MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21b/2017/12.07.15.25-TDI

ESTUDO E COMPARAÇÃO DE ARQUITETURAS E FERRAMENTAS PARA SIMULAÇÃO DISTRIBUÍDA COM APLICAÇÃO AO SISTEMA DE CONTROLE DE ATITUDE E ÓRBITA (SCAO) DA PLATAFORMA MULTIMISSÃO (PMM)

Suely Maria Campos Romeiro

Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Engenharia e Gerenciamento de Sistemas Espaciais, orientada pelo Dr. Marcelo Lopes de Oliveira e Souza, aprovada em 12 de dezembro de 2017.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34P/3Q6CAAH>

INPE
São José dos Campos
2017

PUBLICADO POR:

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

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21b/2017/12.07.15.25-TDI

ESTUDO E COMPARAÇÃO DE ARQUITETURAS E FERRAMENTAS PARA SIMULAÇÃO DISTRIBUÍDA COM APLICAÇÃO AO SISTEMA DE CONTROLE DE ATITUDE E ÓRBITA (SCAO) DA PLATAFORMA MULTIMISSÃO (PMM)

Suely Maria Campos Romeiro

Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Engenharia e Gerenciamento de Sistemas Espaciais, orientada pelo Dr. Marcelo Lopes de Oliveira e Souza, aprovada em 12 de dezembro de 2017.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34P/3Q6CAAH>

INPE
São José dos Campos
2017

Dados Internacionais de Catalogação na Publicação (CIP)

Romeiro, Suely Maria Campos.

R664e Estudo e comparação de arquiteturas e ferramentas para simulação distribuída com aplicação ao sistema de controle de atitude e órbita (SCAO) da plataforma multimissão (PMM) / Suely Maria Campos Romeiro. – São José dos Campos : INPE, 2017.

xxx + 199 p. ; (sid.inpe.br/mtc-m21b/2017/12.07.15.25-TDI)

Dissertação (Mestrado em Engenharia e Tecnologia Espaciais/Engenharia e Gerenciamento de Sistemas Espaciais) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2017.

Orientador : Dr. Marcelo Lopes de Oliveira e Souza.

1. Simulação. 2. DIS. 3. HLA. 4. RTI. I.Título.

CDU 629.7.062.2

Aluno (a): ***Suely Maria Campos Romeiro***

Título: "ESTUDO E COMPARAÇÃO DE ARQUITETURAS E FERRAMENTAS PARA SIMULAÇÃO DISTRIBUÍDA COM APLICAÇÃO AO SISTEMA DE CONTROLE DE ATITUDE E ÓRBITA (SCAO) DA PLATAFORMA MULTIMISSÃO (PMM)".

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

Engenharia e Tecnologia Espaciais/Eng. Gerenc. de Sistemas Espaciais

Dr. Alírio Cavalcanti de Brito

Presidente / INPE / São José dos Campos - SP

Dr. Marcelo Lopes de Oliveira e Souza

Orientador(a) / INPE / SJCampos - SP

Dr. Maurício Gonçalves Vieira Ferreira

Membro da Banca / INPE / SJCampos - SP

Dr. Eloy Martins de Oliveira Junior

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

Este trabalho foi aprovado por:

( ) maioria simples

(X) unanimidade

São José dos Campos, 12 de dezembro de 2017

"O cosmos está dentro de nós, somos feitos do mesmo material das estrelas. Somos uma maneira do cosmos conhecer a si mesmo."

Carl Sagan

A Geraldo e Julia

AGRADECIMENTOS

Ao INPE, pela oportunidade de continuar meus estudos, fornecendo as ferramentas adequadas de ensino e aprendizado.

Agradeço ao professor Marcelo pelos ensinamentos, pela amizade, paciência e atenção ao longo da orientação deste trabalho e, principalmente, pela confiança em mim depositada, para desenvolver um tema de grande importância.

Aos amigos da Embraer, André Moraes, Rodrigo Britto e Marco Ortiz, que contribuíram com suas ideias e sugestões.

À EMBRAER e ao meu supervisor, Cristian Ando, que permitiu meu estudo e investiu horas para que eu pudesse concluir este trabalho, favorecendo meu crescimento. Ao meu colega de trabalho, Flávio Borges, que me ajudou com as tarefas da empresa para que eu pudesse utilizar minhas horas de estudo.

Agradeço também aos amigos da VT- MÄK que forneceram, além dos *softwares*, um curso sobre as arquiteturas DIS e HLA/RTI, nas próprias dependências da empresa VT-MÄK, em Cambridge, MA, nos Estados Unidos. Também, durante o desenvolvimento deste trabalho, me deram suporte sobre minhas dúvidas, principalmente na utilização das ferramentas, me atendendo através de e-mails e vídeo conferências. São eles: Steve Peart, Melinda Minear, Felix Rodrigues e Iván Lopez.

Agradeço imensamente à minha mãe, Suely, por ter me dado palavras de apoio.

Agradeço ao meu marido e minha filha Julia, por terem entendido minhas ausências, mesmo quando eu estava presente.

RESUMO

Atualmente, as arquiteturas de simulação distribuídas tem apresentado importância nos cenários de desenvolvimento e treinamento, principalmente pelas suas vantagens de poder conectar simuladores diferentes e geograficamente distantes, além de permitirem o reuso de recursos, tais como códigos e máquinas. Devido a estas características, essas arquiteturas têm sido utilizadas no desenvolvimento de sistemas complexos, encontrando ampla aplicação no setor aeroespacial. Dessa forma, o estudo das mesmas pode apresentar aplicação para as pesquisas realizadas pelo INPE, conforme apresentado. Este trabalho apresenta o estudo e comparação de ambientes para simulação utilizando padrões de simulação distribuída DIS e HLA/RTI. Para isso, ele inicia com a apresentação de conceitos e revisão da literatura, seguidas pela apresentação de modelos e ferramentas. Exemplos de simulações utilizando modelos aeroespaciais são desenvolvidos. Em especial, é apresentado um caso de simulação utilizando o modelo do Sistema de Controle de Atitude e de Órbita (SCAO), operando no Modo Nominal. Esse modelo apresenta os parâmetros de órbita e características físicas do Satélite Amazônia-1, satélite este que utiliza o módulo de serviço PMM. Ao final, é apresentada uma discussão sobre as ferramentas estudadas, uma comparação entre elas e conclusões a respeito da utilização destas.

Palavras-chave: Simulação. DIS. HLA. RTI.

STUDY AND COMPARISON OF DISTRIBUTED SIMULATION ARCHITECTURES AND TOOLS WITH AN APPLICATION TO THE ATTITUDE AND ORBIT CONTROL SYSTEM (AOCS) OF THE MULTIMISSION PLATFORM (MMP)

ABSTRACT

Currently, the architectures of distributed simulation have gained importance for project developing and training, especially due to their characteristics of connecting different simulators geographically distributed, moreover because they enable reuse codes and machines. Due to this, the study of those architectures may present application to the resources developed by INPE, as presented here. In this work, we present a study and comparison of distributed simulation architectures and tools with an application to the Attitude and Orbit Control System (AOCS) of the MultiMission Platform (MMP). To do that, we first review the related concepts and literature. Then, we present models and tools herein used. We develop examples of simulations using aerospace models, specially, a case of simulation using the Attitude and Orbit Control System (AOCS) model, operating in its Nominal Mode. This model uses the orbital parameters and physical features of Amazonia-1 satellite, which employs the PMM service module. In the end, we do a discussion and comparison concerning the tools used here.

Keywords: Simulation. Satellite. DIS HLA. RTI.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

LISTA DE SIGLAS E ABREVIATURAS

ADS	*Advanced Distributed Simulation*
ALSP	*Aggregate Level Simulation Protocol*
API	*Application Programming Interface*
BSD	*Berkeley Software Distribution*
CC	Corrente Contínua
CDDL	*Common Developer and Distribution License*
COTS	*Commercial-Off-The-Shelf*
CPU	*Central Processing Unit*
CTC	*Command and Telemetry Computer*
DLL	*Dynamically Linked Library*
DIS	*Distributed Interactive Simulation*
DMSO	*Defense Modeling and Simulation Office*
DoD	*Department of Defense*
EUA	Estados Unidos da América
FOM	*Federation Object Model*
GPL	*GNU General Public License*
GNU	*GNU is not Unix*
HLA	*High Level Architecture*
IEEE	*Institute of Electrical and Electronics Engineers*
INPE	Instituto Nacional de Pesquisas Espaciais
IP	*Internet Protocol*
ISS	*International Space Station*
LEO	*Low Earth Orbit*
LGPL	*GNU Lesser General Public License*
M&S	Modelagem e Simulação
NASA	*National Aeronautics and Space Administration*
OBDH	*Onboard Data Handling*
OMT	*Object Model Template*
OSS	*Open Source Software*
PC	*Personal Computer*

PMM	Plataforma Multi-Missão
RI	Referencial Inercial
RO	Referencial Orbital
RS	Referencial do Satélite
RTI	*Run-Time Infrastructure*
SAE	*Society of Automotive Engineers*
SACI	Satélite de Aplicações Científicas
SCAO	Sistema de Controle de Atitude e de Órbita
SISO	*Simulation Interoperability Standards Organization*
SOM	*Simulation Object Model*
TENA	*Test and Training Enabling Architecture*
UML	*Unified Modeling Language*

LISTA DE SÍMBOLOS

k	=	Constante elástica da mola (MMA)
m	=	Massa da mola (MMA)
b	=	Fator de amortecimento (MMA)
h	=	Passo da simulação (s)
Ti	=	Ganho Integral PID
Kp	=	Ganho Proporcional PID
Td	=	Ganho Derivativo PID
td	=	Tempo de atraso (*Delay time*)
tr	=	Tempo de subida (*Rise time*)
Mp	=	Sobre-elevação (*Overshoot*)
ts	=	Tempo de acomodação (*Settling time*)
Z	=	Fator de amortecimento
ωn	=	Frequência natural não amortecida
ω_d	=	Frequência natural amortecida
c	=	Pólo dominante do regime transitório
Ti x	=	Ganho Integral PID em x
Kp x	=	Ganho Proporcional PID em x
Td x	=	Ganho Derivativo PID em x
Ti y	=	Ganho Integral PID em y
Kp y	=	Ganho Proporcional PID em y
Td y	=	Ganho Derivativo PID em y
Ti z	=	Ganho Integral PID em z

Kp z	=	Ganho Proporcional PID em z
Td z	=	Ganho Derivativo PID em z
Isx	=	Momento de Inércia do Satélite em x (kg.m2)
Isy	=	Momento de Inércia do Satélite em y (kg.m2)
Isz	=	Momento de Inércia do Satélite em z (kg.m2)
ϕ	=	Ângulo de rolamento (radianos)
θ	=	Ângulo de arfagem (radianos)
Ψ	=	Ângulo de guinada (radianos)
Xϕ ref	=	Posicionamento desejado em x (radianos)
Yθ ref	=	Posicionamento desejado em y (radianos)
ZΨ ref	=	Posicionamento desejado em z (radianos)
Xϕ	=	Posicionamento em x (radianos)
Yθ	=	Posicionamento em y (radianos)
ZΨ	=	Posicionamento inicial em z (radianos)
ω	=	Velocidade angular do satélite (rpm)
ωx	=	Velocidade angular x do satélite (rpm)
ωy	=	Velocidade angular y do satélite (rpm)
ωz	=	Velocidade angular z do satélite (rpm)
$\dot{\omega}$	=	Aceleração angular do satélite (rpm/m)
$\dot{\omega}_x$	=	Aceleração angular x do satélite (rpm/m)
$\dot{\omega}_y$	=	Aceleração angular y do satélite (rpm/m)
$\dot{\omega}_z$	=	Aceleração angular z do satélite (rpm/m)
Mx	=	Momento de Inércia do Satélite em x (kg.m^2)

My = Momento de Inércia do Satélite em y ($kg.m^2$)

Mz = Momento de Inércia do Satélite em z ($kg.m^2$)

ωr = Velocidade angular da roda (rpm)

ωrx = Velocidade angular x da roda (rpm)

ωry = Velocidade angular y da roda (rpm)

ωrz = Velocidade angular z da roda (rpm)

Mr = Momento de Inércia da roda ($kg.m^2$)

Mrx = Momento de Inércia da roda em x ($kg.m^2$)

Mry = Momento de Inércia da roda em y ($kg.m^2$)

Mrz = Momento de Inércia da roda em z ($kg.m^2$)

ωo = Velocidade orbital média (rad/s)

Iw = Momento de Inércia das rodas ($kg.m^2$)

ωrmax = Velocidade angular máxima das rodas (rpm)

Mrmax = Torque máximo das rodas (N.m)

Kw = Ganho das rodas

Tw = Constante de tempo das rodas (s)

SUMÁRIO

Pág.

1 INTRODUÇÃO

1.1. Contexto e Motivação para a Utilização de Modelagem e Simulação no Desenvolvimento de Sistemas de Engenharia Complexos ou Altamente Integrado e Aplicações ao INPE

Desde a antiguidade, simulações e treinamentos já eram utilizados, principalmente por tropas de exércitos, para se prepararem antes do campo de batalha. Essas atividades acrescentavam grande vantagem a essas forças que a utilizavam. Através dos treinamentos, traçavam-se estratégias, experimentavam-se novas armas, simulavam-se tropas inimigas e imaginavam-se as mais diversas situações de ataques, preparando-se para as situações reais.

Ao passar dos anos e da tecnologia, os métodos de simulação evoluíram e surgiram, gradativamente, simuladores cada vez mais complexos e elaborados.

Os avanços na microeletrônica, redes de computadores e projeção gráfica, fizeram surgir supercomputadores, a internet e tornaram possível a projeção de ambientes sintéticos para realidade virtual.

Essa eletrônica moderna permitiu então que os simuladores apresentassem níveis de detalhamento cada vez mais elevados, indo dos mais simples, como cabines de aeronaves não energizadas (ilustrativas) a sistemas altamente integrados e complexos, que envolvem diferentes máquinas, computadores, softwares, entidades virtuais (modelos humanos) e seres humanos operando simultaneamente.

Esses simuladores complexos apresentam vasta aplicabilidade para treinamento de tropas militares, colocando-as em cenários de guerra quase reais. M&S são utilizados então para treinamentos (para que mantenham a prontidão), análise (dos efeitos das táticas propostas e dos sistemas empregados), ensaios e planejamento de operações e para a demonstração de novas tecnologias tais como novos armamentos.

A Figura 1.1 ilustra um dos primeiros e mais conhecidos simuladores de voo, chamado de *Link-Trainer*, de 1929, muito utilizado durante a Segunda Guerra Mundial para treinamento de pilotos. A foto é de 1943, onde o simulador está sendo utilizado numa estação de treinamento de voo das forças armadas britânicas.

Figura 1.1 – Simulador de Voo *Link-Trainer*.

Fonte: Wikipedia (2018).

Do mesmo modo, as práticas de Modelagem e Simulação (M&S) também foram absorvidas por outras áreas, tais como medicina e indústria, tendo em vista seus benefícios de antecipar erros e reduzir riscos.

Na indústria, principalmente automobilística, aeronáutica e aeroespacial, a M&S tem sido amplamente usada durante diversas fases de desenvolvimento de projetos, desde a concepção, gerenciamento de requisitos, até testes e validação. É também usada como ferramenta para antecipar erros de integração e desenvolvimento, direcionar adequação para cumprimento de requisitos e até

mesmo auxiliar no desenvolvimento de técnicas e ferramentas para correção de problemas ou para manutenção.

É nesse contexto de M&S de modelos complexos e altamente integrados que a utilização de simulações distribuída é essencial. Essas arquiteturas foram padronizadas, permitindo assim, mais facilmente a integração e reuso de modelos, ferramentas e conhecimentos.

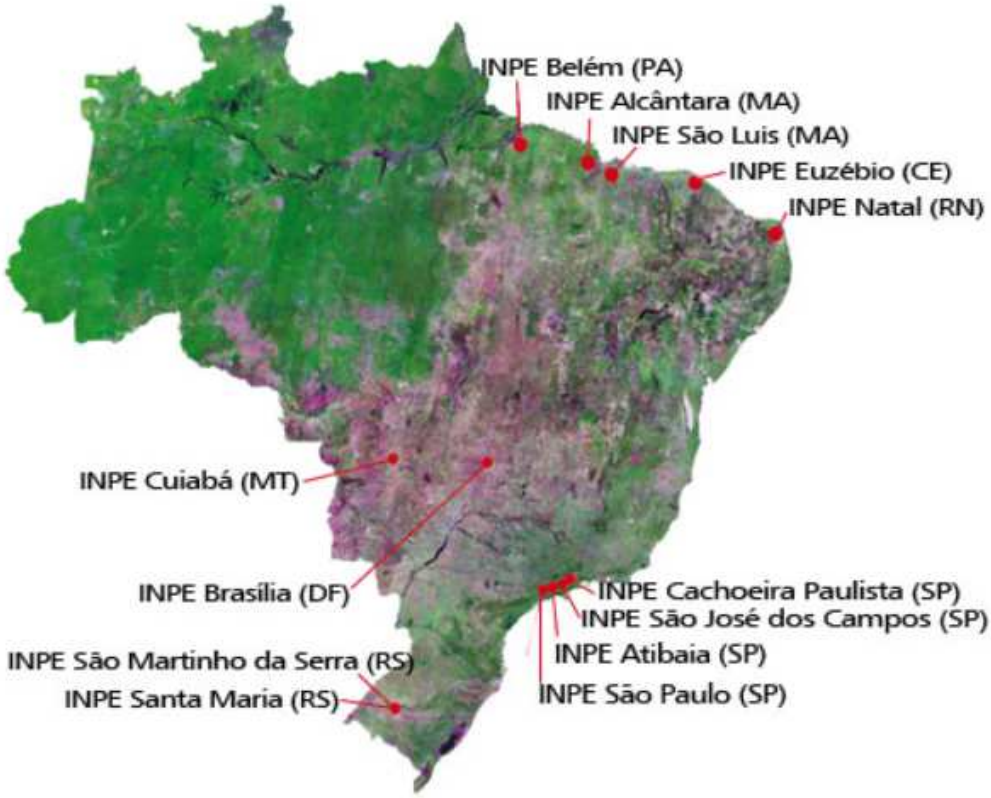
As arquiteturas de simulação distribuída atualmente mais utilizadas são DIS (*Distributed Interactive Simulation)*, HLA (*High Level Architecture*) e TENA/CTIA *(Test and Training Enabling Architecture/ Common Training Instrumentation Architecture)* (Salio et al, 2012). O principal escopo dessas arquiteturas é, com a padronização, tornar possível a interoperação de diferentes modelos feitos em diferentes ferramentas e por diferentes pessoas.

Ao longo deste trabalho iremos apresentar os conceitos, bem como um breve histórico da utilização das simulações distribuídas, estudos de casos e resultados.

1.2. Como a Simulação Distribuída Beneficiaria o INPE

O INPE apresenta diversos laboratórios espalhados pelo Brasil em seus 12 sítios (Figura 1.2). Estes se beneficiariam da simulação distribuída.

Figura 1.2 – Laboratórios do INPE no Brasil.



Fonte: INPE (2018).

O sítio localizado em São José dos Campos é o escritório principal, sendo responsável por Pesquisas Científicas e Desenvolvimento de Tecnologias em Ciências do Espaço e Atmosféricas.

Em termos de Laboratórios, os principais são os seguintes:

- LIT – Laboratório de Integração e Testes;
- LAC/ CTE – Laboratório Associado de Computação e Matemática Aplicada;
- LAP – Laboratório de Plasma;

- LCP – Laboratório de Combustão e Propulsão.

Muitos desses laboratórios exercitam diferentes modelos para sistemas espaciais (satélites) ou sistemas terrestres (estações), em diferentes ferramentas de simulação, sendo que esses modelos poderiam ser interconectados para a troca de informações, incluindo cenários com *hardware* real integrado. Indo mais além, modelos espaciais de outras entidades que apresentam contratos de desenvolvimento de tecnologias com o INPE, tais como o IAE (*Instituto de Aeronáutica e Espaço*), poderiam ser interconectados aos modelos do INPE.

Sendo assim, a utilização de arquiteturas distribuídas e ferramentas padronizadas, poderiam beneficiar as pesquisas científicas espaciais do INPE, e até mesmo projetos espaciais já em desenvolvimento, trazendo benefícios para a comunidade científica.

1.3. Contexto

O padrão HLA (*High Level Architecture*) resultou da necessidade do DoD (*US Department of Defense*) de uma arquitetura padronizada que pudesse conectar um grande número de simuladores operando em diferentes plataformas, em apenas uma plataforma (Jense & Kuijpers, 1997).

Essa arquitetura surgiu entre 1995 e 2000 como uma evolução do padrão anterior, DIS (*Distributed Interactive Simulation*) e do ALSP (*Aggregated Level Simulation Protocol*), ambos os padrões usados em arquiteturas de simulação distribuída. Porém, o HLA apresenta a grande vantagem de suportar diferentes tipos de protocolos de comunicação, permitindo também que um grande número de simuladores opere simultaneamente (Jense & Kuijpers, 1997).

Em Reis (2009) há uma apresentação sobre as diferenças entre Simulação Paralela e Simulação Distribuída.

A Simulação Paralela é possível com máquinas apresentando o mesmo tipo de configuração, onde as máquinas estão relativamente próximas, e apresentam

protocolos de comunicação feitos para atender essa rede. As memórias e processamentos algumas vezes são compartilhados (Reis, 2009).

Na Simulação Distribuída há um número diversos de máquinas diferentes, operando sem compartilhamento de memória e processamento, e espalhadas geograficamente (Reis, 2009).

Tabela 1.1 – Comparação entre Simulações Paralela e Distribuída.

	Simulação Paralela	**Simulação Distribuída**
Conjunto de Computadores	Homogênea	Heterogênea
Localização	Proximidade Física	Geograficamente Distribuídos
Mecanismo de Comunicação	Específicos para Computação Paralela	Padronizado (Padrões disponíveis em mercado)

Fonte: Fujimoto (2000), Reis (2009).

Dessa forma, entre 1995 e 2000, tentou-se estabelecer o HLA como arquitetura de simulação distribuída e o mesmo deveria ser desenvolvido para atender os requisitos extensivos do DoD para simulações, suportando o reuso de software e hardware, reduzindo tempo e custos de projetos, bem como sendo mais eficiente (Reid, 2000).

O HLA foi patrocinado pelo DoD desde 1995, sendo padronizado e aprovado pelo IEEE em 2000.

O primeiro HLA foi chamado de padrão HLA 1.3; depois surgiu o HLA 1516-2000; e, finalmente, o 1516-2010 (HLA *Evolved*).

Como um padrão de arquitetura patrocinado pelo DoD, o HLA foi desenvolvido inicialmente para aplicações militares; contudo, tem sido recentemente utilizado em um número variado de aplicações, o que inclui simulações de sistemas aeroespaciais e aeronáuticos, assim como em aplicações de tempo real, do tipo semelhante a jogos (*game-like*), muitas vezes voltados para treinamentos de pessoas.

Atualmente, existem muitas empresas de capital privado que desenvolvem RTIs para ambientes e aplicações em HLA. Porém todos esses *softwares* (RTIs) devem ser submetidos ao DMSO (*Modeling and Simulation Coordination Office*) para análise e aprovação de que cumprem com os requisitos do padrão HLA.

As simulações apresentadas neste trabalho utilizam o protocolo DIS e/ou HLA/RTI.

A RTI da VT-MÄK foi verificada pelo DMSO (*Modeling and Simulation Coordination Office*) e aprovada como apropriada para utilização como interface dos HLA 1.3, 1516 e 1516 *Evolved*, após aquele ter confirmado que todos os serviços foram implementados (VT-MÄK, 2016).

Também apresentamos neste trabalho exemplos com utilização das RTIs da PITCH (verificado pelo DMSO) e da PORTICO. A RTI da PORTICO é um *software open-source*, podendo ser baixado gratuitamente e, usuários cadastrados podem contribuir com o código.

1.4. Motivação

A principal motivação é a importância que a Modelagem e Simulação têm ganhado no desenvolvimento de projetos, especialmente na indústria aeroespacial, mas principalmente os benefícios que as aplicações das arquiteturas padronizadas de simulação distribuída trariam para a comunidade de pesquisas do INPE.

Assim, os conceitos e modelos aqui apresentados, podem ser reutilizados e aplicados em outros estudos e desenvolvimentos em andamento no INPE.

1.5. Objetivo deste Trabalho

O objetivo deste trabalho é estudar algumas das arquiteturas atualmente disponíveis para simulação distribuída, dando enfoque para o HLA, com aplicação à simulação de um modelo de satélite PMM, utilizando as equações de controle de atitude e órbita.

Este estudo será conduzido primeiramente através da apresentação de conceitos, seguido pela apresentação de ferramentas que utilizam esses padrões, com demonstrações de casos de simulação.

Ao final, será apresentada uma discussão a respeita da utilização dessas ferramentas e uma comparação com uma ferramenta já amplamente utilizada e estabelecida no meio acadêmico e na indústria (MATLAB).

1.6. Formulação do Problema e Abordagens para a Solução

O problema proposto é demonstrar a aplicabilidade de arquiteturas e ferramentas para a simulação distribuída a sistemas espaciais, considerando a importância de modelagem e simulação (M&S) no desenvolvimento de sistemas complexos e altamente integrados.

Para tanto, usaremos três abordagens: teoria e análise (apresentadas nos capítulos 2 e 6), modelagem e simulação (capítulos 3, 5 e 7).

Como caso de estudo foi escolhido um sistema de controle de atitude e órbita (SCAO) compatível com os satélites em desenvolvimento no INPE, mais especificamente o Amazônia-1, e sua implementação em um modelo compatível com o HLA/RTI.

O modelo desenvolvido em C++ e utilizado numa simulação HLA/RTI, também foi implementado em MATLAB, para comparações.

Outros casos de estudo também são apresentados, com o propósito de análise e experimentação de outras ferramentas que apresentam a interface HLA/RTI.

1.7. Organização deste Trabalho

No capítulo 1 são apresentados o contexto e a motivação, os objetivos e a organização deste trabalho.

No capítulo 2 são apresentados os conceitos básicos e a revisão da literatura.

No capítulo 3 são apresentados conceitos de sistemas de coordenadas, órbitas de satélites e as equações rotacionais e dinâmicas de um satélite em órbita, necessários para o modelamento do sistema a ser simulado.

No capítulo 4 são apresentadas a PMM e o Satélite Amazônia 1, descrevendo a composição e missão destes.

O capítulo 5 apresenta a definição dos parâmetros, os quais serão aplicados às equações de movimento do satélite.

No capítulo 6 são apresentadas as ferramentas utilizadas e as respectivas características destas.

