c-4279-92d7-3c6e79efc185</Uuid>
     <Base xlink:title="Interface IEventManager"
xlink:href="http://www.esa.int/2005/02/Smp#Smp.Services.IEventManager"/>
   </Type>
  </Namespace>
</catalogue:Catalogue>
```

A.3. Assembly do Modelo de Órbita

SCD1

```
<?xml version="1.0" encoding="UTF-8"?>
<Assembly:Assembly xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:Assembly="http://www.esa.int/2005/02/Smdl/Assembly"
xmlns:xlink="http://www.w3.org/1999/xlink" Id="SCD1" Name="SCD1"
Creator="teste" Date="2011-10-27T13:33:35.856Z" Title="SCD1" Version="1.0">
  <Description>ImplementaÃ§Ã£o do modelo de Ã³rbita do SCD1</Description>
  <Model xlink:title="Orbita.Orbita" xlink:href="Orbita.cat#Orbita.Orbita"/>
  <Implementation>00000000-0000-0000-0000-000000000000</Implementation>
  <ModelInstance Id="mSCD1" Name="mSCD1">
    <Description>Modelo do CBERS2B</Description>
    <Model xlink:title="Class Orbita" xlink:href="Orbita.cat#Orbita.Orbita"/>
    <Implementation>d584ebcd-c444-4621-898d-4653fae1d639</Implementation>
  </ModelInstance>
  <Link xsi:type="Assembly:InterfaceLink" Id="iSatelite" Name="iSatelite">
    <Description>Recupera informaÃ§Ãµes sobre o satÃ©lite</Description>
    <Reference xlink:title="Reference get_dados_satelite"
xlink:href="Orbita.cat#Orbita.Orbita.get_dados_satelite"/>
  </Link>
  <Link xsi:type="Assembly:InterfaceLink" Id="iEstacao" Name="iEstacao">
    <Description>Obtem as informaÃ§Ãµes sobre a estaÃ§Ã£o</Description>
```

```
   <Reference xlink:title="Orbita.Orbita.get_dados_estacao"
xlink:href="Orbita.cat#Orbita.Orbita.get_dados_estacao"/>
  </Link>
  <Link xsi:type="Assembly:InterfaceLink" Id="iPCDs" Name="iPCDs">
   <Description>Recupera as informaÃ§Ãµes sobre as pcds</Description>
   <Reference xlink:title="Reference get_dados_pcd"
xlink:href="Orbita.cat#Orbita.Orbita.get_dados_pcd"/>
  </Link>
  <Link xsi:type="Assembly:InterfaceLink" Id="eSatelite" Name="eSatelite">
   <Description>Informa qual o satÃ©lite a ser monitorado pelo modelo</Description>
   <Reference xlink:title="Reference set_satelite"
xlink:href="Orbita.cat#Orbita.Orbita.set_satelite"/>
  </Link>
</Assembly:Assembly>
```

SCD2

```
<?xml version="1.0" encoding="UTF-8"?>
<Assembly:Assembly xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:Assembly="http://www.esa.int/2005/02/Smdl/Assembly"
xmlns:xlink="http://www.w3.org/1999/xlink" Id="SCD2" Name="SCD2"
Creator="teste" Date="2011-10-27T13:33:35.856Z" Title="SCD1" Version="1.0">
  <Description>ImplementaÃ§Ã£o do modelo de Ã³rbita do SCD2</Description>
  <Model xlink:title="Orbita.Orbita" xlink:href="Orbita.cat#Orbita.Orbita"/>
  <Implementation>00000000-0000-0000-0000-000000000000</Implementation>
  <ModelInstance Id="mSCD2" Name="mSCD2">
   <Description>Modelo do SCD2</Description>
   <Model xlink:title="Orbita.Orbita" xlink:href="Orbita.cat#Orbita.Orbita"/>
   <Implementation>d584ebcd-c444-4621-898d-4653fae1d639</Implementation>
  </ModelInstance>
  <Link xsi:type="Assembly:InterfaceLink" Id="iSatelite" Name="iSatelite">
   <Description>Recupera informaÃ§Ãµes sobre o satÃ©lite</Description>
   <Reference xlink:title="Orbita.Orbita.get_dados_satelite"
xlink:href="Orbita.cat#Orbita.Orbita.get_dados_satelite"/>
  </Link>
  <Link xsi:type="Assembly:InterfaceLink" Id="iEstacao" Name="iEstacao">
   <Description>Obtem as informaÃ§Ãµes sobre a estaÃ§Ã£o</Description>
   <Reference xlink:title="Orbita.Orbita.get_dados_estacao"
xlink:href="Orbita.cat#Orbita.Orbita.get_dados_estacao"/>
  </Link>
  <Link xsi:type="Assembly:InterfaceLink" Id="iPCDs" Name="iPCDs">
   <Description>Recupera as informaÃ§Ãµes sobre as pcds</Description>
   <Reference xlink:title="Orbita.Orbita.get_dados_pcd"
xlink:href="Orbita.cat#Orbita.Orbita.get_dados_pcd"/>
  </Link>
```