No capítulo 7 são apresentadas as simulações utilizadas para verificação das ferramentas.

No capítulo 8 são apresentadas, a conclusão, as dificuldades encontradas da utilização das ferramentas, bem como uma comparação entre elas.

2 CONCEITOS BÁSICOS E REVISÃO DA LITERATURA

Este capítulo apresenta os conceitos básicos sobre as arquiteturas de simulação distribuída e também uma revisão da literatura, discorrendo sobre publicações a cerca deste tema, incluindo publicações com foco no cenário aeroespacial e aeronáutico.

2.1. Padrões de Arquiteturas de Simulação Distribuída, Conceitos, Histórico e Cenário Atual

A seguir, serão apresentados alguns dos conceitos que surgiram junto com os padrões de arquitetura de simulação distribuída.

2.1.1. DIS (*Distributed Interactive Simulation*)

O DIS foi um dos primeiros padrões de simulação distribuída utilizados.

Esse padrão também surgiu da necessidade do DoD interconectar um grande número de simuladores heterogêneos, através de rede local e de grandes áreas de extensão (Jense & Kuijpers, 1997).

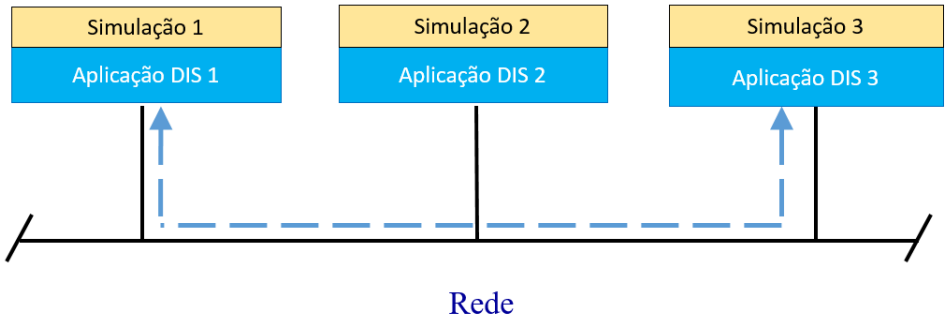
Este protocolo é o predecessor do HLA como padrão e foi originalmente moldado para aplicações militares. As dificuldades encontradas durante a utilização desse padrão, bem como o conceito inicial de interoperabilidade em simulações distribuídas, serviram de sedimentação para a concepção das regras do HLA. Contudo, esse padrão também continuou evoluindo mesmo depois do HLA, isso devido principalmente a muitos esforços em estudos e aplicações do protocolo DIS em tecnologias aeroespaciais (Jense & Kuijpers, 1997).

O DIS é simplesmente um protocolo de comunicação. Não há requisitos definidos para a arquitetura de *software* dos simuladores que cumprem com o DIS. Pacotes de *software* podem ser adquiridos para prover o programador com o básico de comunicação DIS, porém, um conhecimento profundo do DIS se faz necessário para a utilização desses pacotes (Jense & Kuijpers, 1997). Nesse aspecto, o DIS

se mostra menos flexível, requerendo maiores adaptações nos simuladores, para que os mesmos possam interoperar.

Dessa forma, resumidamente, o DIS é um protocolo de comunicação para simulações distribuídas e que operem em rede. Nesse padrão, as simulações interagem através de aplicações DIS, que consistem de pacotes de comunicação formatados, chamados de PDUs (*Protocol Data Units*). A formatação das PDUs define a semântica (significado dos dados) e a sintaxe (formato e regras) da comunicação (MacCall & Murray 2010).

Figura 2.1 – Interoperação utilizando o DIS.



Fonte: Adaptado de VT-MÄK (2016).

O DIS foi desenvolvido durante o programa SIMNET do governo americano, iniciado na década de 90, que manteve várias atualizações; e, apesar dos novos padrões que surgiram após, o DIS ainda é muito utilizado, por ser considerado de fácil uso e pela maturidade acumulada ao longo dos anos (Clark et al, 2000). Contudo, há aspectos da comunicação DIS que merecem atenção, tais como a largura de banda da rede e o impacto computacional no simulador, devido à técnica de *broadcast* usada pelos simuladores inseridos na simulação, podendo ser considerada como uma desvantagem do DIS, quando comparado a outros padrões.

Como o DIS é um padrão IEEE, qualquer simulador conectado à rede e que implemente a mesma versão de protocolo DIS, pode participar da comunicação (Clark et al, 2000). Essa característica já foi considerada por alguns autores como um ponto de vulnerabilidade do padrão, tendo em vista as informações sensíveis que trafegariam em uma rede de simulação de sistemas militares, por exemplo.

Comercialmente, existem vários *software* e ferramentas que viabilizam o desenvolvimento de uma aplicação para operação DIS, como, por exemplo, a ferramenta VR-Link da VT-MÄK (Clark et al, 2000).

Em Zalcman (2004) são apresentados alguns conceitos sobre DIS, porém o maior enfoque é com relação à utilização das PDUs. Zalcman apresenta um exemplo de utilização do DIS num simulador da força aérea australiana, traçando uma diretriz sobre o mínimo recomendado de PDUs para o simulador em questão, deixando claro que a escolha do número de PDUs dependerá do projeto. Este artigo discorre também sobre os métodos de migração do DIS para o HLA e sobre a escolha do DIS em programas da RAAF (*Royal Australian Air Force*) e RAN (*Royal Australian Navy*).

Hill (2005) propõe um guia sobre a utilização das PDUs, onde o enfoque é na apresentação das PDUs que foram criadas, porém ainda não pertenciam ao padrão IEEE 1278.1. Essas PDUs não padronizadas foram criadas para atender projetos ou requisitos específicos de sistemas. Algumas PDUs são comerciais, outras propriedades de governos, e as demais são disponíveis como *open-source*. Existe uma preocupação em se disponibilizar as PDUs mais utilizadas incluindo-as no padrão. Nesse documento, inicialmente são apresentadas as PDUs do padrão; em seguida, as que estão em propostas; e, por fim, algumas que não pertencem ao padrão.

MacCall & Murray (2010), membros da SISO (*Simulation Interoperability Standards Organization*) apresentam uma publicação com bastante detalhes sobre o DIS. Apresentam o histórico, definições e conceitos próprios do protocolo, informações sobre as PDUs, os documentos publicados pelo IEEE, as versões do

padrão, aplicações e os próximos passos relacionados à manutenção e utilização do DIS.

Rogerson (1997) é um trabalho de mestrado publicado pela Universidade de Toronto, no qual são apresentadas técnicas para se obter os melhores valores para estimação de posição (*dead reckoning algorithm*) e o número de passos para suavizar (*smothing steps*) a trajetória de um modelo de um helicóptero que pousa no convés (*deck*) de um navio, sendo que esses modelos rodam em rede utilizando o protocolo DIS e integrados aos simuladores *Sea King* e *Canadian Patrol Frigate Landing*. Esse trabalho se inicia com a descrição e apresentação dos conceitos relacionados aos DIS; a seguir, são apresentados os simuladores, descrição de como a interface DIS é implementada, são apresentadas as técnicas para melhorar os valores da trajetória do helicóptero; e, por fim, é apresentado um exercício de simulação com a conclusão sobre os resultados.

2.1.1.1. Versões do DIS

O DIS passou pelo processo formal de liberação do IEEE três vezes, 1993, 1995 e 1998, ocorrendo liberação *draft* entre esses processos (Clark et al, 2000). A última versão em processo é a de 2012 (IEEE, 2012).

As versões geradas foram:

- Versão 1.0: Maio de 1992 – *Standard for Distributed Interactive Simulation - Application Protocols - Draft*;
- Versão 2.0: IEEE 1278-1993
- Versão 2.0: Maio de 1993 - *Standard for Distributed Interactive Simulation - Application Protocols – Draft*;
- Versão 2.0: Março de 1994 - *Standard for Distributed Interactive Simulation - Application Protocols – Draft*;
- Versão 5: IEEE 1278.1-1995

- Versão 6: IEEE 1278.1A-1998

- IEEE 1278.1-2012: *DIS Product Development Group* (IEEE, 2012).

2.1.1.2. Terminologias do DIS

A seguir são apresentados os conceitos e terminologias relacionados ao padrão DIS relacionados a como são chamados os componentes (computadores e simulações), ambientes e formatação do protocolo.

a) Aplicação de Simulação (*Simulation Application*):

É o *software* que modela e simula todo ou parte de um fenômeno com propósito de treinamento ou experimento/ experimentação.

As aplicações de simulação recebem e processam informações relativas às entidades/ objetos criados pelas outras aplicações de simulação participantes do processo, sendo essa troca realizada através das trocas entre as PDUs.

É equivalente ao Federado (*Federate*) dos padrões HLA e TENA.

b) Computadores *Host*:

São computadores que suportam as Aplicações de Simulação.

Os computadores *Host* que participam de um exercício de simulação estão conectados através de redes, que podem ser LAN, WAN ou redes *Wireless*.

c) Objeto de Simulação (*Simulation Object*):

É o elemento que faz parte do ambiente sintético, criado e controlado pela aplicação de simulação e é afetado pelas trocas realizadas através das PDUs. Uma simulação de aplicação pode controlar mais de um objeto de simulação.

São entidades que representam objetos físicos. Os principais tipos de entidades são: aeronaves, navios, veículos, armas e humanos no ambiente são os principais tipos de objetos.

É equivalente ao objeto (*Object*) dos padrões HLA e TENA.

d) Exercício de Simulação (*Simulation Exercise*):

Consiste de uma ou mais interação entre as aplicações de simulação.

As simulações que participam do mesmo exercício de simulação recebem um número identificador chamado de identificador de exercício (*exercise identifier*).

É equivalente à Federação (*Federation*) dos padrões HLA e TENA.

e) Ambiente de Simulação (*Simulation Environment*):

É o ambiente operacional que envolve as entidades de simulação. Inclui terrenos, atmosfera, informação oceanográfica, etc.

2.1.1.3. Características do DIS

A seguir estão listadas as principais características do DIS:

a) É um protocolo de rede padronizado para unir simulações em tempo real com pessoas inseridas ou não na simulação (simulações do tipo *wargame*).

b) Os simuladores podem estar distribuídos geograficamente, sendo conectados através de uma grande rede WAN (*Wide Area Network*).

c) O DIS provê coerente representação sintética de tempo e espaço em ambientes que operam em tempo real.

d) As PDUs (*Protocol Data Units*) definem a sintaxe (formato e regras) dos dados, e a semântica (significado) para a troca de dados na rede e da interoperabilidade da simulação;

e) O ambiente sintético é criado através da interoperação de PDUs em tempo real, em simulações computacionalmente autônomas.

f) Nenhum computador central é requerido para controlar o exercício de simulação. No DIS, a responsabilidade pelo estado de cada entidade (tais como, tanque, submarino, navio aeródromo, aeronave, míssil) depende de aplicações de simulação separadas, executadas nos computadores que se comunicam através da rede.

g) As aplicações de simulação são autônomas e são responsáveis por manter o estado das entidades e objetos simulados. As aplicações são responsáveis por modelar as ações das próprias entidades, usando um modelo com alta fidelidade. Essa simulação é responsável por enviar mensagem aos outros, e informar qualquer atualização observada. As simulações participantes são responsáveis por interpretar e responder as mensagens de interesse dos outros simuladores no exercício, mantendo um modelo do estado das entidades do exercício de simulação. Essa autonomia também permite que componentes da simulação distribuída se juntem ou saiam, mesmo com a simulação em progresso, sem que ocorra uma quebra (*crash*) de operação.

h) Um protocolo padrão é utilizado para comunicar dados verdadeiros (*ground truth* - dados observados e não estimados). Cada parte da simulação comunica os dados observados de estados ou de controle das entidades (tais como, velocidade, localização, posição de partes articuladas) a outras simulações na rede. A simulação que recebe esses dados, é a responsável por tratar o dado recebido e identificar se a entidade que o enviou é detectável por meio visual ou eletrônico. Esse estado de percepção é então mostrado ao usuário, conforme cada aplicação individual (Zalcman, 2004).

i) Mudanças no estado das entidades/objetos devem ser comunicadas através das aplicações de simulação que as controla.

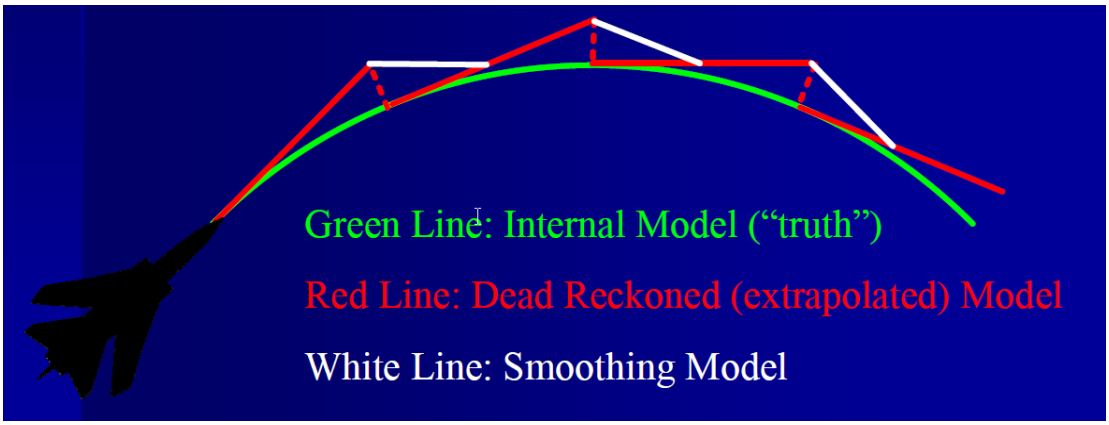
j) A percepção de eventos ou outras entidades/objetos é determinada pela aplicação receptora.

k) Algoritmos de estimação de posição (*dead reckoning*) são usados para reduzir o processo de comunicação. Esses algoritmos são métodos utilizados para estimar posição/orientação usando posições/orientações anteriores.

   Em simulações distribuídas esses algoritmos auxiliam na redução do carregamento dos recursos, como redes e máquinas, pois reduzem a necessidade da atualização das entidades, reduzindo a latência da comunicação.

Cada simulação mantém um modelo fiel da entidade que a mesma representa; do mesmo modo, a simulação mantém um modelo simplificado da entidade, a qual representa o modo como a mesma será vista por outras aplicações na rede. Esse último modelo é uma extrapolação da posição e orientação usando um algoritmo de estimação de posição próprio. Geralmente, a simulação compara os modelos de alta fidelidade e os valores estimados das entidades. Se a diferença entre os dois excede um valor predeterminado (*threshold*), a simulação vai atualizar o modelo mais simples usando a informação do modelo de alta fidelidade. Ao mesmo tempo, a simulação envia a informação atualizada para as outras simulações na rede, para que as mesmas possam atualizar seus modelos de entidade. Se uma entidade mantém um movimento, como por exemplo, velocidade constante, voo reto e nivelado, a taxa de atualização cai para uma taxa mínima predeterminada. Assim, utilizando esse método, as simulações não necessitam enviar atualizações a cada quadro (*frame*).

Figura 2.2 – Suavização após o Método de Estimação de Posição.



Fonte: MacCall & Murrey (2010).

2.1.1.4. Descrição das PDUs:

As simulações que utilizam DIS devem operar através da troca de dados entre as PDUs (*Protocol Data Units*). As PDUs são pacotes de dados transmitidos de um simulador a outro numa simulação distribuída, cujos formatos e tipos são especificados pelo protocolo DIS. Essa especificação é descrita por documentos conhecidos como documentos de enumeração do DIS (*DIS Enumeration Documents*) que estabelecem a sequência de transmissão dos bits em cada tipo de PDU (Herdman, 1995).

Ao longo dos anos, o padrão foi atualizado e aumentou o número de definições de PDUs de 27 até 67. Essas modificações permitiram que novas informações fossem incluídas. Por exemplo, novas versões do DIS incluem transmissão eletromagnética, facilitando a simulação de comunicações via rádio, informação de dados táticos e estratégicos sobre os inimigos, no ambiente de simulações de defesa (militares) (Herdman, 1995).

A última atualização do padrão IEEE 1278.1 DIS, apresenta 67 PDUs que podem ser divididas em grupos, de acordo com a informação que as mesmas gerenciam, definidas conforme a Figura 2.3.

As subdivisões são chamadas de famílias, como exemplificado na Figura 2.3, que podem ser informações/interações da entidade, logística, guerra (*Warefare*), gerenciamento da simulação, comunicações via rádio, geradas nas liberações (*releases*) do DIS.

Figura 2.3 – PDUs do Padrão IEEE 1278.1.

Fonte: Adaptado de Clark et al. (2000) e MacCall e Murrey (2010).

Nota: As PDUs de 68 a 225 estão disponíveis para experimentos e para outras aplicações necessárias, e ainda não fazem parte do padrão.

Como observado, as PDUs apresentam muita aplicação para simulações de sistemas militares, porém, o DIS também foi utilizado em aplicações de simulações de sistemas espaciais (Jense & Kuijpers, 1997).

Jense & Kuijpers (1997) discorrem sobre a utilização de DIS em uma aplicação espacial descrita em Martín (1996). Essa publicação examina os efeitos da utilização do algoritmo de estima de posição (*dead reckoning*) para calcular a posição de satélites na órbita. Como os algoritmos tradicionais são baseados no movimento linear em um sistema de coordenadas cartesiana, eles não são suficientemente precisos para os movimentos curvilíneos de espaçonaves. Ao longo da publicação são feitas comparações entre os valores obtidos com os algoritmos tradicionais e com o algoritmo de estimação de posição do DIS, concluindo com a recomendação de se utilizar o segundo que, adicionalmente, usa de maneira otimizada a largura de banda da rede.

2.1.1.5. Vantagens e Desvantagens do DIS

Diversas fontes bibliográficas enumeram as vantagens e desvantagens do DIS. Algumas até fazem comparações com relação a outros padrões, tais como o HLA.

As principais vantagens e desvantagens estão sumarizadas a seguir:

a) Vantagens

- O DIS provê um meio padronizado para interconectar vários simuladores. Com os conceitos das PDUs e a maturidade do DIS, muitas ferramentas para desenvolvimento já estão prontas. Atualmente existem muitas aplicações COTS (*Commercial-Off-The-Shelf*) para a utilização do DIS.

- O DIS padronizou enumeradores de entidades, sensores, dispositivos de comunicação e outros atributos. O cumprimento com o padrão de enumeração é mandatório para participar no exercício DIS. Sendo a lista mantida pela SISO (*Simulation Interoperability Standards Organization*). Essa enumeração inclui diversos países tais como Estados Unidos da

América (EUA), França, Reino Unido, Alemanha e até a desfeita União Soviética tem inventários cadastrados no DIS. Cada país recebe um número, Austrália, por exemplo, é 13, Estados Unidos da América, 225 e Rússia, 222.

- A utilização do algoritmo de estimação de posição (*dead reckoning*), o qual reduz o tráfego na rede, otimizando a comunicação. O padrão DIS conta com não apenas um, mas um conjunto desses algoritmos.

- Os formatos dos pacotes trocados na rede são bem definidos permitindo que simuladores e simulações possam interoperar com tempo reduzido de integração.

b) Desvantagens

- O DIS tem sido considerado rígido e não flexível para algumas aplicações; por isso, mais PDUs foram criadas, na tentativa de contornar essa característica. Essa ação ocasionou que alguns parâmetros se tornaram redundantes, dependendo da aplicação.

- Para atender as regras do DIS, todos os campos das PDUs devem ser preenchidos e de forma correta o que, para sistemas em escalas maiores e mais integrados, irá requer muito recurso computacional e largura de banda de rede. Essa característica do DIS é também conhecida como problema de latência (*SISO DIS Guide*, 2007).

- O DIS apresenta suporte limitado para agregar e desagregar entidades e foi desenvolvido especificamente para aplicações em tempo real, tais como simuladores.

- O código não é reutilizável entre diferentes fornecedores; dessa forma, não garante interoperabilidade quando numa simulação há participantes de fornecedores diversos.

- Não há um nível de segurança (*security*), pois as PDUs são padrões publicados e qualquer aplicação de simulação no formato DIS pode entrar e participar da simulação e obter dados a que se desejava sigilo, reduzindo a integridade da informação.

2.1.2. HLA (*High Level Architecture*) Flexibilidade e Transparência nas Regras de Comunicação

O HLA, assim como o DIS, é uma arquitetura de *software* padronizada, a qual torna possível a interoperação de diferentes simuladores, operando em uma simulação distribuída.

O HLA foi concebido com o objetivo de ser um padrão internacional aberto, sendo desenvolvido pela SISO e publicado pelo IEEE. O processo de desenvolvimento é aberto e transparente. Dessa forma, a filosofia de desenvolvimento do HLA era de ser um processo aberto e transparente, de tal maneira que qualquer entidade ou pessoa poderia participar, seja através de sugestões e até mesmo participando em fóruns (Möller, 2012).

O HLA também prometia ser menos rígido que o DIS pois, como arquitetura, ele abstrai para um nível mais alto a formatação dos dados e o modo como esses dados são trocados entre as simulações.

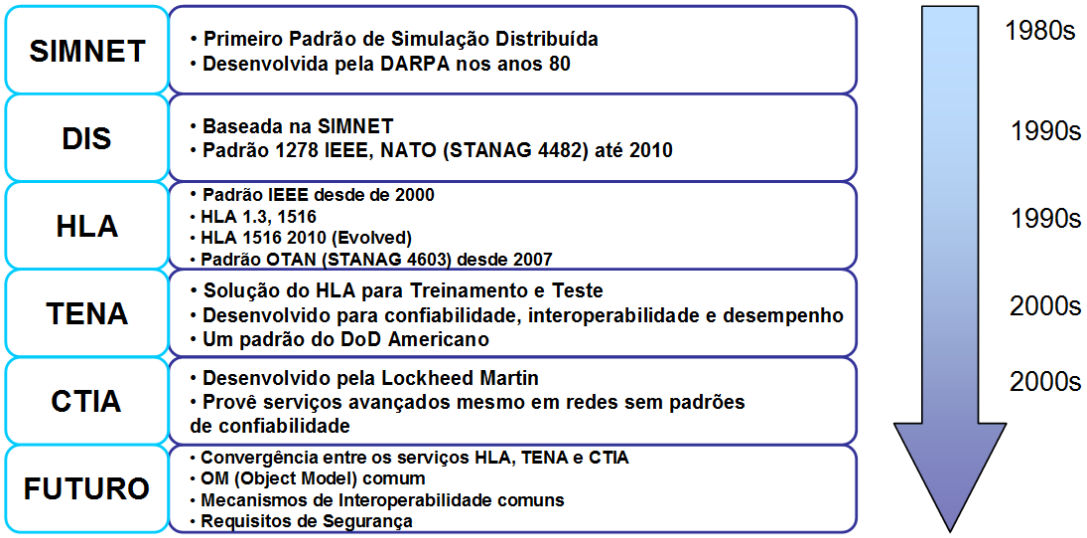
A flexibilidade do HLA é viabilizada através dos FOMs (*Federation Object Model*), os quais permitem que as simulações definam seus próprios objetos e interações e também pela RTI. A RTI (*Run Time Infrastructure*) é o *software* que provê serviços, tais como gerenciamento do tempo, gerenciando as transmissões de dados (VT-MÄK, 2016), podendo-se fazer uma analogia com um sistema operacional que provê serviços a outros pacotes de *software* e usuários.

As terminologias associadas ao HLA estão descritas em detalhes mais adiante, bem como os serviços que a RTI provê numa simulação que utiliza HLA.

A evolução do HLA é apresentada em Salio et al (2012), através de um breve histórico acerca dos projetos que culminaram nas arquiteturas para simulações distribuídas. O histórico se inicia com o projeto SIMNET (*Simulation Network*) nos anos 80, como uma iniciativa do governo norte-americano de investir em simulações. Do SIMNET surgiu o DIS; em seguida, o HLA surgiu da junção do ALSP e DIS. Ao longo dos anos o HLA evoluiu apresentando os padrões 1.3, 1516-2000 e 1516-2010 (*Evolved*).

Depois do HLA, os padrões de maior importância que surgiram foram o TENA e o CTIA.

Figura 2.4 – Sumário do Histórico dos Padrões de Simulação Distribuída.



Fonte: Adaptado de Salio et al (2012).

Devido ao grande sucesso do DIS, o mesmo se tornou um padrão OTAN (*NATO - North Atlantic Treaty Organization*) até 2010, sendo um dos padrões de simulação distribuída mais utilizados até os dias atuais.

Möller et al (2010) apresenta as principais melhorias implementadas no HLA 1516 *Evolved* em relação ao 2000, dentre as elas:

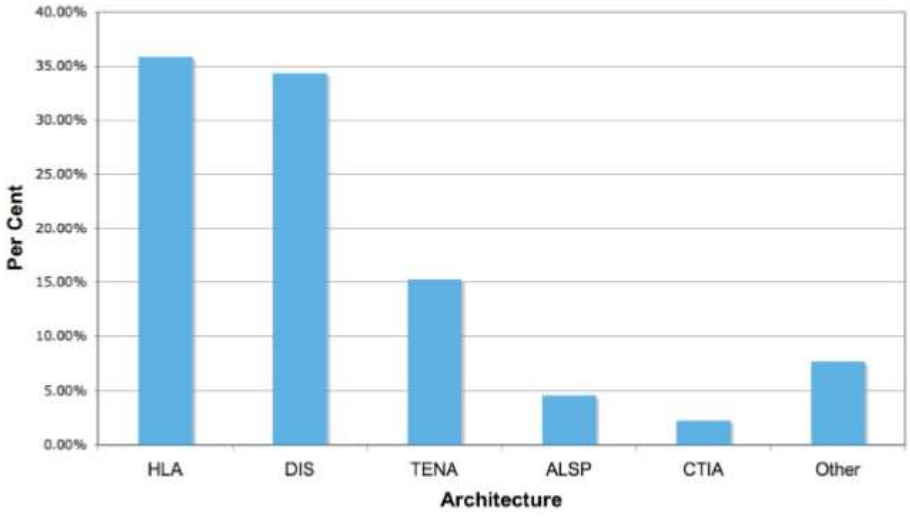
- FOMs modulares, facilitando reuso de FOMs;
- Tolerância a falhas que inclui uma semântica bem definida para a RTI comunicar ao federado a perda de conexão ou se outro federado perdeu a conexão;
- API de serviços de *Web* para que seja possível ao federado chamar a RTI usando serviços de rede ao invés de um API em C++ ou Java;

Möller et al (2010) também apresentam um sumário das experiências observadas na migração do HLA 2000 para 2010, mostrando uma espécie de roteiro para essa implementação. Esta mesma publicação apresenta estudos de caso, com resultados de simulações reais.

Henninger et.al. (2009) apresentam um relatório do estudo e pesquisa sobre as melhorias em ambientes de simulação *Live, Virtual and Constructive* (LVC). Neste relatório são apresentadas a metodologia, a característica dos grupos envolvidos na pesquisa, a formatação da coleção dos dados, os detalhes sobre os *workshops*, o estado atual das principais arquiteturas utilizadas e recomendações.

Neste relatório, um dos dados apresentados foi o percentual de utilização de cada um dos principais padrões de simulação distribuída, dentre as entidades que utilizavam esses padrões. No total foram 135 entidades interrogadas, entre militares e civis, sendo que essas entidades utilizavam simulação distribuída para variadas aplicações: treinamento, testes e experimentação. O resultado obtido está apresentado na Figura 2.5.

Figura 2.5 – Utilização dos Padrões de Simulação Distribuída.



Fonte: Salio et al. (2012).

Do gráfico observa-se uma preferência pelo HLA, seguida do DIS. O DIS ainda é bastante utilizado por ser mais antigo, existindo assim muitas ferramentas já desenvolvidas que o utilizam. O TENA também apresenta uma utilização significativa de aproximadamente 15%, principalmente pela utilização em aplicações militares.

Dahmann et al, (1997) apresenta os principais conceitos relacionados ao HLA. A publicação se inicia com um histórico das simulações distribuídas, descreve o processo de desenvolvimento do HLA, descreve a funcionalidade, as regras, os softwares de suporte (RTI e ferramentas de *Object Model*) e algumas diretrizes para o teste de cumprimento com o HLA (aplicado pelo DMSO).

2.1.2.1. Documentos do HLA

Os quatro documentos abaixo especificam de forma exata e completa as regras do HLA sem entrarem em muitos detalhes em termos de linguagens e implementações (IEEE 1516, 2010).

- Regras (*Rules IEEE 1516-2010*): estabelece o comportamento da federação e federados [IEEE-HLAA]. Contém princípios e convenções que devem ser seguidas para ocorrer a correta interação entre os federados durante a Execução da Federação. Esse documento apresenta no total dez regras. Cinco regras são relacionadas à Execução da Federação (*Federation Execution*), e outras cinco regras são específicas dos Federados. Esse documento também descreve as responsabilidades dos Federados.

- Especificação de *Interface* (*Interface Specification IEEE 1516.1-2010*): define a interface RTI-Federado (*RTIAmbassador*) e a interface Federado-RTI (*FederateAmbassador*) [IEEE-HLA1].

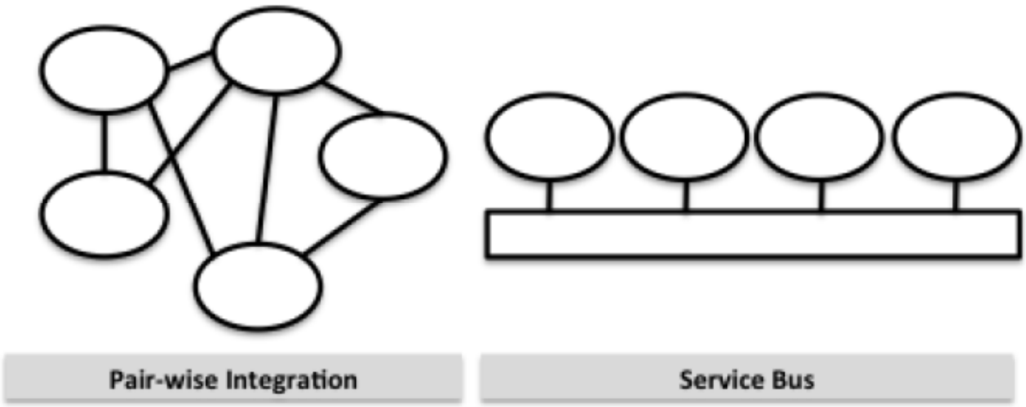
- *Template* do Modelo de Objeto (*Object Model Template IEEE 1516.2-2010*): define o formato dos FOMs [IEEE-HLA2].

- Processo de Desenvolvimento e Execução da Federação (*Federate Execution and Development Process IEEE 1516.3-2010*), o qual provê um processo para desenvolvimento das simulações que utilizam HLA [IEEE-HLA3].

2.1.2.2. Topologia do HLA

O HLA apresenta uma topologia de comunicação do tipo Barra de Serviço (*Service Bar*) ao invés da Topologia de Pares (*Pair-wise*). A Figura 2.6 apresenta as duas topologias para exemplificação e ilustração.

Figura 2.6 – Topologias Pares e Barra de Serviço.



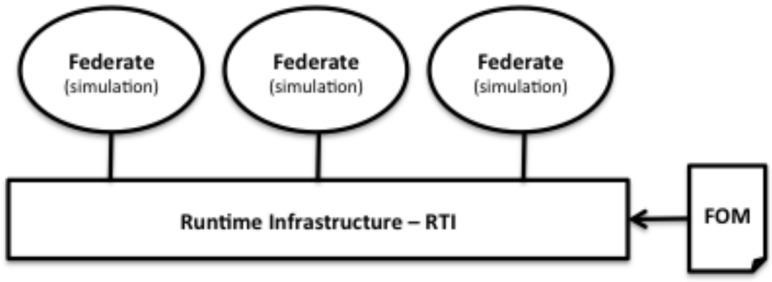
Fonte: Möller (2012).

A topologia de barra de serviços apresenta algumas vantagens sobre a topologia de pares. Por exemplo, na topologia de pares, os sistemas trocam serviços em pares, sendo que para cada par que troca informação, deve-se estabelecer um acordo sobre como a informação será trocada, tornando a comunicação muito complexa quando se aumenta o número de conexões nessa rede. Outra desvantagem é que os sistemas envolvidos devem saber mutuamente sobre todos os sistemas com os quais haverá troca de informação, aumentando o custo de modificação quando se deve acrescentar outro sistema (Möller, 2012).

Nesse aspecto, o HLA propõe simplificação, portanto utiliza a topologia de barra de serviços. A simplificação se dá no fato de que cada sistema apresenta apenas uma conexão na simulação, sendo essa feita através da barra de serviços. Por esta barra de serviço, um conjunto de serviços é provido e cada sistema só utiliza o serviço necessário. Além disso, essa topologia permite o acréscimo de sistemas, facilitando o reuso das simulações em novas combinações.

A topologia do HLA é comumente chamada de *Lollipop* (pirulito), pela aparência da representação (Möller, 2012) mostrada na Figura 2.7.

Figura 2.7 – Topologia do HLA em *Lollipop*.



Fonte: Möller (2012).

Sendo assim, na topologia do HLA, os sistemas são interconectados através de uma barra de serviços, chamada de RTI (*Run Time Infrastructure*). O FOM faz o papel de apresentar o conjunto de informações (dados) que se deseja trocar, onde cada conjunto de sistema apresenta o seu próprio FOM.

2.1.2.3. Terminologias do HLA

Os conceitos a seguir sugiram com padrão HLA e são importantes para o entendimento do funcionamento dessa arquitetura:

a) RTI (*Run Time Infrastructure*):

RTI é o software que administra a interconexão das simulações e provê a especificação de interface entre elas. Este software provê os mecanismos de comunicação para as entidades (Federados) e permite que os mesmos participem da simulação. Todas as aplicações em HLA devem rodar através de uma RTI. No total, a RTI provê sete serviços aos Federados, esses serviços estão descritos no capítulo 2.1.2.4 mais adiante.

b) Federado (*Federate*):

É um sistema que se conecta à RTI, geralmente um simulador. Os Federados podem modelar um grande número de objetos na simulação. Por exemplo, o

Federado pode ser uma aeronave ou um conjunto de centenas de aeronaves. Outros tipos de Federados são ferramentas gráficas para visualização 3D.

c) Federação (*Federation*):

A Federação é um conjunto de Federados com um FOM em comum e suportados por uma RTI. Consiste de um grupo de Federados capazes de operar simultaneamente na execução da federação (VT-MÄK, 2016).

d) Acordo da Federação (*Federation Agreement*):

Corresponde ao documento que descreve exatamente como os Federados irão trocar serviços. Em Möller (2012), esse acordo é chamado de “contrato exato” ou “documento de *design*”.

e) Modelo de Objeto da Federação (*FOM - Federation Object Model*):

É um conjunto de classes de objetos e interações, suportados por um grupo particular de simulações (VT-MÄK, 2016).

É um arquivo que contém uma descrição da troca de dados da Federação, por exemplo, os objetos e as interações que serão trocadas. Em Möller (2012), o FOM é exemplificado como a linguagem da Federação.

O FOM não contém todos os dados dos Federados, mas apenas as informações que os Federados irão trocar.

O FOM faz parte do Acordo da Federação.

Simulações de áreas diferentes irão ter FOMs diferentes, contento diferentes tipos de dados que serão trocados.

O FOM deve conter pelo menos as seguintes informações:

- Classe dos Objetos (*Object Classes*): apresenta os atributos que podem ser atualizados no tempo. Exemplos são: nome, posição e velocidade;

- Classe das Interações (*Interaction Classes*): existe por um breve momento, geralmente apresenta parâmetros, exemplos são: início (*start*), parada (*stop*), explosões e comunicações de rádio.

- Tipo dos Dados (*Data Type*): descreve a semântica e a representação técnica de uma classe de objeto e parâmetros de uma interação. Existem vários tipos pré-definidos no HLA.

- Versões do FOM: nome dos autores, datas, etc., são informações que podem ser incluídas no FOM, porém não são obrigatórias.

O FOM pode ser estendido para incluir mais federados sem que haja quebra da simulação.

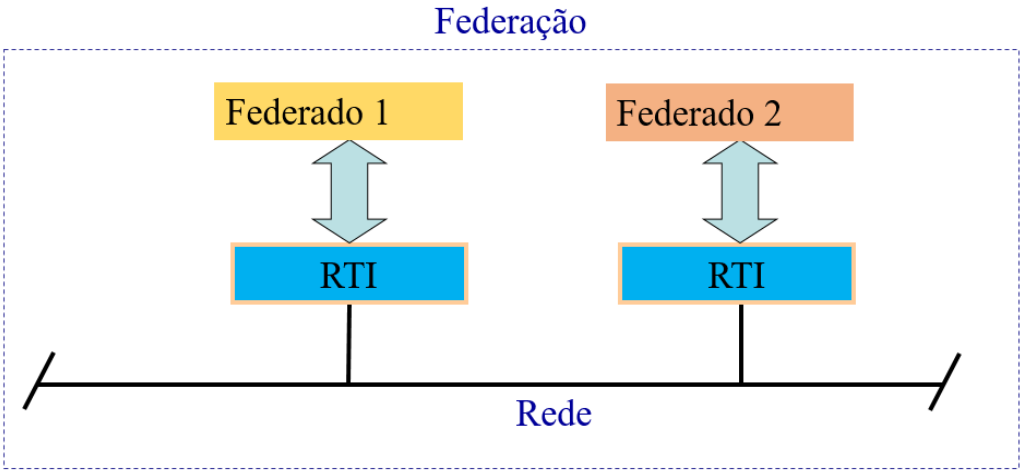
O FOM segue um formato chamado de OMT (*Object Model Template*), o qual é escrito no formato de XML. Este formato é descrito no padrão HLA.

f) Execução de Federação (*Federation Execution*):

Uma execução de Federação é uma sessão ou um exercício da simulação. Quando se roda uma Federação diversas vezes, se obtém diferentes execuções de Federação, podendo conter diferentes cenários.

Em VT-MÄK (2016) uma execução de Federação é descrita como o cenário atual de operação no tempo de um subconjunto de federados e dos dados de inicialização de uma RTI obtidos de uma federação particular. É equivalente ao exercício de simulação do DIS.

Figura 2.8 – HLA



Fonte: Adaptado de VT-MÄK (2016).

g) *Template* do Modelo de Objeto (*Object Model Template - OMT*):

O OMT estabelece um padrão para documentar a Informação do Modelo de Objeto (*Object Model Information*) do HLA. O OMT define o Modelo de Objeto da Federação (FOM), o Modelo de Objeto da Simulação (ou Federado) e o Modelo de Gerenciamento de Objeto (MOM - *Management Object Model*).

O Modelo de Objeto da Simulação (SOM - *Simulation Object Model*) é uma porção do FOM usado em cada Federado.

2.1.2.4. Serviços da RTI

Os sete serviços providos pela RTI estão apresentados a seguir:

a) Gerenciamento da Federação: administra a criação, o controle dinâmico, a modificação e a exclusão de uma execução de federação;

b) Gerenciamento da Declaração: habilita os federados a declararem à RTI a intenção de publicar ou receber (subscrever/ refletir) estados de objetos e informação de interações. Federados podem se subscrever a apenas

objetos que eles desejam receber informações. Por exemplo, aeronaves podem querer receber informações referentes apenas a atividades relacionadas a voos, assim como, tanques podem querer receber apenas informações relacionadas a movimentos terrestres (Clark et al, 2000);

c) Gerenciamento do Objeto: torna possível a criação, modificação e interação dos objetos. Esse é o serviço que ocupa mais a rede e gera mais tráfego durante a simulação;

d) Gerenciamento da Propriedade: permite que os federados transfiram a propriedade de atributos de objetos a outros participantes na simulação;

e) Gerenciamento do Tempo: provê serviços para ajustes, sincronização e modificação dos relógios (*clocks)* de simuladores. O serviço de gerenciamento do tempo é casado com o serviço de gerenciamento do objeto, de tal forma que atualizações de estado e interações são distribuídas em ordem cronológica;

f) Gerenciamento da Distribuição de Dados: os federados podem ditar as condições de quando começar ou parar de transmitir ou receber certos dados;

g) Serviços de suporte a funções de utilidades;

Os serviços de gerenciamento da Federação, da declaração, do objeto e da propriedade são funcionalidades semelhantes às funcionalidades e Informação de Entidade (*Entity Information*), Gerenciamento de Entidade (*Entity Management*) e Gerenciamento de Simulação no DIS, porém com uma arquitetura superior (Clark et al., 2000).

Os serviços de Gerenciamento do Tempo e da Distribuição de Dados, não têm equivalência no DIS. O DIS é implicitamente assume que as interações são em tempo real e entre sistemas sincronizados com um mecanismo de *broadcast* para distribuição dos dados.

Esses serviços tornam a RTI muito vantajosa quando comparada com o tradicional DIS.

2.1.2.5. RTIs Existentes

Reis (2009) apresenta uma lista de RTIs então disponíveis, algumas comerciais (

Tabela 2.1) e outras não comerciais (*softwares open-source* ou livres) (Tabela 2.2).

Tabela 2.1 – Licenças Comerciais de RTIs.

Nome	Fornecedor/ Desenvolvedor	Padrão	Interface	Tipo de Licença
Chronos RTI	Magnetar Games	IEEE 1516	C++, NET	Comercial
MÄK-RTI	VT MÄK	1.3, IEEE 1516-2000,	C/C++, JAVA	Comercial
Openskies	Cybernet Systems	1.3, IEEE	C++	Comercial
pRTI	Pitch Technologies	1.3, IEEE 1516-2000,	JAVA, C++, *Web Services*	Comercial
Mitsubishi ERTI	Mitsubishi Electric Corp. and Mitsubishi Space Software Co. Ltd	1.3	C++	Comercial
RTI NG Pro	Raytheon Virtual Technology Corporation	1.3, IEEE 1516-2000, IEEE-1516-2010 (*Evolved*)	C++, JAVA	Comercial
CAE RTI	CAE Inc.	1.3, IEEE 1516	C++, .NET	Comercial
RTI NG Pro	Raytheon Company	1.3, IEEE 1516-2000, IEEE-1516-2010	C++, JAVA	Comercial

Fonte: Reis (2009).

Tabela 2.2 – Licenças Não comerciais de RTIs.

Nome	Fornecedor/ Desenvolvedor	Padrão	Interface	Tipo de Licença
CERTI	ONERA	1.3 parcial, IEEE	C++, MATLAB, Fortran90	GPL, LGPL
MATREX RTI	Dynamic Animation Systems	1.3	C++, JAVA	GPL, LGPL
EODiSP HLA	P&P Software	IEEE	JAVA	GPL
PORTICO	OPEN LVC	1.3, IEEE	JAVA,	CDDL
OpenRTI	Flight Gear Project	1.3, IEEE 1516-	C++	*Lesser General*
RTI-S	US JFCOM J9 Directorate	1.3, IEEE 1516	JAVA, C++	US *Government*
Rendezvous RTI	National University of Sciences and Technology (NUST), Pakistan	1.3	C++, JAVA	NUST

Fonte: Reis (2009).

A RTI NG que era disponibilizado pelo DMSO foi descontinuada (Reis, 2009).

Möller (2010) apresenta a quantidade de RTIs existentes na época, estimando a existência de cerca de vinte e cinco RTIs mais conhecidas, sendo a maioria COTS e comercialmente disponíveis, cerca de dez com manutenção muito boa e outros dez com manutenção razoável.

2.1.2.6. Vantagens e Desvantagens do HLA

a) Vantagens:

- O HLA surgiu das tentativas de se contornar as deficiências do DIS. Assim, uma das vantagens do HLA é que a definição dos dados que devem ser enviados pela rede é feita pelos membros da federação, através dos mecanismos de publicar e subscrever do HLA;

- Qualquer sistema de coordenadas pode ser usado ao invés do 3D DIS sistema geocêntrico;

- O HLA suporta gerenciamento do tempo real e do tempo virtual, suportando interações entre ambos os tipos de simulação, virtual e construtiva, que possam utilizar tempo não real;

- O HLA reduz a necessidade de largura de banda da comunicação, pois há uma escolha dos dados a serem transmitidos e quais os atributos devem ser atualizados;

- Como o conjunto de dados é definido por um FOM específico, apresenta um nível de segurança maior que o do DIS. A interpretação dos dados só é possível através do conhecimento dos dados e formatos do FOM em questão.

b) Desvantagens:

- O HLA é mais flexível que o DIS, porém todos os federados devem concordar em um FOM; de outra forma, não conseguirão interoperar, mesmo que estejam cumprindo com as regras do HLA. Logo, estar cumprindo com as regras do HLA não significa interoperabilidade;

- Cada FOM precisa de um conjunto de enumerados, o qual no DIS era provido pelo padrão. Cada FOM é único e requer desenvolvimentos de ferramentas (*softwares*), pois atualmente não há uma ferramenta COTS para isso.

2.1.3. TENA (*Test and Training Enabling Architecture*) e CTIA (*Common Training Instrumentation Architecture*)

TENA (*Test and Training Enabling Architecture*) também é uma arquitetura distribuída idealizada pelo DoD. Da mesma maneira, foi idealizada para atender baixa latência, alto desempenho em aplicações em tempo real (Henninger, 2008).

O TENA foi definido como padrão de arquitetura para simulações distribuídas em 2010, sendo seu desenvolvimento patrocinado pelo Ministério da Defesa americano (Petz, 2010).

O TENA e CTIA deveriam corrigir alguns problemas do HLA, relacionados principalmente a qualidade da distribuição dos dados e do gerenciamento de rede. Esses padrões, no entanto, não são tão utilizados, tendo mais aplicações em programas desenvolvidos e financiados pelo governo americano (Salio, 2012).

Hollenback (2009) discorre sobre as características que o TENA deveria ter, de forma a justificar a necessidade dessa nova arquitetura. Ele esclarece que no início a diretriz era de que o TENA e o HLA deveriam ser complementares em propósito, projeto, desenvolvimento e implementação. Porém o TENA acabou divergindo do HLA, não porque o HLA não pudesse prover as funcionalidades necessárias, mas para evitar a utilização de RTIs COTS as quais, apesar de mais

maduras, requerem a necessidade de investimento financeiro recorrente para a manutenção dos mesmos.

Atualmente o TENA é mais empregado para atividades militares, tendo em vista que foi desenvolvido para isto, ou seja, seguindo requisitos especificamente de sistemas de defesa (Salio, 2012).

O TENA deveria ser similar ao HLA/RTI para muitas implementações de simulação distribuída, porém duas características não seriam bem exploradas pelo TENA: gerenciamento de tempo (*Time Management*) e *game-pause* ou pontos de sincronização. A explicação é simples, os patrocinadores do TENA simplesmente não têm interesse nessas características (Noseworthy, 2010).

Existe também a arquitetura CTIA (*Common Training Instrumentation Architecture*), com conceito muito similar aos das arquiteturas de simulação distribuída, porém muito mais próximo ao TENA. Tem sido usada para conectar um grande número de recursos em tempo real (*live*) e tem sido usada para suportar exercícios e treinamentos das forças armadas (Noseworthy, 2010).

O Mediador (*Middleware*) do TENA pode ser baixado gratuitamente, porém é necessário um cadastro de senha e usuário, para a criação de uma conta, devido a restrições de exportação. O pedido é analisado pela comunidade TENA SDA ORG, para verificação de acordo com as diretrizes de exportação (*compliance with export guidelines*). O acesso ao *download* é feito através da página: //www.tena-sda.org/display/TENAintro/Home.

Figura 2.9 – Página da Comunidade TENA SDA.

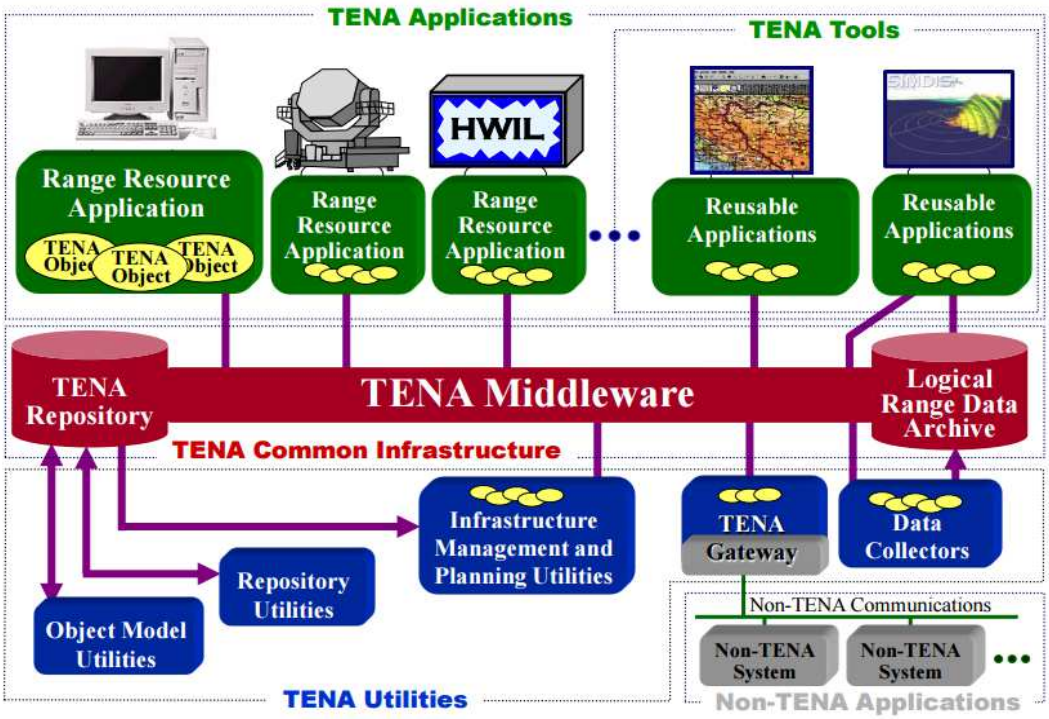
Fonte: TENA (2016).

Atualmente existem ferramentas comercialmente disponíveis para interconectar DIS, TENA e HLA, servindo como tradutores (*gateways*), exemplos são os *softwares* VR-Link e VR-Exchange, ambos fornecidos pela VT-MÄK.

2.1.3.1. Arquitetura do TENA

O TENA tem uma interface comum que interliga as simulações. Esta interface é composta de: Mediador (*Middleware*), Repositório (*Repository*) e Arquivo Lógico de Dados (*Logical Range Data Archive*), como mostrado na Figura 2.10 (Petz, 2010).

Figura 2.10 – Arquitetura do TENA.



Fonte: Petz (2010).

O componente principal é o Mediador TENA (TENA *Midleware*), o qual utiliza UML (*Unified Modeling Language*) como modelo para geração automática de código e também apresenta APIs (*Application Programming Interfaces*) e componentes reutilizáveis (Petz, 2010).

O Repositório TENA (TENA *Repository*) guarda os dados relevantes sobre o TENA, incluindo padrões TENA de definição de objetos, meta-dados implementados, ferramentas e utilidades próprias, biblioteca de *softwares* para o TENA *Middleware*. Contém *templates* do TENA em UML ou na própria Linguagem de Definição TENA, TDL (*TENA Definition Language*), cujas representações são transformadas em código C++ e são apresentadas no TENA *Middleware*.

O Arquivo Lógico de Dados TENA (*Logical Range Data Archive*) inclui arquivos necessários para conectar as simulações e pode ser dividido entre várias máquinas.

Um dos usuários do TENA é o Ministério da Defesa da Eslováquia, que utiliza o padrão para treinamento de militares (Petz, 2010).

2.1.3.2. Conceitos Gerais:

a) LVC (*Live, Virtual, Constructive*) podem ser definidos como três tipos de simulação que integram: L) entidades reais; V) simuladas; C) em um ambiente sintético; em tempo real ou tempo virtual;

L (*Live*) - entidades físicas reais (por exemplo, não simulado, ou uma aeronave real, hardwares).

V (*Virtual*) - simuladores, tais como ambientes virtuais emulando entidades físicas reais, operados por um humano (por exemplo, um simulador de uma aeronave).

C (*Constructive*) - mundo puramente sintético, um número arbitrariamente muito grande de entidades interagem seguindo modelos complexos, como por exemplo, em jogos de guerra (treinamentos militares).