```
  <Link xsi:type="Assembly:InterfaceLink" Id="eSatelite" Name="eSatelite">
    <Description>Informa qual o satÃ©lite a ser monitorado pelo modelo</Description>
    <Reference xlink:title="Orbita.Orbita.set_satelite"
xlink:href="Orbita.cat#Orbita.Orbita.set_satelite"/>
  </Link>
</Assembly:Assembly>
```

CBERS2B

```
<?xml version="1.0" encoding="UTF-8"?>
<Assembly:Assembly xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:Assembly="http://www.esa.int/2005/02/Smdl/Assembly"
xmlns:xlink="http://www.w3.org/1999/xlink" Id="CBERS2B" Name="CBERS2B"
Creator="teste" Date="2011-10-27T13:33:35.856Z" Title="CBERS2B" Version="1.0">
  <Description>ImplementaÃ§Ã£o do modelo de Ã³rbita do CBERS2B</Description>
  <Model xlink:title="Class Orbita" xlink:href="Orbita.cat#Orbita.Orbita"/>
  <Implementation>00000000-0000-0000-0000-000000000000</Implementation>
  <ModelInstance Id="mCBERS2B" Name="mCBERS2B">
    <Description>Modelo do CBERS2B</Description>
    <Model xlink:title="Model Orbita" xlink:href="Orbita.cat#Orbita.Orbita"/>
    <Implementation>d584ebcd-c444-4621-898d-4653fae1d639</Implementation>
  </ModelInstance>
  <Link xsi:type="Assembly:InterfaceLink" Id="iSatelite" Name="iSatelite">
    <Description>Recupera informaÃ§Ãµes sobre o satÃ©lite</Description>
    <Reference xlink:title="Reference get_dados_satelite"
xlink:href="Orbita.cat#Orbita.Orbita.get_dados_satelite"/>
  </Link>
  <Link xsi:type="Assembly:InterfaceLink" Id="iEstacao" Name="iEstacao">
    <Description>Obtem as informaÃ§Ãµes sobre a estaÃ§Ã£o</Description>
    <Reference xlink:title="Reference get_dados_estacao"
xlink:href="Orbita.cat#Orbita.Orbita.get_dados_estacao"/>
  </Link>
  <Link xsi:type="Assembly:InterfaceLink" Id="iPCDs" Name="iPCDs">
    <Description>Recupera as informaÃ§Ãµes sobre as pcds</Description>
    <Reference xlink:title="Reference get_dados_pcd"
xlink:href="Orbita.cat#Orbita.Orbita.get_dados_pcd"/>
  </Link>
  <Link xsi:type="Assembly:InterfaceLink" Id="eSatelite" Name="eSatelite">
    <Description>Informa qual o satÃ©lite a ser monitorado pelo modelo</Description>
    <Reference xlink:title="Reference set_satelite"
xlink:href="Orbita.cat#Orbita.Orbita.set_satelite"/>
  </Link>
</Assembly:Assembly>
```

A.4. Assembly do Modelo SAG

SCD1

```
<?xml version="1.0" encoding="UTF-8"?>
<Assembly:Assembly xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:Assembly="http://www.esa.int/2005/02/Smdl/Assembly"
xmlns:types="http://www.esa.int/2005/02/Core/Types"
xmlns:xlink="http://www.w3.org/1999/xlink" Id="SAGSCD1" Name="SAGSCD1"
Creator="teste" Date="2011-10-27T16:41:18.974Z" Title="SAGSCD1"
Version="1.0">
  <Metadata xsi:type="types:Attribute" Id="SMP2InterfaceRemote"
Name="SMP2InterfaceRemote">
    <Description>Host='150.163.17.174' Protocol='Corba' Kind='Static'</Description>
  </Metadata>
  <Model xlink:title="Class SAG" xlink:href="SAG.cat#SAG.SAG"/>
  <Implementation>00000000-0000-0000-0000-000000000000</Implementation>
  <ModelInstance Id="mSAGSCD1" Name="mSAGSCD1">
    <Description>Modelo do SAG para o Satélite SCD1</Description>
    <Model xlink:title="Model SAG" xlink:href="SAG.cat#SAG.SAG"/>
    <Implementation>535beac0-14a9-4852-ad67-cd0d0fc34d4b</Implementation>
  </ModelInstance>
  <Link xsi:type="Assembly:InterfaceLink" Id="iSatelite" Name="iSatelite">
    <Description>Nome do satélite para o SAG</Description>
    <Reference xlink:title="Reference set_satelite"
xlink:href="SAG.cat#SAG.SAG.set_satelite"/>
  </Link>
  <Link xsi:type="Assembly:InterfaceLink" Id="iDadosSolar" Name="iDadosSolar">
    <Description>Recupera os dados do SAG para o satélite</Description>
    <Reference xlink:title="Reference get_dados_solar"
xlink:href="SAG.cat#SAG.SAG.get_dados_solar"/>
  </Link>
</Assembly:Assembly>
```

SCD2