Exemplos de sistemas LVC:

- Piloto em um caça real, mas treinando (*LIVE*);

- Dois pilotos em simuladores de caça, agindo como em uma missão (*Virtual*);

- Ambos engajados em um ambiente completamente sintético, com inimigos simulados (*Constructive*).

b) *Middleware/ Wrapper:* ou Mediador, no campo da computação distribuída, é um programa de computador que faz a mediação entre um *software* e demais aplicações. É utilizado para mover ou transportar informações e dados entre programas de diferentes protocolos de comunicação, plataformas e dependências do sistema operacional.

c) *Gateway/ Translator:* é um *software* ou conjunto de *softwares* que implementam uma interface entre duas aplicações para que ocorra a comunicação de dados; é uma espécie de tradutor.

d) UML: Na área de engenharia de *software*, a Linguagem de Modelagem Unificada (UML - *Unified Modeling Language*) é uma linguagem de modelagem que permite representar um sistema de forma padronizada (com intuito de facilitar a compreensão pré-implementação) de forma a torna possível a geração automática de códigos. Atualmente, já existem ferramentas que são capazes de traduzir o UML de forma automática para códigos em linguagens de alto nível, tais como C++. O processo inverso também é possível utilizado, ou seja, é pode-se reverter códigos C++ ou Java para UML.

e) API (*Application Programming Interface*): é uma aplicação de *software*, que apresenta um conjunto de definições de sub-rotinas, protocolos e ferramentas para se desenvolver uma aplicação de *software*. Em geral, é um conjunto de definições para a comunicação entre vários componentes de *software*. A especificação de um API pode apresentar vários formatos, mas geralmente apresenta especificações de rotinas, estrutura de dados, classes de objeto e variáveis. Exemplos de diferentes formas de API são:

POSIX, Microsoft Windows *API, C++ Standard Template Library e Java APIs*.

f) GNU *General Public License* (GPL): um código licenciado sob GLP não pode ser incluído em *softwares* que tenham propriedade. Qualquer modificação deve ser submetida à comunidade OSS (*Open Source Software*) para que seja aprovada e emitida (*released*) (Temizer, 2007);

g) *Library* (ou *Lesser*) GPL (LGPL): Esse tipo de licença é derivado da GPL, permite integração do código fonte com *softwares* que apresentam propriedade (Temizer, 2007);

h) *Berkeley Software Distribution* (BSD): impõem poucas restrições ao usuário e permite integração do código fonte com *softwares* que apresentam propriedade (Temizer, 2007).

2.2. Exemplos de Utilização de Simulações HLA/RTI em Cenários Espaciais.

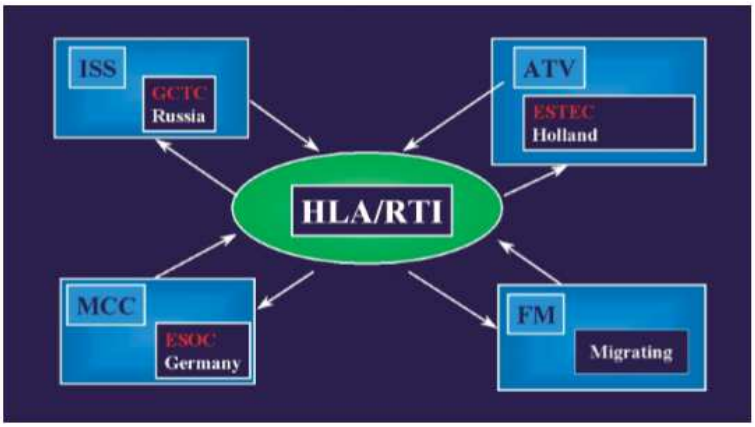
Argüello & Miró (2000) apresentam uma publicação não apenas com conceitos relacionados à simulação distribuída, mais especificamente HLA/RTI, mas também uma análise dos benefícios dessa utilização em aplicações no domínio do espaço, sendo esse trabalho o primeiro no ramo espacial, utilizando HLA em exemplos práticos. Os exemplos práticos contaram com a colaboração de alguns países europeus e da Rússia (Figura 2.11).

A figura de mérito para a análise foi se o erro introduzido pela simulação distribuída poderia ser mantido dentro de limites aceitáveis. Essa análise foi feita comparando-se resultados da simulação distribuída com resultados de simulação não distribuída.

O primeiro exemplo é a Simulação do Encontro e Acoplamento *(RVD, Rendez-Vous and Docking)* entre um Veículo de Transferência Automática (*ATV, Automated Transfer Vehicle*) e a Estação Espacial Internacional (*ISS, International Space Station*). Esse exemplo foi escolhido por ter um requisito de

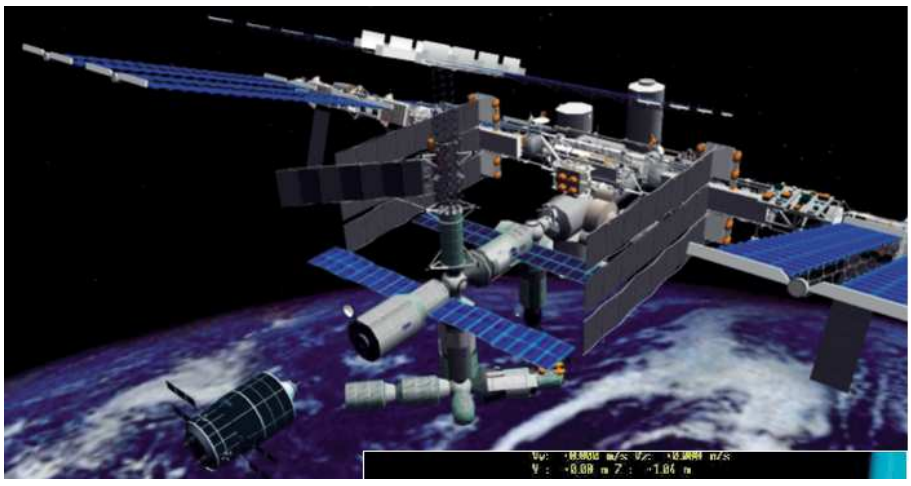
precisão elevada. Os resultados obtidos demonstraram a adequação do processo de simulação distribuída, não havendo perdas significativas em termos de atrasos, devido a processamentos computacionais, quando comparado com simulações que não utilizavam o HLA (Figura 2.12 e Figura 2.13).

Figura 2.11 – Simulações Geograficamente Distribuídas no exemplo do ATV e da ISS.



Fonte: Argüello & Miró (2000).

Figura 2.12 – Visualização tridimensional do ATV e da aproximação da ISS durante a Simulação.



Fonte: Argüello & Miró (2000).

Figura 2.13 – Vista do ATV através da câmera ISS.

Fonte: Argüello & Miró (2000).

O outro experimento utilizou uma pequena missão de um satélite, com o propósito de avaliar a aplicação do HLA para familiarização e treinamento de usuários de carga útil. Nesse experimento, os modelos do simulador de missão rodavam separadamente dos modelos de tarefas de controle e monitoramento (telecomando, telemetria e *displays* de visualização 3D). Na Holanda ficou o Centro de Controle e no Reino Unido era simulado o Centro do Usuário.

Em ambos os exemplos, a utilização do HLA se demonstrou viável, de fácil disponibilidade e apresentou custos e tempo adequados para instalação de equipamentos. No caso da simulação do ATV, as análises demonstraram ganhos de produtividade e uma redução em até 20% em tempo de desenvolvimento através da colaboração entre engenharias durante essa fase, dependendo do escopo e da duração da campanha de simulação.

Falcone, Garro, Longo & Spadafora (2014) é uma publicação do IEEE que tem como proposta apresentar o projeto UNICOM. O objetivo deste projeto é, utilizando o HLA, simular num cenário de uma base localizada na Lua (i) serviços de comunicação disponíveis para entidades nesta base e (ii) uma visualização 3D em tempo real deste cenário, tudo através de uma conexão RTI (*Run Time*

Infrastructure), fornecido pela VT-MÄK e uma unidade 3D. O principal objetivo da publicação é compartilhar a experiência da Universidade da Calábria (Itália) em participar do evento SEE (*Simulation Exploration Experience*).

O evento SEE é organizado pela SISO (*Interoperability Standards Organization*) em colaboração com a NASA entre outros parceiros industriais, cujo objetivo principal é promover a adoção do HLA e das ferramentas que cumprem com o padrão, através do envolvimento de times universitários em um projeto específico que simula uma base lunar, utilizando especificamente o HLA. No ano de 2014, universidades de diversos países contribuíram com modelos relacionados ao cenário espacial, conforme abaixo:

- Universidade do Alabama (EUA) contribuiu com três federados: (i) uma constelação de satélites de comunicação (*Comm. SAT*), (ii) um *drone* de carga para transporte de materiais (*Cargo Transport*) e (iii) um veículo usado para lançar o *drone* na órbita lunar (*Lunar Mass Driver*);

- Universidade Estadual da Pensilvânia (EUA) apresentou um veículo espacial (ônibus lunar) que inclui comunicação, subsistemas térmico e de potência (*Lunar Shuttle*);

- Universidade de Nebraska (EUA) criou um modelo de veículo que pode transferir carga entre instalações e o ônibus lunar (*Cargo Rover*);

- Universidade de Munique (Alemanha) desenvolveu um posto (*Outpost*) localizado no ponto dois de libração (*L2-Outpost*);

- Instituto Politécnico da Califórnia (EUA) criou um modelo de um elevador que vai da base lunar ao posto L2 (*Space Elevator*);

- Universidade de Bordeaux (França) criou um modelo de instalação para fornecedores (*Supply Depot*);

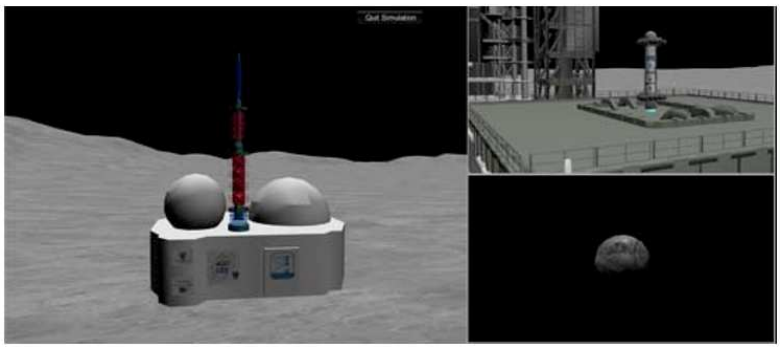
- Universidade de Brunel (Inglaterra) desenvolveu três federados, (i) uma simulação em tempo real de um astronauta que transporta regolito de uma mina a uma fábrica (*Astronaut*), (ii) uma fábrica que extrai oxigênio e outros gases de regolito lunar (*Factory*) e (iii) a simulação da mineração de regolito (*Mine*);

- Universidade de Genoa (Itália) produziu (i) um sistema de defesa contra asteróides (*IPHITOS*), o qual inclui um sistema de detecção de asteróides e uma base para lançamento de mísseis e (ii) uma instalação médica que checa constantemente o estado de saúde dos astronautas (*SISMA*);

- Universidade da Calábria (Itália) desenvolveu dois federados um para os serviços de comunicação e o outro para visualização 3D em tempo real (UNICOM).

O cenário escolhido na publicação foi o de uma situação de emergência em que um asteróide iria colidir com a Lua, foi chamado de cenário de alerta de asteróide. No caso a solução é interceptar e destruir o asteróide com um míssil. Essas atividades são realizadas pelo Federado base de míssil IPHITOS (desenvolvido pela Universidade de Genoa, Itália) que recebe os alertas do UNICOM (desenvolvido pela Universidade de Calábria, Itália) (Figura 2.14 e Figura 2.15). De fato, o Federado UNICOM envia o alerta de asteróide a todas as entidades da base lunar. Cada entidade reage de uma maneira, por exemplo, o Federado Posto L2 (desenvolvido pela Universidade de Munique) reage preparando uma possível evacuação e a decisão é notificada às outras entidades através do UNICOM. Quando o míssil é destruído a mensagem é enviada pela IPHITOS através do UNICOM.

Figura 2.14 – Míssil no lançador, asteróide e Base de Controle.



Fonte: Falcone & Longo (2014).

Figura 2.15 – Visualização 3D da comunicação e explosão do asteróide.

Fonte: Falcone & Longo (2014).

Ao longo da publicação os autores apresentam detalhes técnicos sobre as ferramentas e metodologias utilizadas na implementação das soluções, tanto do modelo de visualização 3D em tempo real, quanto no modelo do UNICOM de alerta de asteróide.

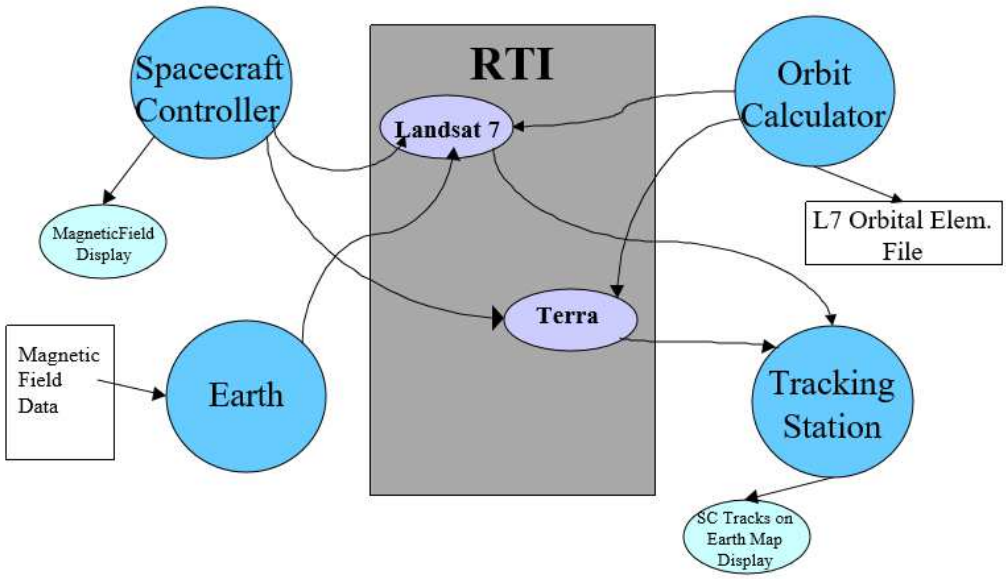
Ao final da publicação, os autores discorrem sobre os benefícios trazidos à comunidade universitária em decorrência da iniciativa da realização do evento SEE para estimular a utilização do HLA.

Outra publicação aplicando o cenário espacial é a de Reid et al (2000). Esta publicação apresenta uma análise da utilização do HLA com aplicação aos processos de modelagem e simulação da NASA. Nesta publicação, é apresentado um protótipo para simulações de missões espaciais, sendo esse protótipo desenvolvido de forma a cumprir com as regras do HLA bem como os estudos conduzidos para avaliar se a aplicação do HLA é viável à NASA. As figuras de mérito dos estudos conduzidos eram o reuso de componentes nas simulações, mitigação de riscos e redução de custo nos projetos e planejamento de missões espaciais.

A publicação se inicia apresentando os conceitos do HLA, o propósito desta arquitetura, as nomenclaturas e das regras. Também discorre sobre os benefícios que a utilização do HLA traria a NASA.

Em seguida, é apresentado o protótipo, considerado como um modelo que consistia de quatro pequenos simuladores, representando federados: um controlador de espaçonave, um calculador de órbita, um simulador da Terra e uma estação de rastreamento. Os simuladores foram rodados tanto simultaneamente quanto separadamente.

Figura 2.16 – Protótipo para análise do HLA como arquitetura.



Fonte: Reid (2000).

A RTI utilizada nas simulações foi a DMSO RTI, versão 1.3NG, descontinuada e atualmente não mais disponível para *download*.

Uma comparação interessante sobre o aproveitamento dos códigos é feita, mostrando que do total de códigos usados apenas 37% era novo, porém podendo chegar a um aproveitamento de aproximadamente 87% de código dos federados.

Ao final, o autor conclui que o HLA é viável como tecnologia para as simulações de missões espaciais, incluindo as que envolveriam múltiplas espaçonaves. Porém o autor levanta um ponto negativo, pois, apesar do HLA ter sido feito para tornar a integração de simuladores remotamente espalhados viável, esta arquitetura ainda não apresenta uma realidade totalmente *plug-and-play*, fazendo-se necessário, quando há múltiplos simuladores a se integrar, uma adaptação a

respeito da informação a ser trocada, para que se tornem compatíveis e a comunicação possível.

2.3. Principais Referências em Modelagem e Simulação de Satélites usadas neste Trabalho e publicadas pelo INPE

Prudêncio (1997) apresenta o projeto e a simulação em tempo real de sistemas de controle de atitude (SCA) de satélites utilizando um computador e o ambiente MATRIXx/SystemBuild. Este trabalho utiliza como aplicação o SCA do Satélite de Aplicações Científicas (SACI-1). Neste trabalho são usados como referência os modelos matemáticos do ambiente e da dinâmica.

Gobato (2006) apresenta os controles monovariáveis e multivariáveis aplicados a sistemas aeroespaciais fracamente ou fortemente acoplados. Gobato adapta parte dos modelos desenvolvidos por Prudêncio (1997) na simulação do SCA da PMM no seu Modo Nominal de Operação utilizando e comparando várias leis de controle monovariáveis e multivariáveis. Foi desenvolvido outro SCAO para este satélite, visto que a PMM possui estabilização nos três eixos, enquanto o SACI possui estabilização por rotação.

Moreira (2006) discute a análise, projeto e simulação de um controle discreto para a Plataforma MultiMissão e sua migração para um sistema operacional de tempo real. Este trabalho aplica parte dos modelos desenvolvidos por Gobato (2006) na simulação do SCAO da PMM no seu Modo Nominal de Operação utilizando um processador e um sistema operacional de tempo real.

Amorim Terceiro (2007) realiza a simulação paralela do SCAO da PMM no Modo Nominal de Operação utilizando dois processadores e um sistema operacional de tempo real. Este trabalho aplica parte dos modelos desenvolvidos por Gobato (2006) na simulação do SCAO da PMM no seu Modo Nominal de Operação.

Reis (2009) apresenta-se o estudo e o desenvolvimento de uma simulação distribuída em tempo real de um Sistema de Controle de Atitude e de Órbita (SCAO) para a Plataforma MultiMissão (PMM) em seu Modo Nominal de

Operação utilizando a arquitetura HLA (*High Level Architecture*). Reis desenvolveu o trabalho escolhendo um esquema de sincronização do mecanismo de troca de dados mais apropriados para a aplicação proposta. Em seguida, desenvolveu uma interface SystemBuild/HLA. Adaptou uma simulação *stand-alone* existente à arquitetura HLA. E realizou simulações usando um computador sem e com restrições de tempo real e depois em dois computadores sem e com restrições de tempo real.

Tagawa (2013) apresenta um estudo sobre a 1ª. geração da arquitetura aviônica chamada IMA (*Integrated Modular Avionics*), bastante em foco na atualidade para aplicações aeroespaciais e objeto de estudos de entidades tais como NASA e ESA (*European Space Agency*) e empresas como Boeing e Airbus. Para desenvolver o conceito de IMA, Tagawa modela, implementa e simula um Sistema de Controle de Atitude (SCA) compatível com o Modo Nominal da Plataforma MultiMissão (PMM) do INPE em um Simulador de IMA (SIMA). As simulações foram feitas em MATLAB, linguagem C e SIMA, e os valores são comparados e concordam entre si. Tagawa dividiu o modelo do SCA em seis partições e, inseriu falhas em uma das partições. Apesar da falha desta partição, as demais partições do SCA continuaram seu funcionamento normalmente demonstrando a viabilidade no uso da plataforma IMA, mostrando-se robusto.

Moraes (2017) estendeu o trabalho de Tagawa (2013) à 2ª geração da arquitetura IMA, chamada de DIMA (*Distributed* IMA), instalando-a e testando-a nas mesmas condições que Tagawa (2013), mas agora num computador DIMA que ele mesmo construiu, com dois processadores *Beagle-Bone*, um roteador, e o sistema operacional AIR da GMV.

2.4. Referências em Modelagem e Simulação de Satélites sobre Simulação Distribuída HLA publicadas pelo INPE

Trivelato (2003) é uma nota técnica que apresenta uma introdução sobre simulação distribuída com ênfase na arquitetura *High Level Architecture* (HLA) e em simulação na *Web* (*Web Simulation*). São apresentados resumidamente: conceitos básicos de simulação distribuída; um histórico sobre o esforço do *Department of Defense* (DoD) dos EUA na área de simulação distribuída; a descrição da arquitetura HLA, incluindo *Framework and Rules*, *Federation Interface Specification* e HLA *Object Model Template* (OMT); conceitos sobre simulação na *Web* incluindo simulação como hipermídia, metodologias de pesquisa em simulação, programas de acesso à simulação baseada na *Web*, modelagem e simulação distribuída, e simulação da www; a conexão entre HLA e Simulação na www; e os desenvolvimentos desejados nos próximos anos para Simulação na www nas áreas de: educação e treinamento, aplicações militares, saúde pública, transportes, construção e fabricação, distribuição e logística, e indústria aeroespacial.

Romeiro e Souza (2016) é uma publicação apresentada no congresso SAE Brasil de 2016. Esta publicação apresenta estudos de caso de simulação distribuída utilizando a ferramenta VR-FORCES da VT- MÄK, que foi desenvolvida tanto para o protocolo DIS quanto para a arquitetura HLA. Inicia-se com os conceitos, revisão bibliográfica e ferramentas. Em seguida, é apresentado o ambiente utilizado para as simulações junto com as primeiras experiências e lições aprendidas com as primeiras instalações e execuções do ambiente. São apresentados os cenários das simulações e, por fim, a conclusão.

3 A PLATAFORMA MULTIMISSÃO

Em 2001 O INPE iniciou o programa espacial chamado PMM – Plataforma MultiMissão. Essa plataforma corresponde a um módulo de serviço básico para satélites de tamanho médio. Esse programa foi o primeiro que apresentou o desafio de se desenvolver o sistema de propulsão, o qual ficou sob a responsabilidade da empresa Fibraforte TM E.I.C.Ltda (Zandonadi, 2013).

Sendo assim, a proposta da PMM é de ser genérica e flexível e aplicável a diferentes missões espaciais, fazendo a composição de diversos satélites desenvolvidos pelo INPE, tais como os da série Amazônia. Sendo o primeiro satélite a usar o módulo de serviço, o satélite Amazônia-1. Esse módulo básico de serviço pode ser utilizado por satélites que tenham como requisitos em comum a estabilização em três eixos, a mesma massa (na classe de 500 kg), órbita, atitude, dissipação térmica, mesmas características estruturais e computacionais (um computador) (Reis, 2006). Dessa forma, os diferentes satélites que fazem uso da PMM, irão diferir apenas na carga útil.

Os conceitos sobre módulo de serviço e carga útil de um satélite, bem como a composição da PMM e do Amazônia-1 estão apresentados nos subcapítulos a seguir.

O conceito modular da PMM permite que o desenvolvimento e integração do módulo de serviço do satélite seja realizado separadamente do desenvolvimento de uma carga útil especificamente para uma missão. Isso permite projeto, construção e testes independentes e antes da integração da carga útil à PMM (Silva, 2014).

Para o processo de desenvolvimento e qualificação de satélites, além dos testes de integração, uma série de testes e análises ambientais são feitos para confirmar que o satélite suportará as condições ambientais às quais será submetido. A qualificação ambiental consiste de testes de compatibilidade e interferência eletromagnética (EMC/EMI), análises de dissipação térmica, testes de acústica e

vibração e análises de materiais. Já os testes de integração envolvem a verificação de especificações através de ensaios e de simulações de sistemas e subsistemas, integrados e interagindo. Os testes de montagem verificam se após o processo de fabricação e junção das peças do satélite os sistemas funcionam adequadamente (Silva, 2014).

Essas atividades além das demais que fazem parte do conjunto de atividades de Montagem, Integração e Testes (AIT - *Assembly, Integration and Tests*) foram conduzidas pelo Laboratório de Integração e Testes (LIT) do INPE (Zandonadi, 2013).

Neste trabalho, simulamos as equações da Plataforma MultiMissão, selecionando o Satélite Amazônia-1 como exemplo e utilizando o MATLAB e *dlls* compiladas a partir do C++.

3.1. Composição Básica de um Satélite

Um satélite pode ser decomposto em duas partes, uma delas constituindo o Módulo de Serviço ou Plataforma e a outra o Módulo de Carga Útil.

Fazem parte do Módulo de Serviço todos os componentes necessários para determinar e manter as condições operacionais do satélite.

O Módulo de Serviço é composto de:

a) Controle de Atitude e de Órbita (*Attitude Determination and Control; ou Attitude and Orbit Control System* – AOCS) – cuja a função é garantir o apontamento de um ou mais eixos do satélite para um determinado alvo (Terra, Sol, etc.), de acordo com a particular missão. Controlar os eixos do satélite garantindo um erro máximo e a estabilidade do satélite, de acordo com as exigências da missão.

b) Suprimento de Energia Elétrica e Distribuição – esse módulo deve gerar energia elétrica continuamente, armazenando-a quando necessário, regular e converter as tensões requeridas pelos diversos componentes do satélite.

c) Telecomunicação de Serviço (*Telemetry, Tracking and Command* (TT&C)) – deve enviar dados de telemetria (estado dos diversos equipamentos de bordo) para o solo e receber do solo telecomandos que alteram a configuração. Permite fazer medidas de distância (*ranging*) e de velocidade (*range-rate*/doppler) entre o satélite e a estação de rastreio e controle. Este subsistema é responsável pela comunicação, com o recebimento e transmissão dos sinais.

d) Gestão de Bordo (comando e tratamento de dados) – a função desse sistema é coletar as informações de estado dos diversos componentes de bordo e enviá-las como Telemetria para o solo através do subsistema de TT&C. Receber os Telecomandos oriundos do solo através do subsistema de TT&C, executá-las diretamente ou distribuí-los pelos diversos componentes de bordo em tempo real ou diferenciado. E gerir os diversos modos de operação do satélite.

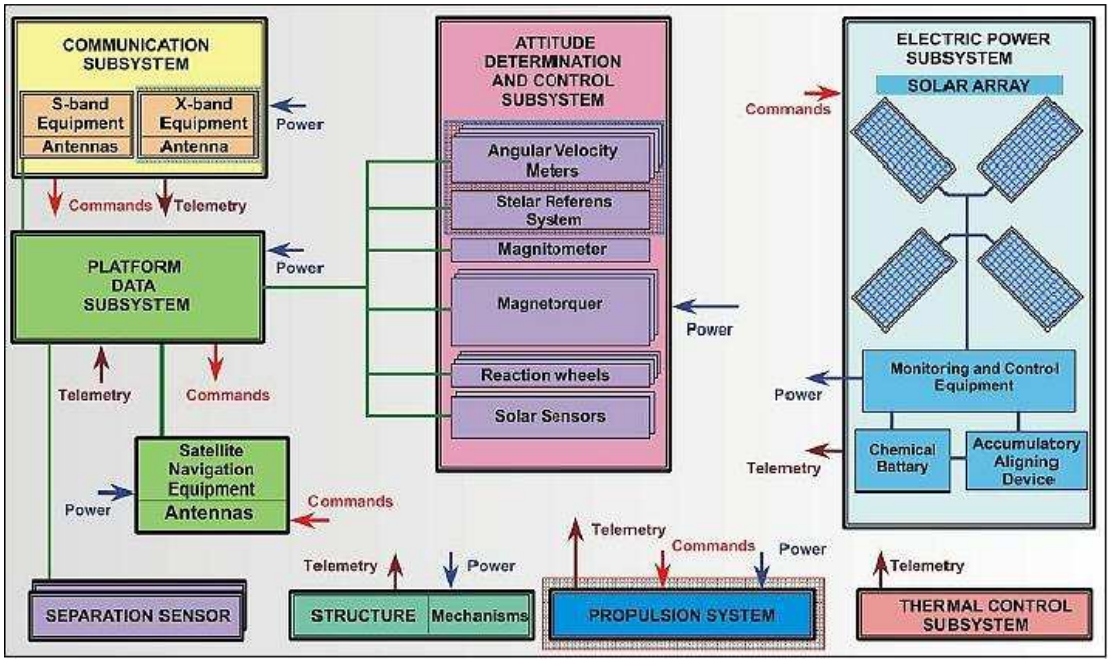
e) Estrutura e Mecanismos – deve fornecer o suporte mecânico e de movimento para as partes do satélite. Também oferece proteção contra as vibrações de lançamento e evita acoplamentos com o veículo lançador.

f) Controle Térmico – deve manter os equipamentos dentro de suas faixas nominais de temperatura, dentro dos limites de estabilidade e gradientes.

g) Propulsão – deve fornece o empuxo necessário para o controle de órbita e produzir os torques necessários para o controle de atitude.

A Figura 3.1 mostra os sistemas básicos do Módulo de Serviço de um satélite.

Figura 3.1 – Sistemas e subsistemas componentes do Módulo de Serviço de um satélite.



Fonte: https://eoportal.org/web/eoportal/satellite-missions/content/-/article/sich2.

A carga útil corresponde aos equipamentos específicos para executar a particular missão para o qual o satélite foi desenvolvido.

Logo, a composição da carga útil de um satélite dependerá da missão do mesmo. Por exemplo, um satélite de sensoriamento remoto apresentará em sua carga os componentes de gravação, rastreamento e comunicação, tais como:

a) Câmeras (ópticas e eletrônicas): o sistema óptico de focar, estabelecer o campo de visada e as bandas do espectro. O sistema eletrônico irá estabelecer a resolução, amostragem, amplificação e faixas de operação e fornecer dados relativos às imagens tais como localização e atitude, além de gerar o sinal de vídeo a ser transmitido incluindo todas as bandas e dados auxiliares.

b) Transmissores de Banda: devem modular o sinal de vídeo com a frequência da portadora, amplificando-o antes de ser transmitido pela antena.

c) Gravadores: devem gravar as imagens quando estiver fora da visibilidade da estação de solo e reproduzi-las quando estiver sob visibilidade da estação.

d) Antenas: devem transmitir a portadora na banda específica de modulação e apresentando um diagrama de radiação específico e adequado à missão.

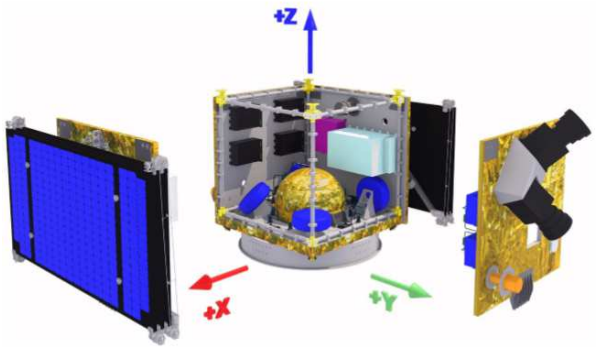
Mais a seguir, são apresentadas a composição do módulo de serviço da PMM e mais adiante uma breve descrição da composição da carga útil da PMM.

3.2. Composição do módulo de serviço da PMM

A PMM, sendo um módulo de serviço, apresenta os equipamentos mínimos para a sobrevivência do satélite independentemente da órbita, missão ou apontamento.

A PMM foi dimensionada de forma otimizada tal que apresenta o formato externo de um cubo de dimensões 1 m × 1 m × 1 m. Dentre deste cubo, estão sistemas organizados de forma a prover ao satélite recursos de potência elétrica, controle, telemetria, telecomando e transmissão de dados. A Figura 3.2 apresenta a forma física da PMM, estando os componentes dos diversos subsistemas deste módulo de serviço agrupados neste cubo.

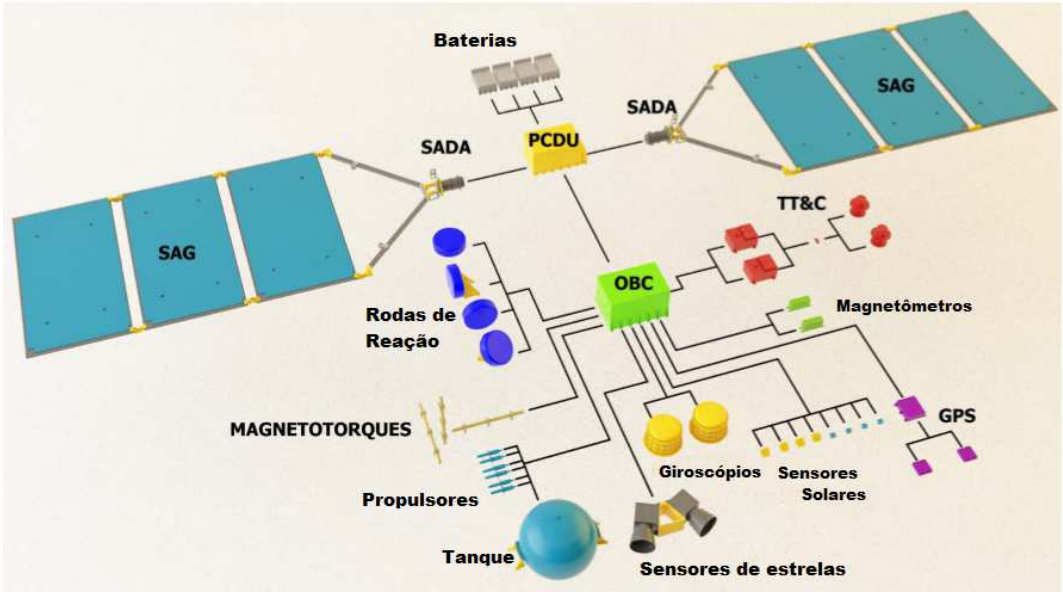
Figura 3.2 – Formato Geométrico da PMM.



Fonte: Zandonadi (2013).

A Figura 3.3 mostra, resumidamente, a interação entre os sistemas da PMM. Pode ser visto neste diagrama que os subsistemas interagem com o computador de bordo, chamado de Sistema de Controle a Bordo (OBCS - *On Board Control System*) (Oliva, 2012).

Figura 3.3 – Diagrama Funcional da PMM Simplificado.



Fonte: Oliva (2012).

3.2.1. Localização dos Componentes da PMM

A Figura 3.4 apresenta uma vista explodida da PMM, permitindo uma visualização mais fácil da localização dos componentes.

Dessa forma, observa-se que no painel lateral correspondente à posição –X, estão a maioria dos componentes do sistema de geração e condicionamento da potência elétrica. Estando nesse painel a PCDU (*Power Conditioning and Distribution Unit*), Unidade para Condicionamento e Distribuição de Energia (elétrica) e um dispositivo chamado de SADA (*Solar Array Drive Assembly*), responsável pelo painel solar localizado neste lado; e mais receptores de GPS. O SADA é o equipamento responsável por mover os painéis solares de forma a obter sempre o máximo de energia solar. O outro SADA está localizado no lado +X para mover o outro painel solar, além da Eletrônica de Controle de Atitude ACE (*Attitude Control Electronics*), o sistema de Gestão de Bordo OBDH (*On Board Data Handling*), a Eletrônica de Controle dos Propulsores TCE (*Trusters Control Eletronics*) e um magnetotorque.

No painel da face correspondente ao eixo +Y, estão *transponders,* giroscópios, o Amplificador de Baixo Ruído LNA (*Low Noise Amplifier*), antenas de GPS, sensores de estrelas e uma antena de Banda-S.

No painel da face –Y estão quatro conjuntos de baterias, uma antena de banda S (2 a 4 GHz) e duas barras de magnetotorque.

O painel superior da PMM (+Z) contém dois magnetômetros orientados com a estrutura; o acoplamento da carga útil é feito através desta face.

O tanque de propelente e de pressurização é uma esfera localizada no centro do cubo. O tanque é conectado aos propulsores através das linhas de dutos e de válvulas de controle.

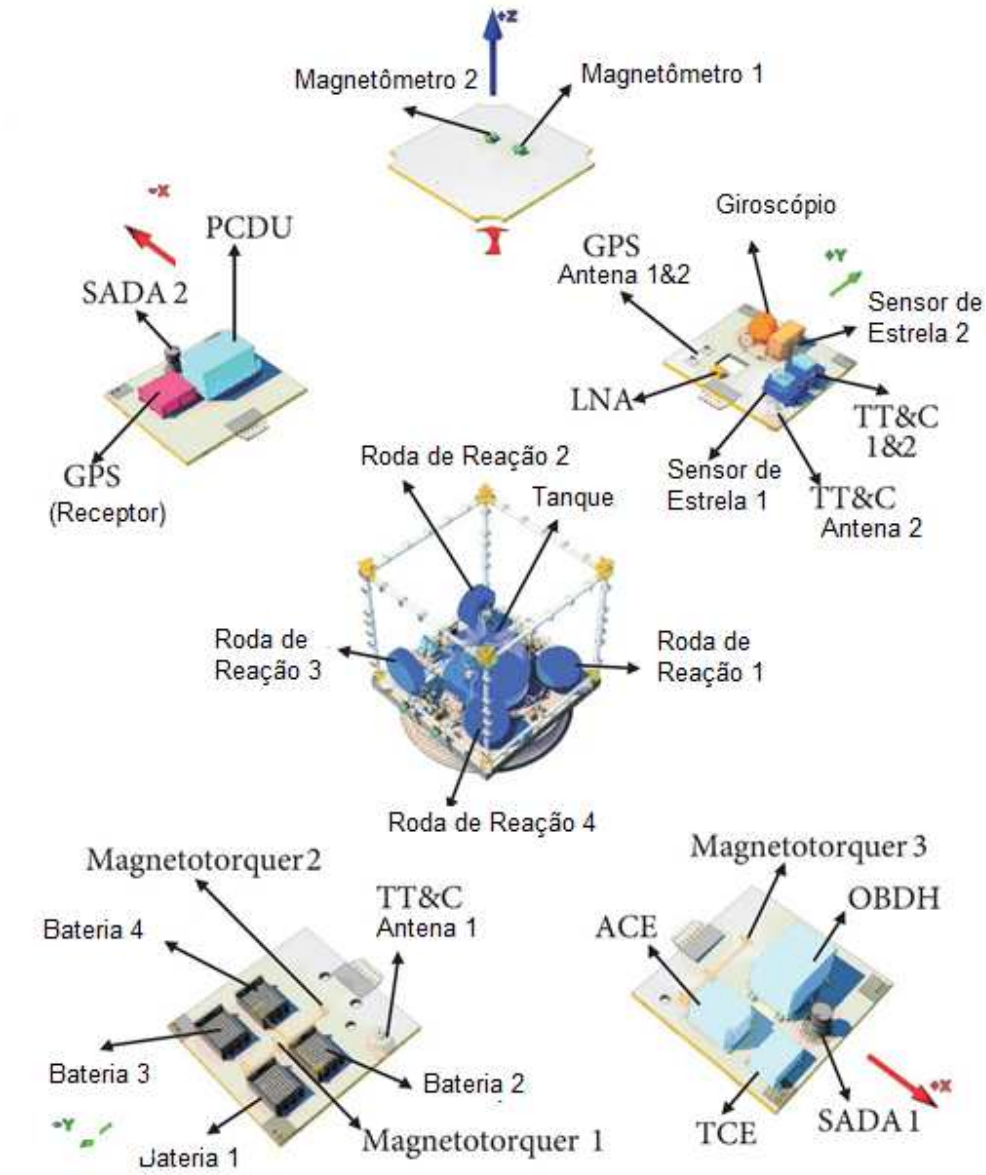
Seis propulsores do tipo 5-N estão apontando na direção –Z do eixo Z. Quatro rodas de reação estão localizadas nas esquinas do painel inferior, também na posição -Z.

A parte inferior da face –Z apresenta o anel para adaptação do satélite ao veículo lançador.

E, em cada vértice do cubo, há colunas de alumínio que, em conjunto com os painéis, formam a estrutura da PMM.

A PMM pesa cerca de 250 kg, podendo aceitar uma carga útil de até 280 kg.

Figura 3.4 – Vista Explodida da PMM.



Fonte: Silva (2014).

3.2.2. O Sistema de Controle de Atitude e Órbita da PMM

A vida útil de um satélite é dependente da sua capacidade de manter a órbita e a atitude (orientação) correta (Oliva, 2012). Dessa forma, o sistema de controle de atitude e órbita desempenha um papel importante na missão do satélite e deve ser desenvolvido atendendo requisitos específicos de confiabilidade e acurácia dos sensores, de forma a prolongar a vida do satélite e o cumprimento da missão.

A PMM provê controle de atitude e órbita num modo estabilizado em três eixos, permitindo o apontamento para a Terra, apontamento inercial e solar. Essas funções são feitas pelo Subsistema de Controle de Atitude e de Gestão de Bordo ACDH (*Attitude Control and Data Handling*).

O subsistema ACDH é dividido em dois subsistemas:

- SCAO – Subsistema de Controle de Atitude e Órbita (*Attitude and Orbit Control*);
- OBDH – Subsistema de Gestão de Bordo (*On-Board Data Handling*).

O SCAO (Subsistema de Controle de Atitude e Órbita) da PMM é responsável pelo controle de atitude e órbita do satélite durante todas as fases da missão. Para isso, este subsistema desempenha as seguintes tarefas:

a) Adiquirir a atitude do satélite logo após a fase de lançamento;

b) Controlar a posição dos painéis solares;

c) Controlar a atitude e órbita do satélite;

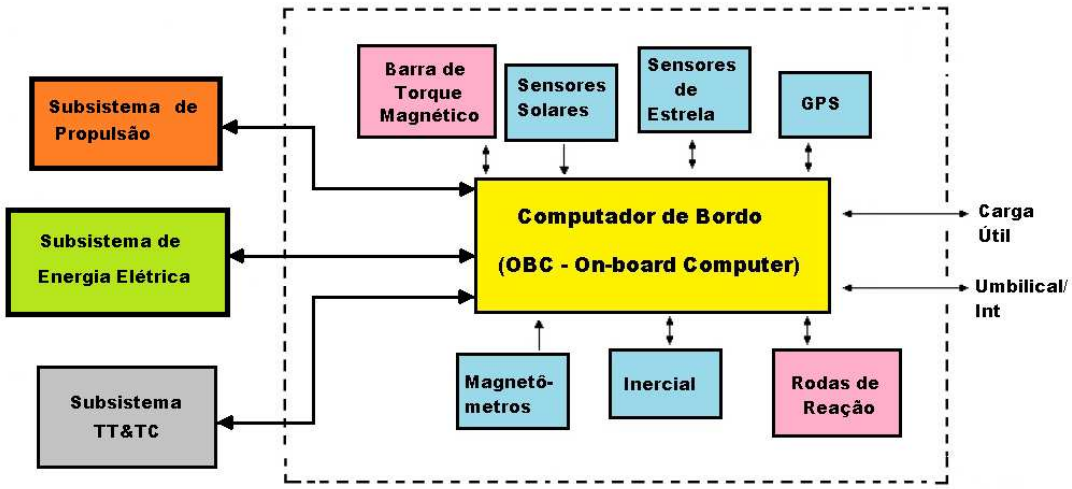
d) Sincronizar o tempo;

e) Gerenciar o empuxo dos propulsores, incluindo: controle de chama, de chama cumulativa e estimativa do combustível disponível;

f) Controlar os modos de transição.

A Figura 3.5 apresenta um diagrama em blocos dos componentes do SCAO da PMM.

Figura 3.5 – Diagrama Simplificado da Composição Básica do SCAO da PMM.



Fonte: INPE (2001).

O controle e estabilização da atitude é feito a partir de três rodas de reação alinhadas com os eixos de rolamento, arfagem, e guinada, respectivamente. A quarta roda de reação está alinhada com um eixo de *offset*, como redundância não ativa.

As rodas de reação corrigem seus momentos angulares usando um conjunto de três barras de torque magnético, também alinhadas com os eixos do satélite e quando necessário, serão utilizados os motores propulsores (subsistema de propulsão).

Os sensores de atitude da PMM são os seguintes:

a) Dois sensores de estrela, usados em configuração de redundância;

b) Dois magnetômetros de 3-eixos;

c) Uma unidade inercial (giroscópica de 4 eixos), numa configuração tetraédrica com redundância interna; e

d) Oito sensores solares.

Como mostrado na Figura 3.5, o ACDH deve gerenciar os equipamentos de controle de atitude e órbita e fazer interface com os outros sistemas da PMM.

O computador de bordo é composto de *software*, hardware e partes mecânicas, sendo o responsável pelo gerenciamento dos sensores, rodas de reação, sistema de propulsão e TT&C (telemetria, rastreamento e comando).

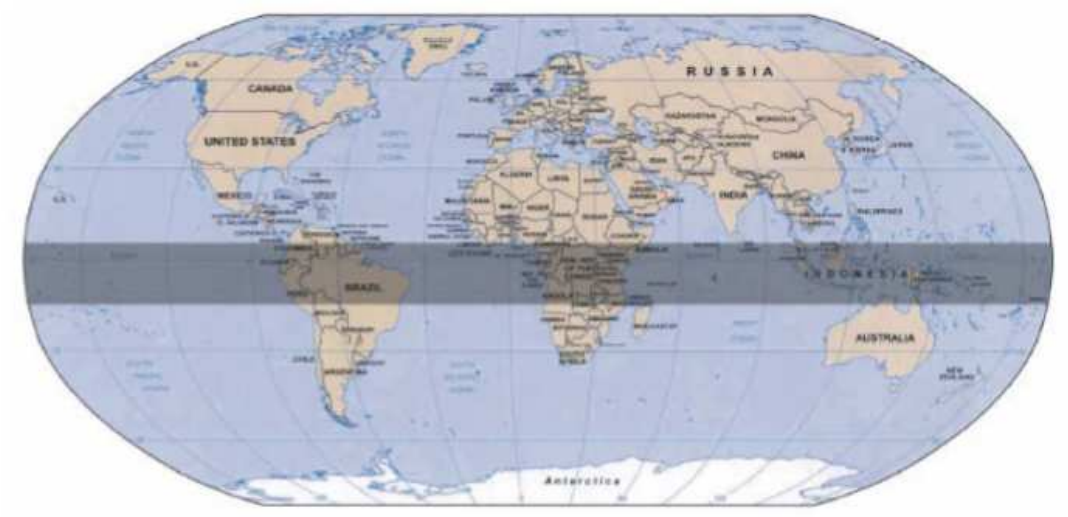
3.3. O Satélite Amazônia-1

O Satélite escolhido neste trabalho é um satélite brasileiro de sensoriamento remoto (RSS - *Remote Sensing Satellite*), o qual utiliza a plataforma PMM, chamado de Amazônia-1.

A principal missão do Amazônia-1 é prover dados de imagem de forma a melhorar o controle do desmatamento da região amazônica.

O Amazônia-1 deve cobrir uma área de 5° ao Norte a 15° ao Sul no território brasileiro (Figura 3.6), com uma resolução da ordem de 40 metros.

Figura 3.6 – Área de Cobertura do Amazônia-1.



Fonte: Reis (2006).

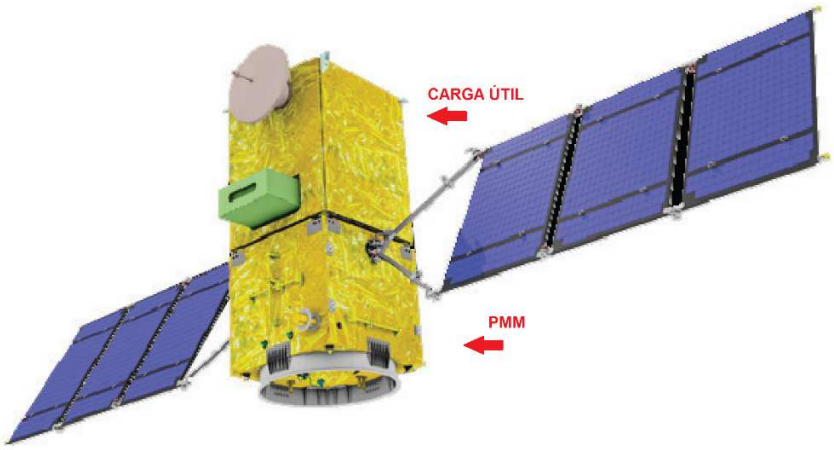
Este satélite deve capturar imagens durante várias vezes ao dia, de forma a se ter melhores dados, eliminando e sobrepondo as amostras que são degradadas devido à cobertura de nuvens.

Os principais usuários deste satélite são: o INPE, agências e órgãos governamentais voltados à proteção ambiental e gestão de recursos naturais, e empresas de processamento de imagens (PNAE, 2012).

As empresas participantes do projeto do Amazônia-1 foram: Atech, Cenic, Fibraforte, Mectron, Omnisys e Opto Eletrônica (PNAE, 2012).

No que concerne à carga útil deste satélite a mesma é composta de sistemas ópticos operando no espectro visível, atendendo à resolução da ordem de 40 metros.

Figura 3.7 – Satélite Amazônia-1.



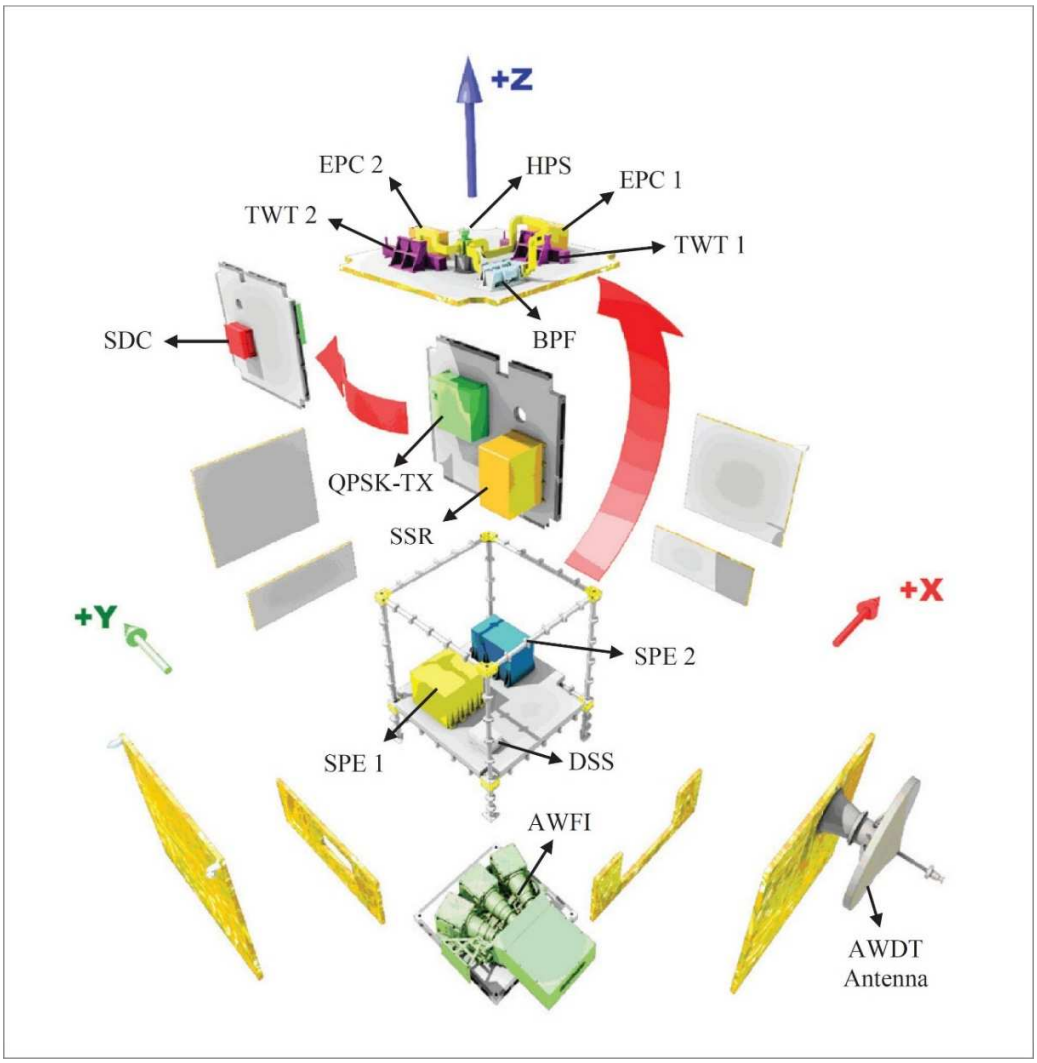
Fonte: Adaptado de Silva (2014).

Figura 3.8 – Satélite Amazônia-1 na configuração de Lançamento.

Fonte: Zandonadi (2013).

O Sistema deste satélite utiliza um instrumento chamado de AWFI (*Advanced Wide Field Imager*) o qual apresenta uma resolução espacial de 40 metros, sendo um diferencial em relação aos atuais chamados de WFI (*Wide Field Imager*).

Figura 3.9 – Vista Explodida da Carga Útil do Amazônia-1.



Fonte: Silva (2014).

De forma a atender à missão, a face –Y do satélite estará sempre apontada para a Terra.