```
<?xml version="1.0" encoding="UTF-8"?>
<Assembly:Assembly xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:Assembly="http://www.esa.int/2005/02/Smdl/Assembly"
xmlns:types="http://www.esa.int/2005/02/Core/Types"
xmlns:xlink="http://www.w3.org/1999/xlink" Id="SAGSCD2" Name="SAGSCD2"
Creator="teste" Date="2011-10-27T16:41:18.974Z" Title="SAGSCD1"
Version="1.0">
```

```
  <Metadata xsi:type="types:Attribute" Id="SMP2InterfaceRemote"
Name="SMP2InterfaceRemote">
    <Description>Host='150.163.17.174' Protocol='Corba' Kind='Static'</Description>
  </Metadata>
  <Model xlink:title="Class SAG" xlink:href="SAG.cat#SAG.SAG"/>
  <Implementation>00000000-0000-0000-0000-000000000000</Implementation>
  <ModelInstance Id="mSAGSCD2" Name="mSAGSCD2">
    <Description>Modelo do SAG para o Satélite SCD1</Description>
    <Model xlink:title="SAG.SAG" xlink:href="SAG.cat#SAG.SAG"/>
    <Implementation>535beac0-14a9-4852-ad67-cd0d0fc34d4b</Implementation>
  </ModelInstance>
  <Link xsi:type="Assembly:InterfaceLink" Id="iSatelite" Name="iSatelite">
    <Description>Nome do satélite para o SAG</Description>
    <Reference xlink:title="SAG.SAG.set_satelite"
xlink:href="SAG.cat#SAG.SAG.set_satelite"/>
  </Link>
  <Link xsi:type="Assembly:InterfaceLink" Id="iDadosSolar" Name="iDadosSolar">
    <Description>Recupera os dados do SAG para o satélite</Description>
    <Reference xlink:title="SAG.SAG.get_dados_solar"
xlink:href="SAG.cat#SAG.SAG.get_dados_solar"/>
  </Link>
</Assembly:Assembly>
```

CBERS2B

```
<?xml version="1.0" encoding="UTF-8"?>
<Assembly:Assembly xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:Assembly="http://www.esa.int/2005/02/Smdl/Assembly"
xmlns:types="http://www.esa.int/2005/02/Core/Types"
xmlns:xlink="http://www.w3.org/1999/xlink" Id="SAGCBERS2B"
Name="SAGCBERS2B" Creator="teste" Date="2011-10-27T16:41:18.974Z" Title="
SAGCBERS2B " Version="1.0">
  <Metadata xsi:type="types:Attribute" Id="SMP2InterfaceRemote"
Name="SMP2InterfaceRemote">
    <Description>Host='150.163.17.174' Protocol='Corba' Kind='Static'</Description>
  </Metadata>
  <Model xlink:title="SAG.SAG" xlink:href="SAG.cat#SAG.SAG"/>
  <Implementation>00000000-0000-0000-0000-000000000000</Implementation>
  <ModelInstance Id="mSAGCBERS2B" Name="mSAGCBERS2B">
    <Description>Modelo do SAG para o Satélite SCD1</Description>
    <Model xlink:title="SAG.SAG" xlink:href="SAG.cat#SAG.SAG"/>
    <Implementation>535beac0-14a9-4852-ad67-cd0d0fc34d4b</Implementation>
  </ModelInstance>
  <Link xsi:type="Assembly:InterfaceLink" Id="iSatelite" Name="iSatelite">
```

```
    <Description>Nome do satélite para o SAG</Description>
    <Reference xlink:title="SAG.SAG.set_satelite"
xlink:href="SAG.cat#SAG.SAG.set_satelite"/>
  </Link>
  <Link xsi:type="Assembly:InterfaceLink" Id="iDadosSolar" Name="iDadosSolar">
    <Description>Recupera os dados do SAG para o satélite</Description>
    <Reference xlink:title="SAG.SAG.get_dados_solar"
xlink:href="SAG.cat#SAG.SAG.get_dados_solar"/>
  </Link>
</Assembly:Assembly>
```