3.3.1. Características do Satélite Amazônia-1

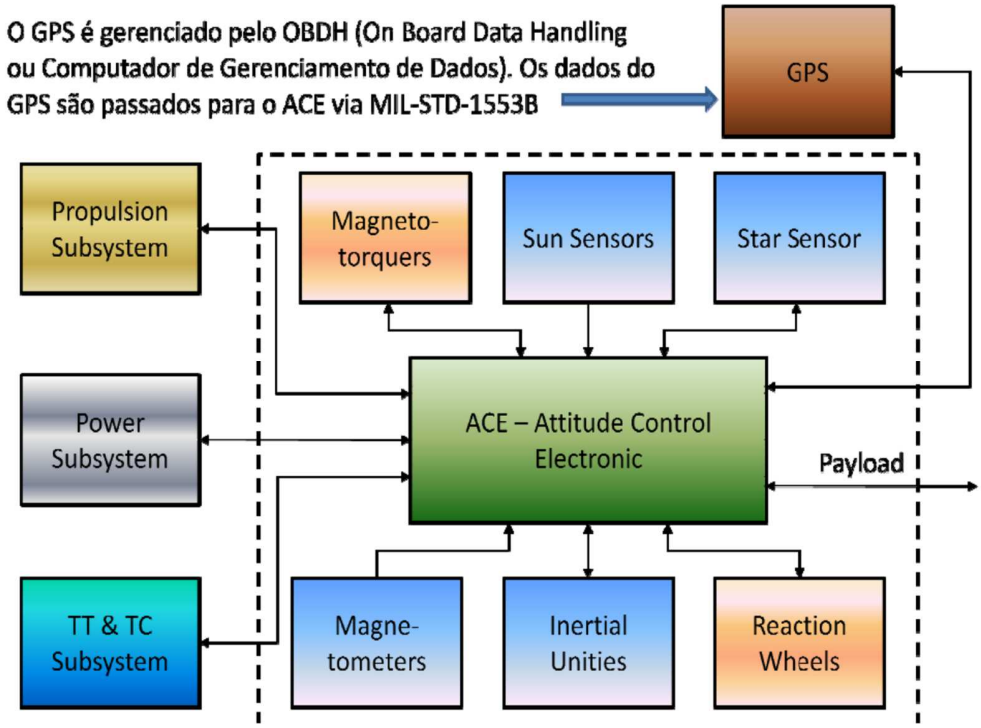
As principais características do satélite Amazonia-1 são:

- Dimensões do corpo principal: 2,200 x 950 x 950 mm;
- Massa: cerca de 500.0 kg (carga útil + PMM);
- Estrutura: Alumínio (em colméia);
- Potência: 420 W (média), painéis solares InGAP/InGaAs/Ge, e baterias de íon *Lithium*;
- Estabilização: em 3 eixos;
- Duração da Missão: 4 anos.

3.4. O SCAO do Satélite Amazônia-1

Consistente com a missão de monitoramento, o Amazônia-1 deverá produzir imagens com maior frequência e maior definição, adequadas para monitorar o ambiente e gerenciar recursos naturais. O satélite será estabilizado em três eixos com o apontamento de uma câmera para a Terra. O Sistema de Controle de Atitude e Órbita, cujo Modo de Operação Nominal é objeto do caso de estudo, conterá atuadores e sensores que permitirão satisfazer os requisitos da missão (Figura 3.10)

Figura 3.10 – Subsistema de Controle de Atitude e Órbita do Satélite Amazônia-1.



Fonte: Tagawa (2013).

A Figura 3.11 ilustra o SCAO do satélite Amazônia-1 mostrando o H/W e a interface com três outros subsistemas, além da interface com o OBDH.

O subsistema é composto de sensores (magnetômetros, unidades inerciais, sensores solares, sensores de estrelas), atuadores (bobinas magnéticas, rodas de reação) e o ACE (Tagawa, 2013).

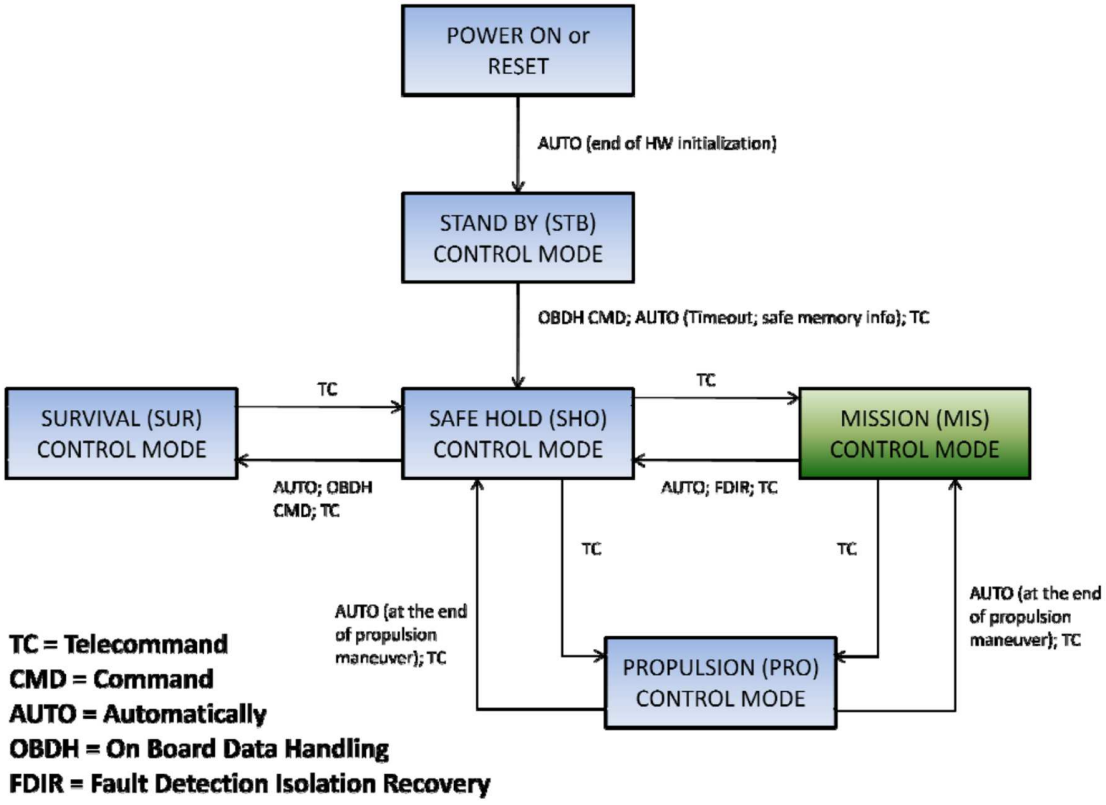
O SCAO deve prover as seguintes funções de controle de atitude e órbita (Tagawa, 2013):

- Controle de atitude estabilizado em três eixos no Modo Nominal, permitindo apontamento para Terra;
- Determinação de atitude a bordo;
- Aquisição e manutenção segura de atitude após a fase de lançamento ou ocorrência de falha;

- Controle do posicionamento/velocidade dos painéis solares;
- Determinação da posição e velocidade do satélite;
- Controle dos propulsores para aquisição e manutenção de órbita;
- Dessaturação das rodas de reação.

Os Modos de Operação do SCAO do Amazônia-1 são ilustrados na Figura 3.11.

Figura 3.11 - Modos de controle do SCAO do satélite Amazônia-1.



Fonte: Tagawa (2013).

No Modo Nominal, i.e., Modo de Controle da Missão (*Mission Control Mode*), o satélite será configurado de forma que sua carga útil cumpra sua missão. Nessa situação, a atitude do satélite é mantida em direção ao alvo. Neste modo, a atitude do satélite bem como a taxa de variação deve ser controlada nos três eixos para cumprir com os seguintes requisitos (Tagawa, 2013):

:

1. Precisão de apontamento: < 0.05º (3σ).
2. Estabilidade:
    a. Deriva (*Drift*) < 0.001º/s com frequências até 2Hz;
    b. Variação pico a pico < 0.005º com frequências de 2Hz a 10Hz.
3. *Jitter* < 0.0005º (1σ) com frequências acima de 10Hz até 100Hz.
4. Determinação de atitude: ≤ 0.005º (3σ).
5. Desvio (*off pointing*) de até 30º em torno do eixo de rolamento em 180 segundos.

4 O MODELO MATEMÁTICO DOS MOVIMENTOS DE UM SATÉLITE E DE SEU CONTROLE DE ATITUDE

Este capítulo apresenta o desenvolvimento das equações utilizadas, bem como alguns conceitos necessários para o desenvolvimento do modelo a ser implementado.

4.1. Sistemas de Referência

Para o entendimento das equações do satélite utilizadas neste trabalho, se faz necessário um conhecimento sobre os sistemas de referência utilizados. Dessa forma, serão apresentados a seguir os referenciais aqui utilizados.

De acordo com Wertz (1978), para a mecânica orbital, os sistemas de referência mais importantes são:

- Referencial Inercial (Fixo no Espaço);
- Referencial Orbital (definido em relação à órbita);
- Referencial do Satélite (definido em relação ao corpo);

4.1.1. Referencial Inercial (RI)

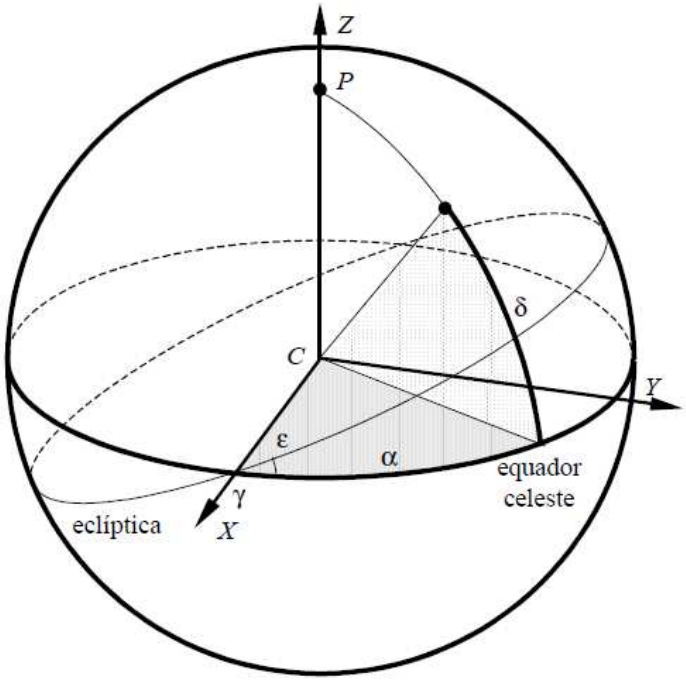
Neste trabalho, da mesma forma que Prudêncio (1997), Amorim Terceiro (2007) e Gobato (2006), o sistema de coordenadas inercial adotado é o sistema de coordenadas Celestial definido relativo ao eixo de rotação da Terra. Esse sistema é caracterizado por três eixos definidos como:

- Eixo Z: É definido do centro da Terra ao Pólo Norte, estando a direção do Pólo Norte desse sistema a aproximadamente 1° da Estrela Polar.
- Eixo X: O ponto no Equador terrestre escolhido como referência é o ponto da Eclíptica, ou plano da órbita do Sol ao redor da Terra, que cruza o Equador indo do Sul para o norte, conhecido como Equinócio Vernal. Essa

é a direção da linha do centro da Terra para o Sol no primeiro dia de primavera.

- Eixo Y: É encontrado usando a regra da mão direita, completando o sistema dextrogiro.

Figura 4.1 – Sistema cartesiano celeste (γ – Equinócio Vernal).

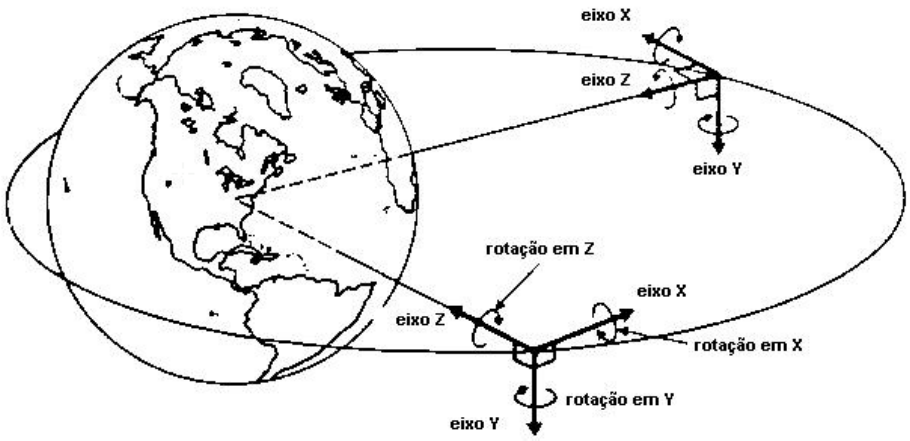


Fonte: Kuga et al (2012).

4.1.2. Referencial Orbital (RO)

O Sistema Referencial Orbital neste trabalho se refere ao sistema vertical local horizontal local (VLHL), (xo, yo, zo), mostrado na Figura 2.2. Nesse sistema o eixo zo é direcionado para o nadir (para o centro da Terra), o eixo yo é direcionado para o negativo da normal a orbita, e o eixo de xo é perpendicular aos outros dois, formando o sistema dextrogiro. O eixo xo coincide com a direção do vetor velocidade orbital para o caso de uma órbita circular (WERTZ, 1978).

Figura 4.2 – Sistema de referência Vertical Local Horizontal Local.



Fonte: Adaptada de Wertz (1978).

4.1.3. Referencial do Satélite ou do Corpo (RS)

O referencial do satélite é um sistema de coordenadas com origem no centro de massa do satélite e são coincidentes com os eixos principais de inércia do mesmo.

Neste estudo, utilizaremos um satélite estabilizado em três eixos, Terra apontado, sendo os três eixos definidos como:

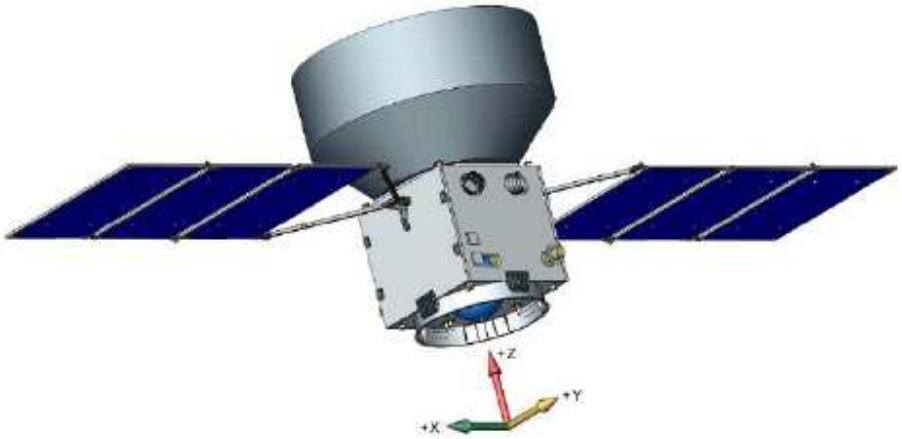
- Eixo de rolamento (*roll*), nominalmente alinhado com o x;
- Eixo de arfagem (*pitch*), nominalmente alinhado com o y;
- Eixo de guinada (*yaw*), nominalmente alinhado com o z.

Da mesma forma, são definidos os ângulos:

- φ de rolamento, ou seja, de rotação em torno do eixo x.
- θ de arfagem, ou seja, de rotação em torno do eixo y.

- ψ de guinada, ou seja, de rotação em torno do eixo z.

Figura 4.3 – Referencial do Satélite.



Fonte: Gobato (2006).

4.2. Definição da Órbita de um Satélite

4.2.1. Elementos Orbitais

Os elementos keplerianos ou clássicos constituem coordenadas que posicionam completamente o satélite e sua órbita (Kuga et al, 2012), os principais são:

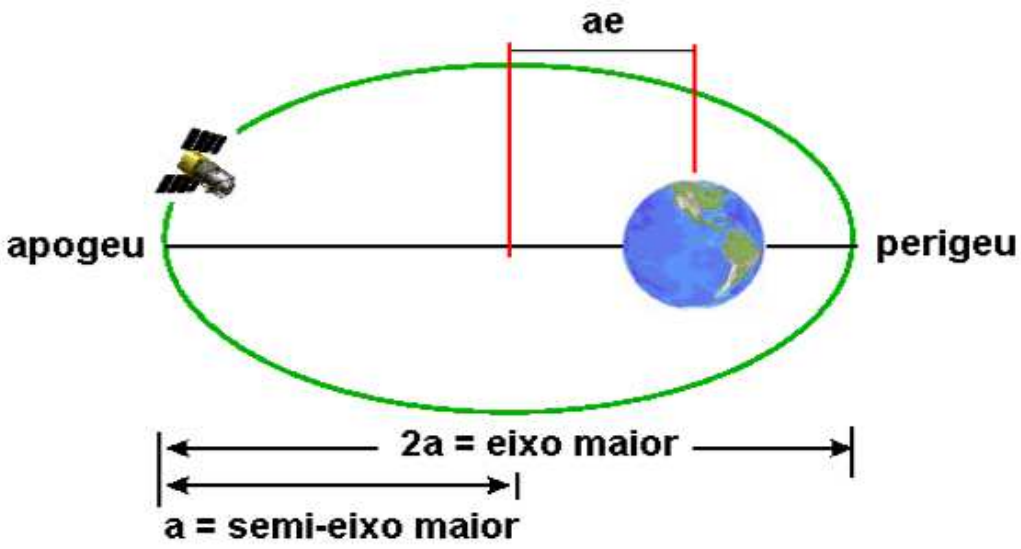
- Semi-eixo maior (a);
- Excentricidade (e);
- Anomalia média (M);
- Inclinação (i);
- Nodo ascendente (Ω);
- Argumento do perigeu (ω).

Os seis principais elementos estão definidos a seguir.

Semi-eixo maior (a) – define o tamanho da órbita e o período de revolução desta, fornecendo a dimensão da elipse.

Excentricidade (e) – determina o alongamento da órbita e descreve a forma da elipse.

Figura 4.4 – Semi-eixo Maior e Excentricidade.



Fonte: Bogossian (2015).

Os valores dá execntricidade definiram o formato da órbita conforme abaixo:

a) Valor de e=0, a órbita é um círculo;

b) Valor de e<1, a órbita é uma elipse;

c) Valor de e=1, a órbita é uma parábola;

d) Valor de e>1, a órbita é uma hibérbole;

Inclinação (i) – é o ângulo entre o plano do equador e o plano da órbita. Também definido como o ângulo entre o vetor momento angular da órbita e o vetor Z do sistema de coordenadas fixo à Terra.

A órbita pode ser classificada de acordo com o valor de i, como a seguir:

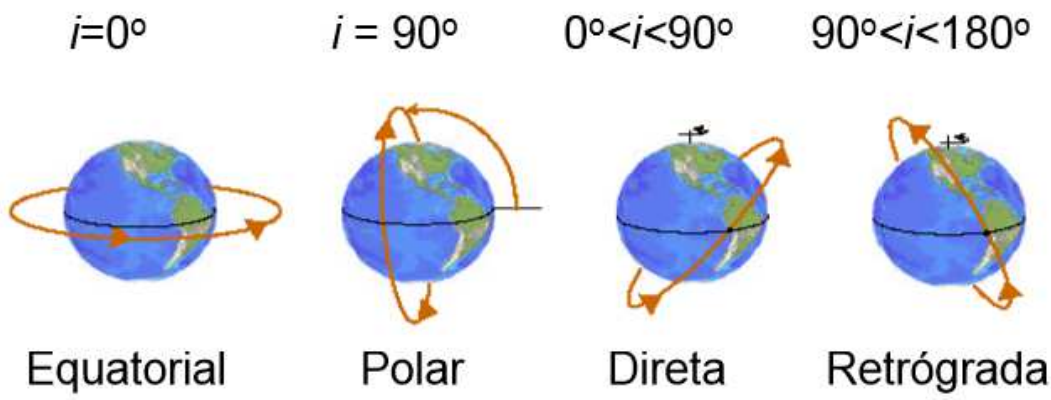
$0^o \leq i \leq 90^o$, órbita direta;

$90^o < i \leq 180^o$, órbita retrógrada;

$i = 0^o$; órbita equatorial;

$i = 90^o$; órbita polar.

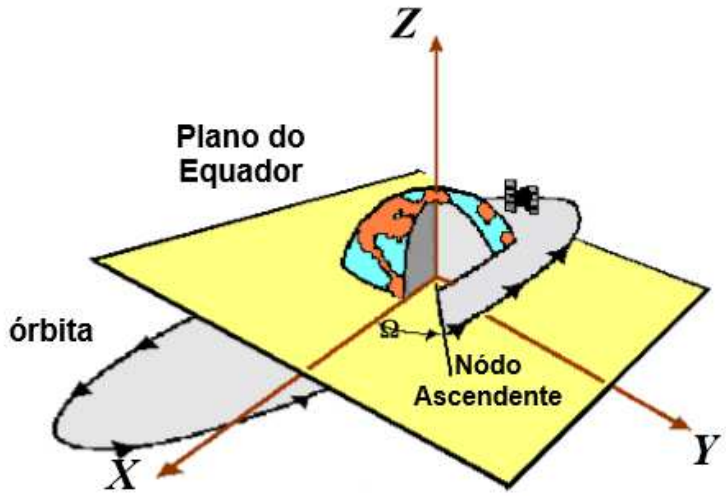
Figura 4.5 – Inclinação de órbita.



Fonte: Bogossian (2015).

Nodo ascendente (Ω) é o ângulo do equinócio vernal ao ponto onde a órbita intercepta o plano do Equador, indo do hemisfério sul para o norte. Os valores de Ω são: $0^o < \Omega \leq 360^o$.

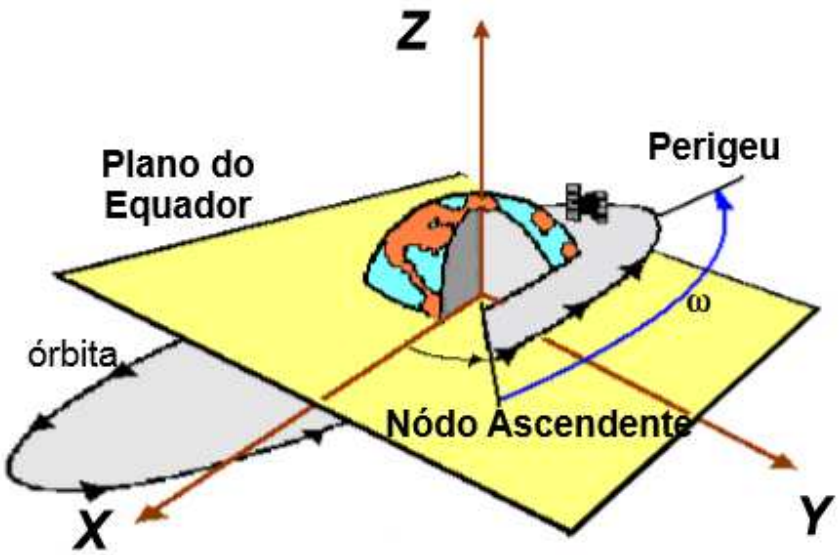
Figura 4.6 – Nodo ascendente (Ω).



Fonte: Bogossian (2015).

Argumento do perigeu (ω) é o ângulo entre as direções do nodo ascendente e do perigeu, medidos ao longo do plano da órbita. O perigeu (Π) é o ponto mais próximo entre o satélite e o centro de massa da Terra. O ponto mais distante é o apogeu.

Figura 4.7 – Argumento de Perigeu (ω).



Fonte: Bogossian (2015).

4.2.2. Equações Simplificadas obtidas de Elementos Orbitais, Constantes e Definições Aplicáveis.

Outros parâmetros da órbita podem ser obtidos a partir dos elementos de Kepler. Essas equações podem ser generalizadas para qualquer tipo de órbita e também podem ser simplificados quando utilizados para a órbita circular.

a) Raio de Apogeu [km]:

$$r_a = a(1+e); \tag{4.1}$$

b) Raio de Perigeu [km]:

$$r_p = a(1-e); \tag{4.2}$$

c) Excentricidade:

$$e = \left(\frac{r_a}{a}\right) - 1 = 1 - \left(\frac{r_p}{a}\right); \quad (4.3)$$

d) Semi-eixo maior [km]:

$$a = \frac{\left(r_a + r_p\right)}{2}; \quad (4.4)$$

e) u é a anomalia excentrica, que pode ser calculada da Equação de Kepler que a relaciona com a anomalia média:

$$M = u - e.sen\left(u\right); \quad (4.5)$$

Essa equação é uma equação transcendental, podendo ser resolvida através do método de Newton-Raphson.

f) n é a velocidade angular média, também chamada de movimento médio:

$$n = \omega = \sqrt{\left(\frac{\mu}{a^3}\right)} \text{ [rad/s] para a [km]}; \quad (4.6)$$

Algumas definições e constantes relacionadas ao movimento em órbita:

g) r corresponde ao vetor posição do satélite relativo ao centro de massa da Terra, seu módulo é definido por:

$$r = R_t + h \text{ [km]}; \quad (4.7)$$

h) h [km] é a altitude geodética do satélite, ou seja, a distânica em relação a superfície da Terra;

i) R_t é o raio da Terra, que para simplisficações nesse trabalho é considerado constante e com valor de 6378 km;

j) $\mu = G.M_{Terra}$ é a constante geo-gravitacional e apresenta melhor precisão para cálculos que a constante gravitacional universal G, o valor de μ é $3.986 \times 10^{15} Km^3 / s^2$.

4.2.3. Órbita circular

Neste trabalho a órbita do satélite escolhido é a circular. Uma órbita circular é um caso particular da órbita elíptica e permite algumas simplificações de parâmetros, confome a seguir:

a) Excenticidade nula:

$$e = 0 ; \tag{4.8}$$

Devido a excentricidade nula, não é possível se determinar o perigeu.

b) Semi-eixo maior igual ao raio:

$$a = r ; \tag{4.9}$$

c) A anomalia média coincide com a anomalia excêntrica e com a anomalia verdadeira, isto é:

$$M = u = f ; \tag{4.10}$$

d) A velocidade orbital é constante:

$$v_o = \sqrt{\frac{\mu}{a}} \text{ [km/s]}; \tag{4.11}$$

e) A Equação Paramétrica é dada por:

$$x^2 + y^2 = a^2 ; \tag{4.12}$$

f) A Energia da órbita (E), obtida a partir da equação $E = \frac{v_o^2}{2} - \frac{\mu}{r}$, é constante e pode ser simplificada para:

$$E = -\frac{\mu}{2a} \text{ [J];} \tag{4.13}$$

g) O período é dado por:

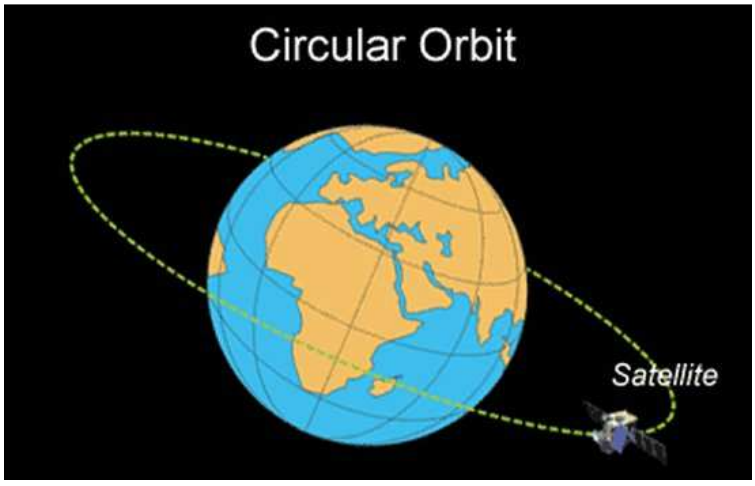
$$Ps = 2\pi\sqrt{\frac{a^3}{\mu}} \text{ [s];} \tag{4.14}$$

$$P\min = 0.0001658\sqrt{a^3};$$

h) Número de voltas que o satélite completa por dia:

$$\text{Re}\, v\_ por\_ dia = \frac{24 \times 60}{P \min}. \tag{4.15}$$

Figura 4.8 – Órbita Circular.



Fonte: http://www.aerospaceweb.org/question/spacecraft/q0164.shtml (24/04/2017).

4.3. Classificação de Órbitas

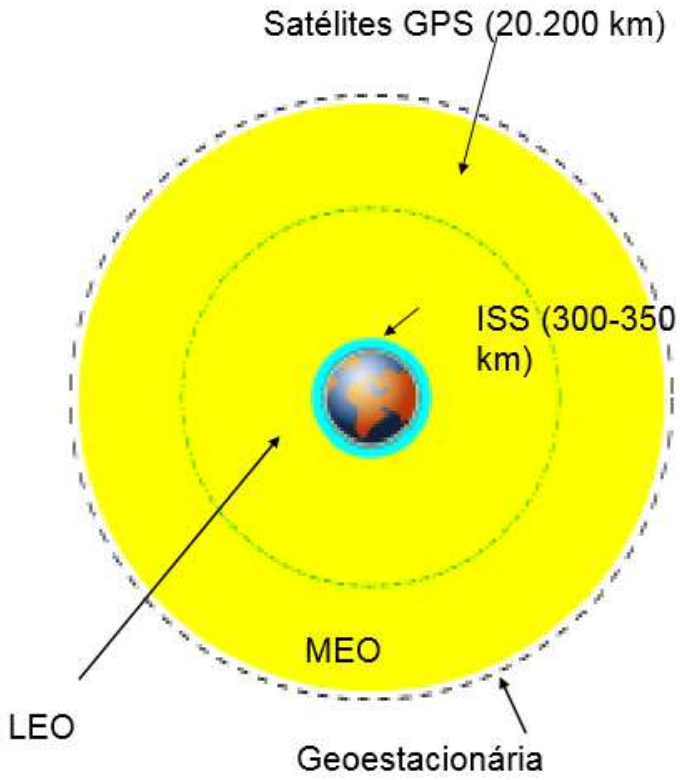
Para órbitas terrestres e geocêntricas, existe uma classificação de acordo com a altitude, conforme a seguir:

Órbitas Terrestres Baixas ou LEO (*Low Earth Orbit*): são órbitas geocêntricas de até 2.000 km da altitude.

Órbitas Terrestres Médias ou MEO (*Medium Earth Orbit*): são órbitas geocêntricas com altitudes superiores a 2.000 km e inferiores a da órbita geoestacionária (35.786 km).

Órbitas Terrestres Altas ou HEO (High Earth Orbit): órbitas geocêntricas com altitudes superiores à da órbita geoestacionária.

Figura 4.9 – Órbitas Terrestres Geoestacionárias.



Fonte: Bogossian (2015).

4.4. Equações Utilizadas no Modelo

De acordo com Wertz (1978), o movimento de um satélite artificial é especificado de duas formas:

- Por Equações Cinemáticas: descritas por sua posição e velocidade lineares (órbita e movimento orbital), caracterizando o movimento de translação do centro de massa do satélite,
- Por Equações Dinâmicas: descritas por sua posição e velocidade angulares (atitude e movimento de atitude), caracterizando o movimento de rotação do satélite em torno do seu centro de massa.

4.4.1. Equações Cinemáticas de Movimento

4.4.1.1. Transformações entre Sistemas de Coordenadas

Para o desenvolvimento das equações Cinemáticas se faz necessário o uso das seguintes técnicas de transformação de coordenadas:

- Ângulos de Euler;
- Parâmetros simétricos de Euler (quatérnions);
- Ângulos e eixo equivalente.

A atitude do Satélite pode ser representada por ângulos de Euler e/ou por quatérnions. Dependendo da aplicação, uma representação pode ser melhor que outra.

Por exemplo, para representação de gráficos, a representação por ângulos de Euler é melhor, pois apresenta maior sentido físico (Moreira, 2006).

Já para cálculos computacionais é melhor utilizar os quatérnions, para se evitar as singularidades das equações trigonométricas. Os ângulos de Euler também apresentam resultados mais simplificados de quantidades de operações, quando comparado com o tratamento que utiliza as funções trigonométricas.

Aqui neste trabalho, se utilizam os ângulos de Euler nas equações cinemáticas.

4.4.1.2. Matriz de Rotação ou Matriz dos Cossenos Diretores

Tagawa (2013) apresenta um desenvolvimento matrizes de rotação para a obtenção dos ângulos de Euler.

Um vetor A pode ser expresso no sistema de coordenadas original () e no sistema rotacionado (´), como definido a seguir:

$$A = A_x i + A_y j + A_z k = A'_x i' + A'_y j' + A'_z k' \tag{4.16}$$

Na forma matricial:

$$\begin{Bmatrix} A'_x \\ A'_y \\ A'_z \end{Bmatrix} = \begin{pmatrix} i'.i & i'.j & i'.k \\ j'.i & j'.j & j'.k \\ k'.i & k'.j & k'.k \end{pmatrix} \begin{Bmatrix} A_x \\ A_y \\ A_z \end{Bmatrix} \tag{4.17}$$

E, inversamente:

$$\begin{Bmatrix} A_x \\ A_y \\ A_z \end{Bmatrix} = \begin{pmatrix} i.i' & i.j' & i.k' \\ j.i' & j.j' & j.k' \\ k.i' & k.j' & k.k' \end{pmatrix} \begin{Bmatrix} A'_x \\ A'_y \\ A'_z \end{Bmatrix} \tag{4.18}$$

Podem ser definidas então as matrizes C e C`, chamadas de matrizes de cossenos diretores:

$$C = \begin{pmatrix} i'.i & i'.j & i'.k \\ j'.i & j'.j & j'.k \\ k'.i & k'.j & k'.k \end{pmatrix} \text{ e } C` = \begin{pmatrix} i.i' & i.j' & i.k' \\ j.i' & j.j' & j.k' \\ k.i' & k.j' & k.k' \end{pmatrix} = C^T = C^{-1} \tag{4.19}$$

Cada termo da matriz é o cosseno dos ângulos entre eixos dos dois sistemas de coordenadas, como mostrado na Figura 4.10.

Para uma rotação Φ em torno do eixo x, temos das propriedades trigonométricas:

$$i.i' = 1,\ j.i' = 0, k.i' = 0; \tag{4.20}$$

$$i.j' = 0,\ j.j' = \cos\ \phi, k.j' = \cos\left(90 - \phi\right); \tag{4.21}$$

$$i.k' = 0,\ j.k' = \cos\left(90 + \phi\right), k.k' = \cos\ \phi\ ; \tag{4.22}$$

Utilizando as propriedades:

$$\cos\left(x + y\right) = \cos\ x.\cos\ y - senx\ .seny \tag{4.23}$$

$$\cos\left(x - y\right) = \cos\ x.\cos\ y + senx\ .seny \tag{4.24}$$

As equações podem ser reduzidas para:

$$\cos\left(90 + \phi\right) = \cos\left(90\right).\cos\ \phi - sen\left(90\right).sen\ \phi \tag{4.25}$$

$$\cos\left(90 + \phi\right) = -sen\ \phi \tag{4.26}$$

$$\cos\left(90 - \phi\right) = \cos\left(90\right).\cos\ \phi + sen\left(90\right).sen\ \phi \tag{4.27}$$

$$\cos\left(90 - \phi\right) = sen\ \phi \tag{4.28}$$

Então, a matriz C pode ser escrita como:

$$C_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & sen\phi \\ 0 & -sen\phi & \cos\phi \end{bmatrix} \tag{4.29}$$

Portanto, considerando uma rotação no eixo x:

$$\begin{bmatrix} A'_x \\ A'_y \\ A'_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & sen\phi \\ 0 & -sen\phi & \cos\phi \end{bmatrix} \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} \tag{4.30}$$

Repetindo a dedução para uma rotação em torno do eixo Y, a matriz C2 é dada por:

$$C_2 = \begin{bmatrix} \cos\theta & 0 & -sen\theta \\ 0 & 1 & 0 \\ sen\theta & 0 & \cos\theta \end{bmatrix} \tag{4.31}$$

Substituindo a transformação matricial fica:

$$\begin{bmatrix} A'_x \\ A'_y \\ A'_z \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & -sen\theta \\ 0 & 1 & 0 \\ sen\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} \tag{4.32}$$

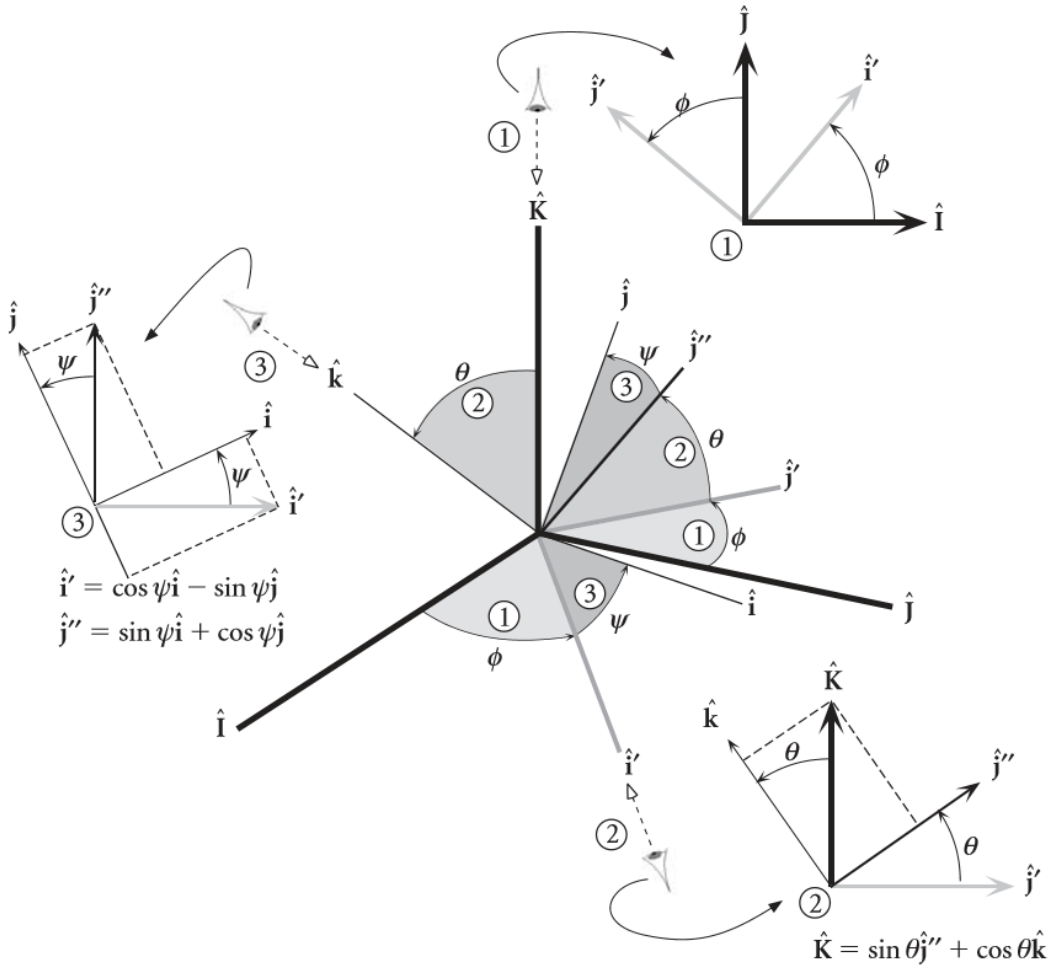
Do mesmo modo para o eixo Z, a matriz C3 é dada por:

$$C_3 = \begin{bmatrix} \cos\psi & sen\psi & 0 \\ -sen\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.33}$$

Substituindo a transformação matricial fica:

$$\begin{bmatrix} A'_x \\ A'_y \\ A'_z \end{bmatrix} = \begin{bmatrix} \cos\psi & sen\psi & 0 \\ -sen\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} \tag{4.34}$$

Figura 4.10 – Rotações nos Eixos x, y e z.



Fonte: Curtis (2005).

4.4.1.3. Ângulos de Euler

Uma das maneiras utilizadas para se representar a orientação ou atitude de um sistema de coordenadas em relação a outro é baseada no uso de três ângulos, cada um em torno de um eixo particular. Esta representação de atitude pelos três

ângulos é conhecida como representação dos ângulos de Euler, e os ângulos propriamente ditos como ângulos de Euler (Tagawa, 2013).

Há diferentes sequências de rotações, sendo que a ordem é escolhida dependendo da situação. Neste trabalho, o ângulo de rolamento (φ) equivale à rotação em torno do eixo x do corpo, o ângulo de arfagem (θ), em torno do eixo y do corpo, e o ângulo de guinada (ψ) em torno do eixo z do corpo.

Na descrição da orientação de um referencial fixo ao corpo do satélite (RS) em relação ao Referencial Orbital (RO) em termos dos três ângulos de Euler (φ, θ, ψ), a sequência de rotações 3-2-1 (C1(φ) ← C2(θ) ← C3(ψ)) de O para S foi considerada. As setas indicam a ordem das rotações; e a disposição das matrizes Ci, i = 1, 2, 3, corresponde à posição das mesmas para a multiplicação de matrizes que resulta na matriz de rotação de RO para RS.

$$\vec{S} = R_{O/S}\vec{O} \tag{4.35}$$

$$R_{O/S} = C_1(\phi)C_2(\theta)C_3(\psi) \tag{4.36}$$

$$R_{O/S} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & sen\phi \\ 0 & -sen\phi & \cos\phi \end{bmatrix}.\begin{bmatrix} \cos\theta & 0 & -sen\theta \\ 0 & 1 & 0 \\ sen\theta & 0 & \cos\theta \end{bmatrix}.\begin{bmatrix} \cos\psi & sen\psi & 0 \\ -sen\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.37}$$

$$R_{O/S} = \begin{bmatrix} \cos\theta.\cos\psi & \cos\theta.sen\psi & -sen\theta \\ sen\phi.sen\theta.\cos\psi - \cos\phi.sen\psi & sen\phi.sen\theta.sen\psi + \cos\phi.\cos\psi & sen\phi.\cos\theta \\ \cos\phi.sen\theta.\cos\psi + sen\phi.sen\psi & \cos\phi.sen\theta.sen\psi - sen\phi.\cos\psi & \cos\phi.\cos\theta \end{bmatrix} \tag{4.38}$$

Corresponde à sequência de rotações 3-2-1, que converte do sistema referencial X, Y, Z para o sistema X3, Y3, Z3.

Essas rotações seguem a seguinte sequência:

- Rotação de ψ graus em torno do eixo Z, transformando o sistema de X, Y, Z em X1, Y1, Z1;

- Rotação de θ graus em torno do eixo Y1, transformando o sistema de X1, Y1, Z1 em X2, Y2, Z2;

- Rotação de ϕ graus em torno do eixo X2, transformando o sistema de X2, Y2, Z2 em X3, Y3, Z3;

Para se retornar ao Sistema original X, Y, Z, basta seguir a sequência inversa de rotação. O caminho inverso irá requerer uma sequência de três rotações para se retornar à posição de ψ, duas rotações para θ e uma para ϕ.

Assim, a velocidade angular nos três eixos do Sistema RS (referencial do satélite) relativa ao Sistema RO (referencial orbital), $\vec{\omega}_{O/S}$ é dada por:

$$\begin{bmatrix} \vec{\omega}_x \\ \vec{\omega}_y \\ \vec{\omega}_z \end{bmatrix} = C_1(\phi)C_2(\theta)C_3(\psi)\begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + C_1(\phi)C_2(\theta)\begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + C_1(\phi)\begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} \tag{4.39}$$

Da multiplicação matricial, resulta:

$$\begin{bmatrix} \vec{\omega}_x \\ \vec{\omega}_y \\ \vec{\omega}_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & -sen\theta \\ 0 & \cos\phi & sen\phi\cos\theta \\ 0 & -sen\phi & \cos\phi\cos\theta \end{bmatrix}\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{4.40}$$

Porém, o satélite apresenta uma velocidade orbital $\vec{\omega}_0$ que deve ser compensada na equação da seguinte forma $\vec{\omega} = \vec{\omega}_{O/S} - \vec{\omega}_0 o_2$. A velocidade angular com que o referencial RO gira em relação ao referencial inercial é:

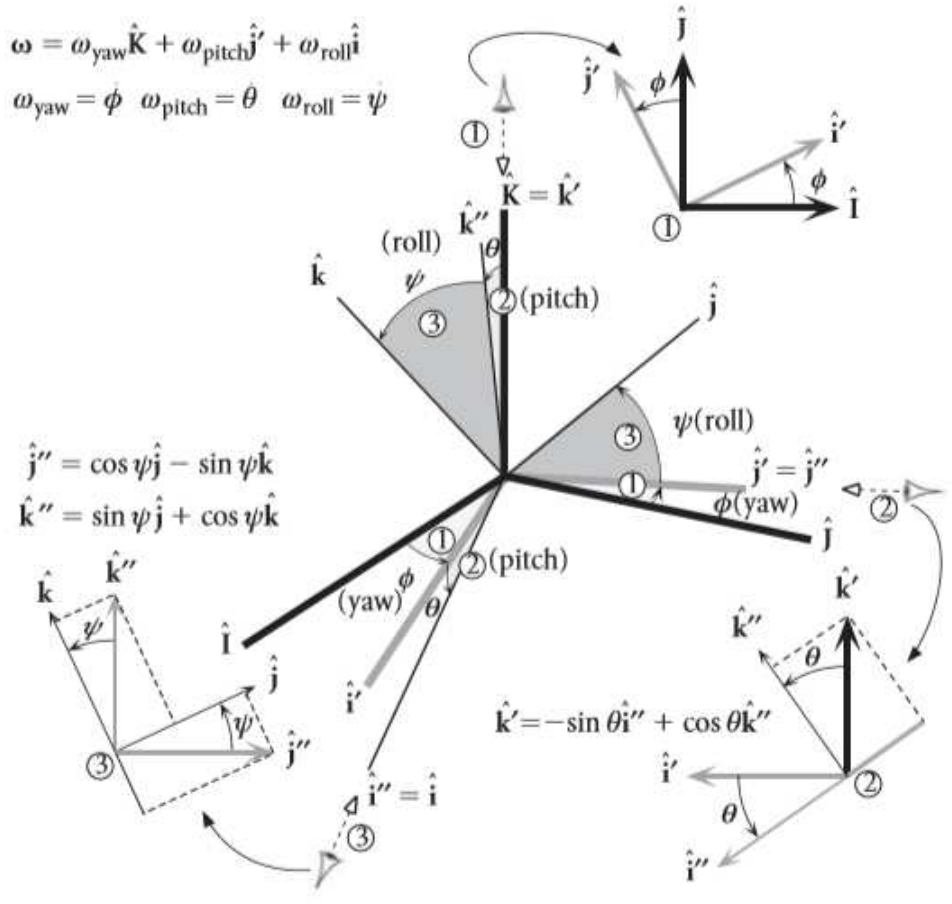
$$\begin{bmatrix} \vec{\omega}_x \\ \vec{\omega}_y \\ \vec{\omega}_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & -sen\theta \\ 0 & \cos\phi & sen\phi\cos\theta \\ 0 & -sen\phi & \cos\phi\cos\theta \end{bmatrix}\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} - \vec{\omega}_0\begin{bmatrix} \cos\theta sen\psi \\ sen\phi sen\theta sen\psi + \cos\phi\cos\psi \\ \cos\phi sen\theta sen\psi - \cos\phi\cos\psi \end{bmatrix} \tag{4.41}$$

Dessa forma, pode-se definir a equação diferencial da cinemática de um corpo rígido em órbita, obtida da sequência de rotação 3-2-1, da seguinte forma:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \frac{1}{\cos\theta} \begin{bmatrix} \cos\theta & sen\phi sen\theta & \cos\phi sen\theta \\ 0 & \cos\phi\cos\theta & -sen\phi\cos\theta \\ 0 & sen\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \varpi_x \\ \varpi_y \\ \varpi_z \end{bmatrix} - \frac{\varpi_0}{\cos\theta} \begin{bmatrix} sen\psi \\ \cos\theta\cos\psi \\ sen\theta sen\psi \end{bmatrix} \qquad (4.42)$$

Essas equações também foram desenvolvidas em Moreira (2006), Gobato (2006) e Tagawa (2013).

Figura 4.11 – Ângulos de Euler, *Yaw* (Guinada), *Pitch* (Arfagem) e *Roll* (Rolamento).



Fonte: Curtis (2005).

O estudo da cinemática de um satélite está focado na geometria do movimento, e não considera os aspectos de massa e força. Dessa forma, esse estudo é composto de métodos de cálculo matricial para descrever posições, velocidades e acelerações de corpos rígidos descritas em diferentes referenciais de coordenadas, acompanhando a evolução da orientação entre os mesmos ao longo do tempo (Gobato, 2006).

As equações cinemáticas do movimento são um conjunto de equações diferenciais de primeira ordem, as quais contém o vetor velocidade angular instantâneo w e especificam a evolução no tempo de parâmetros de atitude (ϕ , θ e ψ).

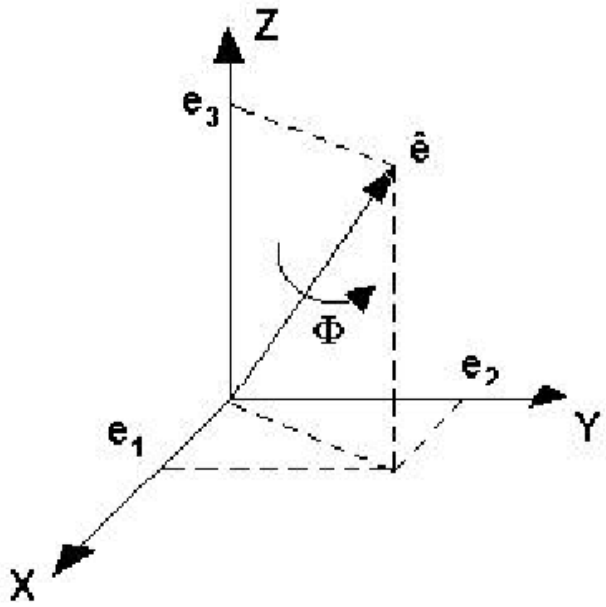
4.4.1.4. Quatérnions

A representação por quatérnions, ou parâmetros simétricos de Euler, é bastante utilizada por apresentar uma formatação mais compacta, apresentando apenas quatro parâmetros, ao invés dos nove quando se utiliza os ângulos de Euler.

Outra vantagem da utilização dos quatérnions é que a matriz de conversão não apresenta funções trigonométricas, as quais além de requererem mais processamento nos cálculos, podem apresentar singularidades, dependendo do ângulo.

Esta representação é baseada na rotação de um ângulo Φ, em torna de um eixo ê, com componentes (e1, e2, e3) em relação a um sistema de referência. Normalmente, o eixo de rotação não é nenhum dos eixos do sistema de referência. A Figura 4.12 representa essa rotação.

Figura 4.12 – Rotação em torno do Eixo ê de um Ângulo Φ.



Fonte: Gobato (2006).

O vetor de quatérnios (q1, q2, q3, q4) é definido a partir dos parâmetros descritos conforme a equação abaixo:

$$q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{\Phi}{2}\right) \\ e_2 sen\left(\frac{\Phi}{2}\right) \\ e_3 sen\left(\frac{\Phi}{2}\right) \\ e_4 sen\left(\frac{\Phi}{2}\right) \end{bmatrix} \tag{4.43}$$

A equação cinemática representada por quaternions, para levar do referencial RO para o RS é dada por:

$$\dot{q} = \frac{1}{2}\begin{bmatrix} 0 & \omega_x & -\omega_y + \omega_o & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y + \omega_o \\ \omega_y - \omega_o & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y - \omega_o & -\omega_z & 0 \end{bmatrix} q \tag{4.44}$$

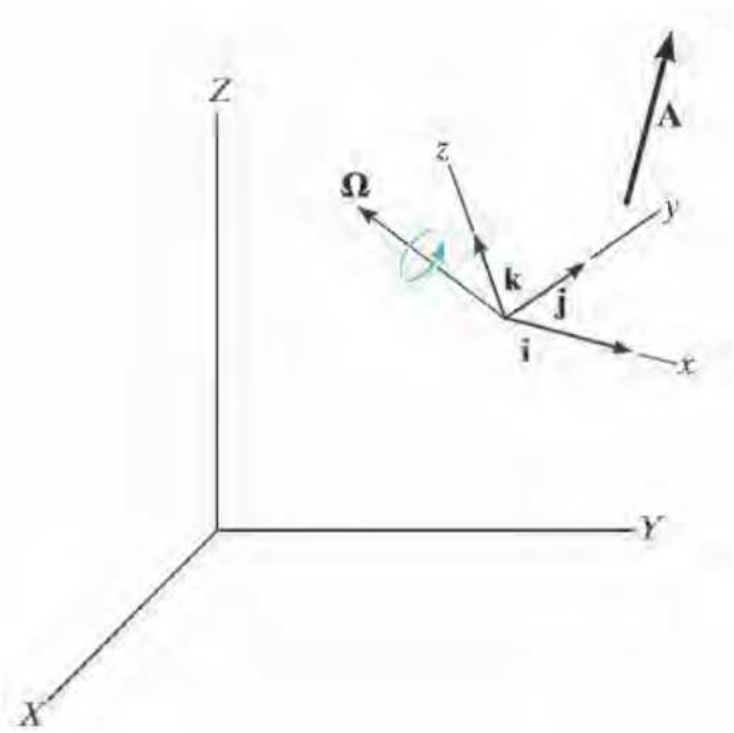
4.4.2. Equações Dinâmicas

O movimento de um satélite artificial é definido por sua posição e velocidade lineares e angulares. A posição e velocidade lineares descrevem o movimento do centro de massa (translação) do satélite, ou seja, seu movimento orbital. A posição e velocidade angulares descrevem o movimento em torno do centro de massa (rotação) do satélite, ou seja, o seu movimento de atitude, descrito a seguir.

4.4.2.1. Rotação de um sistema de coordenadas em relação a um sistema de coordenadas inerciais

O desenvolvimento de uma equação que represente a rotação de um sistema de coordenadas em relação a um sistema de coordenadas inerciais é demonstrado a seguir, conforme apresentado por Tagawa (2013).

Figura 4.13 – Vetor em um Sistema de Coordenadas Girante.



Fonte: Tagawa (2013).

A Figura 4.13 ilustra um vetor A em um sistema de coordenadas x, y, z em rotação. A velocidade angular Ω deste sistema de coordenadas é medida em relação a um sistema de coordenadas inerciais.

O vetor A expresso em função de suas componentes no sistema de coordenadas em rotação é:

$$A = A_x i + A_y j + A_z k \tag{4.45}$$

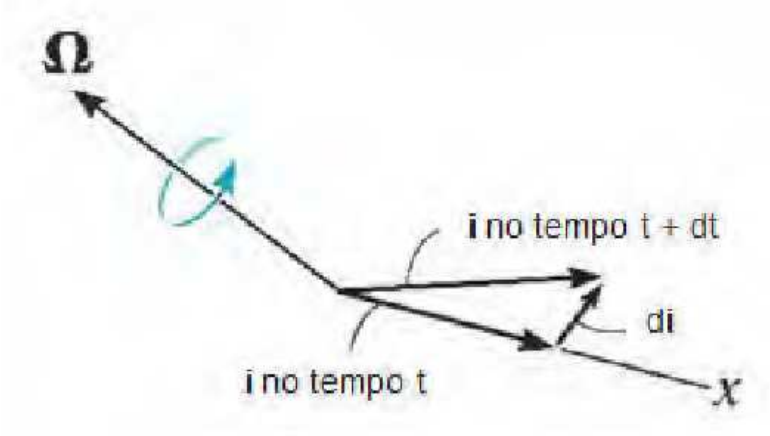
A derivada no tempo de A em relação ao sistema de coordenadas inerciais resulta na equação:

$$\dot{A} = \dot{A}_x i + \dot{A}_y j + \dot{A}_z k + A_x \dot{i} + A_y \dot{j} + A_z \dot{k} \tag{4.46}$$

Nota-se que também há variação das direções (i, j, k), pois este sistema está em rotação em relação ao sistema de coordenadas inerciais.

Como mostrado na Figura 4.14, $\dot{i}$ representa apenas a mudança na direção de i em relação ao tempo, pois i é um vetor unitário, ou seja, apresenta magnitude unitária.

Figura 4.14 – Relação de i e di.



Fonte: Tagawa (2013).

Como di é perpendicular a i, conforme mostra a Figura 4.14 podemos definir di como:

$$\dot{i} = \Omega \times i \tag{4.47}$$

E para os demais eixos:

$$\dot{j} = \Omega \times j \tag{4.48}$$

$$\dot{k} = \Omega \times k$$

Substituindo (4.48) em ($\dot{A} = \dot{A}_x i + \dot{A}_y j + \dot{A}_z k + A_x \dot{i} + A_y \dot{j} + A_z \dot{k}$):

$$\dot{A} = \dot{A}_x i + \dot{A}_y j + \dot{A}_z k + A_x(\Omega \times i) + A_y(\Omega \times j) + A_z(\Omega \times k) \tag{4.49}$$

Utilizando-se a seguinte propriedade de vetores:

$$(la)\times b = l(a\times b) = a\times(lb) \tag{4.50}$$

Obtém-se:

$$A_x(\Omega\times i) = \Omega\times(A_x i) \tag{4.51}$$

$$A_y(\Omega\times j) = \Omega\times(A_y j) \tag{4.52}$$

$$A_z(\Omega\times k) = \Omega\times(A_z k) \tag{4.53}$$

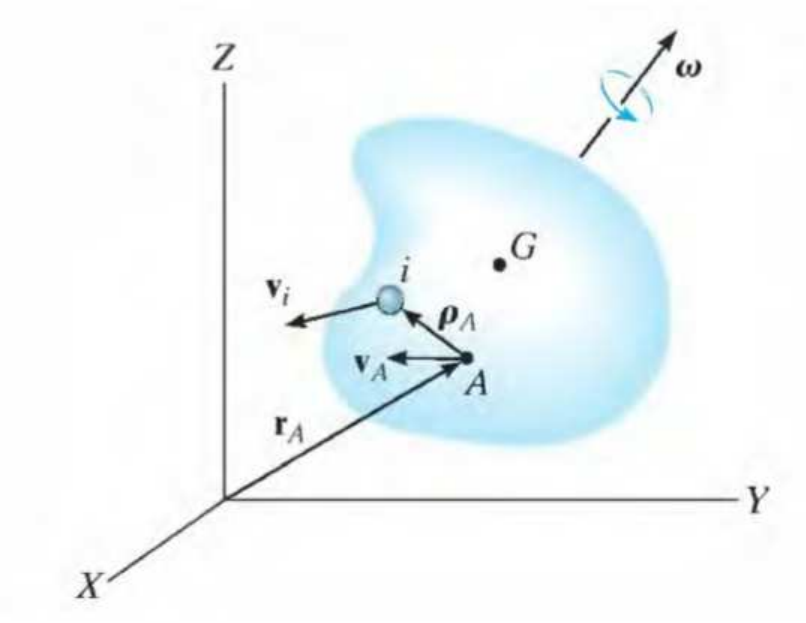
Chega-se na equação:

$$\dot{A} = (\dot{A})_{xyz} + \Omega\times A \tag{4.54}$$

4.4.2.2. Quantidade de movimento angular:

Conforme apresentado por Tagawa (2013), considerando o corpo rígido da Figura 4.15, o qual tem massa m, velocidade angular ω e centro de massa em G, tem-se como formulação da quantidade de movimento angular $(H_Á)_i$ da partícula i com massa m_i e posição $(\rho_A)_i$ em relação a A, em torno do ponto A no corpo:

Figura 4.15 – Sistema de Coordenadas Inerciais.



Fonte: Tagawa (2013).

$$(H_A)_i = \rho_A \times m_i v_i \tag{4.55}$$

Onde v_i representa a velocidade da partícula i medida a partir do sistema de coordenadas inerciais (X, Y, Z). Se o corpo tem uma velocidade angular ω no instante considerado, v_i deve ser relacionada com a velocidade v_A de A utilizando-se a equação (4.55):

$$v_i = v_A + \omega \times (\rho_A)_i \tag{4.56}$$

$$(H_A)_i = (\rho_A)_i \times m_i \left(v_A + \omega \times (\rho_A)_i\right) \tag{4.57}$$

$$(H_A)_i = \left((\rho_A)_i\, m_i\right) \times v_A + (\rho_A)_i \times (\omega \times (\rho_A)_i) m_i \tag{4.58}$$

Para a soma da quantidade de movimento de todas as partículas do corpo é preciso uma integração. Fazendo mi → dm, e integrando sobre m temos:

$$H_A = \left(\int_m \rho_A dm \right) \times v_A + \int_m \rho_A \times (\omega \times \rho_A) dm \tag{4.59}$$

Neste trabalho, considera-se que A está localizado no centro de massa (G) do corpo. Deste modo, $\int_m \rho_A dm = 0$, portanto chega-se:

$$H_A = \int_m \rho_A \times (\omega \times \rho_A) dm \tag{4.60}$$

Expresso em termos das componentes x, y, z:

$$H_x i + H_y j + H_z k = \int_m (xi + yj + zk) \times \left[(\omega_x i + \omega_y j + \omega_z k) \times (xi + yj + zk) \right] dm \tag{4.61}$$

Realizando os produtos vetoriais e combinando os termos chaga-se a:

$$\begin{aligned} H_x i + H_y j + H_z k = & \left[\omega_x \int_m (y^2 + z^2) dm - \omega_y \int_m xy dm - \omega_z \int_m xz dm \right] i \\ & + \left[-\omega_x \int_m xy dm + \omega_y \int_m (x^2 + z^2) dm - \omega_z \int_m yz dm \right] j \\ & + \left[-\omega_x \int_m zx dm + \omega_y \int_m yz dm - \omega_z \int_m (x^2 + y^2) dm \right] k \end{aligned} \tag{4.62}$$

Seperando os momentos de Inércia, chega-se às equações:

$$H_x = I_{xx}\omega_x - I_{xy}\omega_y - I_{xz}\omega_z \tag{4.63}$$

$$H_y = -I_{yx}\omega_x + I_{yy}\omega_y - I_{yz}\omega_z \tag{4.64}$$

$$H_z = -I_{zx}\omega_x - I_{zy}\omega_y + I_{zz}\omega_z \tag{4.65}$$

Considerando-se os eixos alinhados, de forma que coincidam com os eixos principais de inércia, os produtos de inércia tornam-se nulos e definindo $I_{xx} \equiv I_x$, $I_{yy} \equiv I_y$ e $I_{zz} \equiv I_z$, obtem-se o momento angular de cada eixo:

$$H_x = I_x \omega_x \tag{4.66}$$

$$H_y = I_y \omega_y \tag{4.67}$$

$$H_z = I_z \omega_z \tag{4.68}$$

A somatória de todos os torques externos aplicados a um corpo é igual a taxa de variação de quantidade de movimento angular desse corpo:

$$\sum M = \dot{H} \tag{4.69}$$

4.4.2.3. Equações de Movimento de Euler

Considerando que o satélite gira em relação ao sistema de coordenadas inerciais, acrescenta-se a velocidade angular à equação da inércia, para considera essa rotação:

$$\sum M = H_{xyz} + \Omega \times H \tag{4.70}$$

Pode-se ainda fazer uma aproximação, considerando que os eixos de rotação estão fixos no corpo, a velocidade angular do sistema de coordenadas Ω é igual à velocidade angular do corpo ω. Então, a equação pode ser reescrita:

$$\sum M = H_{xyz} + \omega \times H \tag{4.71}$$

Escrevendo as equações de momento de inércia na forma escalar e com os produtos de inércia, para cada eixo:

$$\sum M_x = I_{xx}\dot{\omega}_x - (I_{yy} - I_{zz})\omega_y\omega_z - I_{xy}(\dot{\omega}_y - \omega_z\omega_x) - I_{yz}(\omega_y^2 - \omega_z^2) - I_{zx}(\dot{\omega}_z - \omega_x\omega_y) \quad (4.72)$$

$$\sum M_y = I_{yy}\dot{\omega}_y - (I_{zz} - I_{xx})\omega_x\omega_z - I_{yz}(\dot{\omega}_z - \omega_x\omega_y) - I_{zx}(\omega_z^2 - \omega_x^2) - I_{xy}(\dot{\omega}_x - \omega_y\omega_z) \quad (4.73)$$

$$\sum M_z = I_{zz}\dot{\omega}_z - (I_{xx} - I_{yy})\omega_x\omega_y - I_{zx}(\dot{\omega}_x - \omega_y\omega_z) - I_{xy}(\omega_x^2 - \omega_y^2) - I_{yz}(\dot{\omega}_y - \omega_z\omega_x) \quad (4.74)$$

Considerando-se os eixos alinhados, de forma que coincidam com os eixos principais de inércia, os produtos de inércia tornam-se nulos e definindo $I_{xx} \equiv I_x$, $I_{yy} \equiv I_y$ e $I_{zz} \equiv I_z$, obtem-se :

$$\sum M_x = I_x\dot{\omega}_x - (I_y - I_z)\omega_y\omega_z \quad (4.75)$$

$$\sum M_y = I_y\dot{\omega}_y - (I_z - I_x)\omega_x\omega_z \quad (4.76)$$

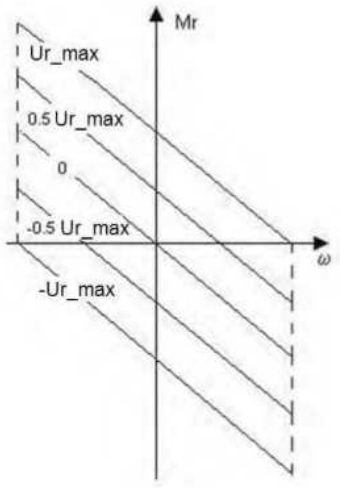
$$\sum M_z = I_z\dot{\omega}_z - (I_x - I_y)\omega_x\omega_y \quad (4.77)$$

Essas equações são conhecidas como as equações de movimento de Euler.

4.4.3. Equações do Atuador

Nesse trabalho é utilizado o modelo de atuador desenvolvido por Souza (1980) e Gobato (2006). Esse modelo apresenta uma aproximação linear da curva característica de um servomotor CC, conforme mostrado abaixo:

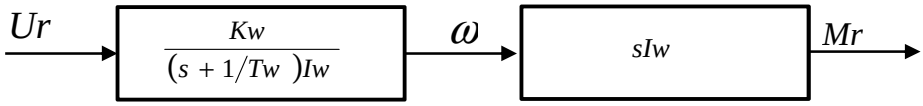
Figura 4.16 – Aproximação linear da curva característica do servomotor.



Fonte: Souza (1980).

O diagrama de blocos do atuador está mostrado na Figura 4.17 abaixo:

Figura 4.17 – Diagrama de Blocos do Atuador.



Os parâmetros do atuador são:

- Tw = constante de tempo da roda [s];
- Kw = ganho da roda;

- Ir = inércia da roda [kg.m2];
- $\omega_{R\max}$ = velocidade angular da roda [rpm];
- $M_{R\max}$ = torque da roda [N.m] e
- $U_{R\max}$ = a tensão da roda [V].

Os parâmetros da roda são obtidos a partir das equações a seguir (Souza, 1980):

$$Tw = \frac{I_R \omega_{R\max}}{M_{R\max}} \tag{4.78}$$

$$Kw = \frac{M_{R\max}}{U_{R\max}} \tag{4.80}$$

As funções de transferência apresentadas na Figura 4.17 foram obtidas das equações a seguir (Souza, 1980; Gobato, 2006):

$$\frac{\omega_R}{U_R} = \frac{Kw}{(s + 1/Tw)Iw} \tag{4.81}$$

$$\frac{M_R}{\omega_R} = sIw \tag{4.82}$$

Logo:

$$\frac{M_R}{U_R} = \left(\frac{Kw}{(s + 1/Tw)Iw} \right) sIw \tag{4.83}$$

Usando a entrada aplicando a transformada inversa na equação (4.81) se obtém a aceleração angular e o momento em função da aceleração:

$$\dot{\omega}_r(t) = \frac{K_w}{I_w} u_r(t) - \frac{\omega_r(t)}{T_w} \tag{4.84}$$

$$M_r(t) = I_w \dot{\omega}_r(t) \tag{4.85}$$

Para os três eixos:

$$\dot{\omega}_{rx}(t) = \frac{K_w}{I_w} u_r(t) - \frac{\omega_{rx}(t)}{T_w}$$

$$\dot{\omega}_{ry}(t) = \frac{K_w}{I_w} u_r(t) - \frac{\omega_{ry}(t)}{T_w} \tag{4.86}$$

$$\dot{\omega}_{rz}(t) = \frac{K_w}{I_w} u_r(t) - \frac{\omega_{rz}(t)}{T_w}$$

$$M_{rx}(t) = I_w \dot{\omega}_{rx}(t)$$

$$M_{ry}(t) = I_w \dot{\omega}_{ry}(t) \tag{4.87}$$

$$M_{rz}(t) = I_w \dot{\omega}_{rz}(t)$$

4.4.4. Equações do Controlador

No modelo do satélite, o controlador utilizado é o PID (Proporcional-Integral-Derivativo), também utilizadas por Gobato (2006).

Os ganhos (Kp, Td, Ti) do PID são obtidos do comportamento dinâmico em termos dos parâmetros ζ (razão de amortecimento) e ω_n (frequência natural não amortecida).

Figura 4.18 – Diagrama de Blocos do Modelo do SCA do Satélite para o eixo X.

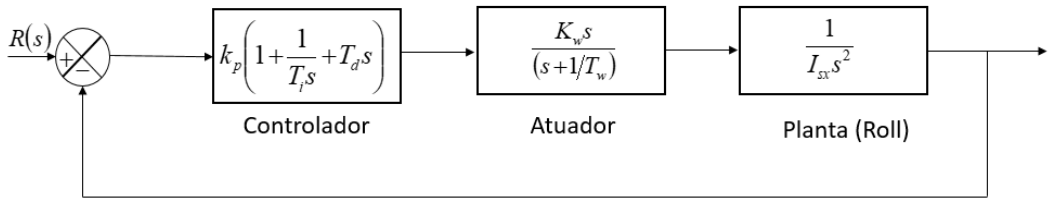


Figura 4.19 – Diagrama de Blocos do Modelo do SCA do Satélite para o eixo Y.

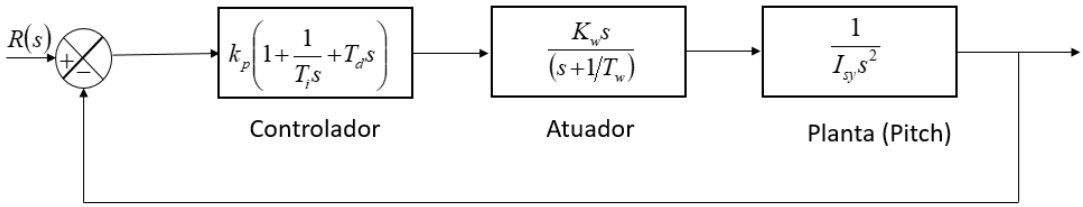
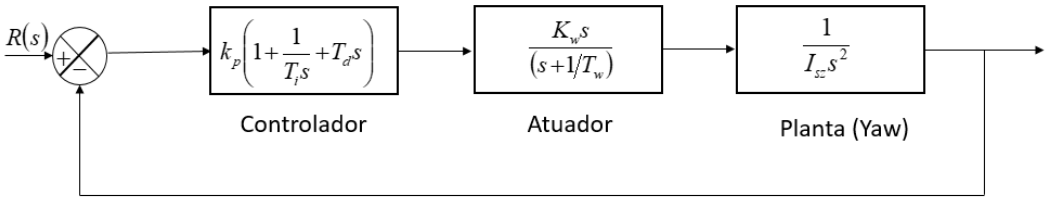


Figura 4.20 – Diagrama de Blocos do Modelo do SCA do Satélite para o eixo Z.

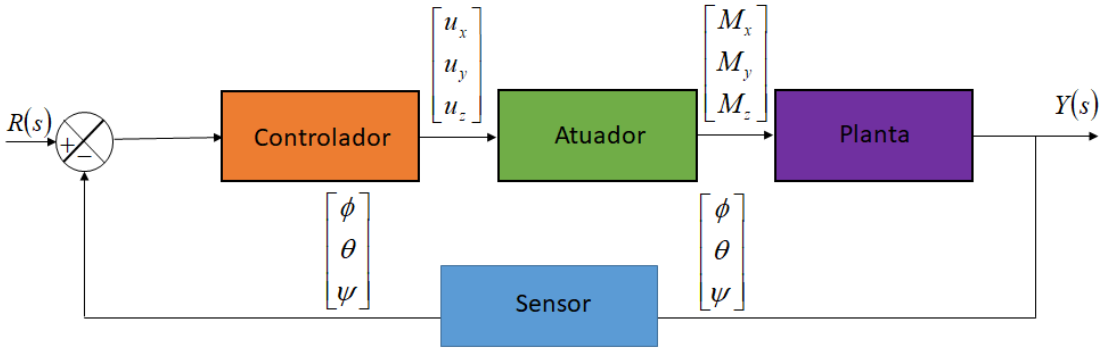


5 ADAPTAÇÃO DO MODELO AOS SIMULADORES COM HLA/RTI

Para a aplicação de uma simulação com HLA/RTI ao modelo aeroespacial, foi utilizado o modelo do SCA do satélite Amazônia-1, implementado em C++ e comparado com um modelo em MATLAB/Simulink dessas mesmas equações.

A implementação do modelo no MATLAB/Simulink está detalhada no Apêndice A.

Figura 5-1. Modelo do SCA implementado em C++.



Fonte: Moraes (2017).

As equações utilizadas para a modelagem são as equações dos movimentos de cinemática e dinâmica desenvolvidas no capítulo 4.

As tabelas apresentadas a seguir contêm os valores utilizados na modelagem e simulação das equações do Satélite. Esses valores consistem de constantes, tais como as relacionadas aos parâmetros orbitais, e a valores iniciais e são os mesmos usados por Gobato (2006) e Tagawa (2013).

5.1. Valores Orbitais

Considerando um satélite de sensoriamento remoto (SSR) colocado em órbita baixa (*Low Earth Orbit* – LEO), os parâmetros orbitais estão resumidos na Tabela 5.1.

Tabela 5.1 – Parâmetros Orbitais.

Altitude (h)	905 [km]
Excentricidade (e)	0
Inclinação (i)	0°
Velocidade Orbital Média (ω_0)	0.001016 [rad/s]
Período Orbital (T_0)	6184[s]

5.2. Dados do Satélite e da Roda de Reação

A Tabela 5.2 apresenta os dados do satélite e da roda de reação utilizados para os cálculos nos programas em C++ e em MATLAB/Simulink.

Tabela 5.2 – Dados do Satélite e da Roda de Reação.

Momento de Inércia das Rodas (I_R)	0.015 $[kg.m^2]$
Velocidade Máxima das Rodas ($\omega_{R\max}$)	7500 [r.p.m]
Torque Máximo das Rodas ($M_{R\max}$)	0.6 [N.m]
Tensão Máxima na Saída do Controlador ($U_{R\max}$)	10 [V]
Ganho das Rodas (K_w)	0.06
Constante de Tempo das Rodas (T_w)	20 [s]
Momento de Inércia do Satélite em X (I_x)	295.71 $[kg.m^2]$
Momento de Inércia do Satélite em Y (I_y)	501.37 $[kg.m^2]$
Momento de Inércia do Satélite em Z (I_z)	364.82 $[kg.m^2]$

5.3. Dados do PID

A Tabela 5.3 apresenta os dados do controlador PID, considerado nos cálculos dos programas em C++ e em MATLAB/Simulink.

Tabela 5.3 – Dados do PID.

Eixo	Ganhos	Valores
X	K_p	40.5931
	T_d	454.1050
	T_i	1.0000
Y	K_p	51.7854
	T_d	556.9350
	T_i	1.0000
Z	K_p	44.3541
	T_d	488.6600
	T_i	1.0000

5.4. Condições Iniciais

A Tabela 5.4 apresenta as condições iniciais consideradas nos cálculos dos programas em C++ e em MATLAB/Simulink.

Tabela 5.4 – Condições Iniciais.

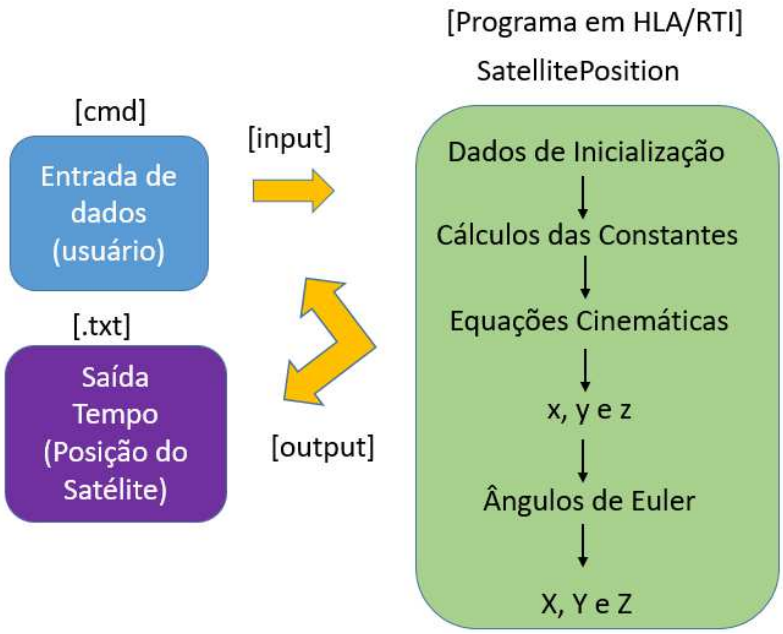
Ângulo em X (ϕ)	30 [◦]
Ângulo em Y (θ)	0 [◦]
Ângulo em Z (ψ)	0 [◦]
Velocidade Angular do Satélite em X (ω_x)	0 [rad/s]
Velocidade Angular do Satélite em Y (ω_y)	0.001016 [rad/s]
Velocidade Angular do Satélite em Z (ω_z)	0 [rad/s]
Torque da Roda em X (M_{Rx})	0 [N.m]
Torque da Roda em Y (M_{Ry})	0 [N.m]
Torque da Roda em Z (M_{Rz})	0 [N.m]
Velocidade Angular da Roda em X (ω_x)	0 [rad/s]
Velocidade Angular da Roda em Y (ω_y)	0.001016 [rad/s]
Velocidade Angular da Roda em Z (ω_z)	0 [rad/s]
Referência em X (ϕ)	0 [◦]
Referência em Y (θ)	0 [◦]
Referência em Z (ψ)	0 [◦]

5.5. Modelos

5.5.1. Satélite em Órbita Usando Ângulos de Euler

As equações cinemáticas foram implementadas em um modelo em C++ e MATLAB (m-file). O objetivo deste modelo é receber dados de entrada do usuário, como as características da órbita do satélite e, a partir disso, o programa calcula a posição do satélite durante um tempo, também especificado pelo usuário. A saída do programa é a posição do satélite em coordenadas terrestres (RI), X, Y e Z a partir da transformação dos ângulos de Euler das coordenadas orbitais (RO), x, y e z.

Figura 5.1 – Inicialização da simulação com HLA/RTI.



5.5.1.1. Dados inseridos pelo usuário

Tabela 5.5 – Condições Iniciais – Satélite em Órbita.

Altitude (h)	[m]
Inclinação (i)	[rad]
Ascensão Direta (omega)	[rad]
Tempo de simulação (t)	[s]

Nota: O usuário pode utilizar os valores da Tabela 5.1 como referência.

5.5.1.2. Constantes e Considerações

Tabela 5.6 – Constantes e Considerações.

Excentricidade (e)	0 (Órbita Circular)
Anomalia Média Inicial (u)	0
Argumento do Perigeu (ω)	0 [rad]
Constante Geo-Gravitacional Universal (G)	$3.986\times10^{15}\ \left[Km^3/s^2\right]$

5.5.1.3. Valores Calculados

Tabela 5.7 – Valores Calculados.

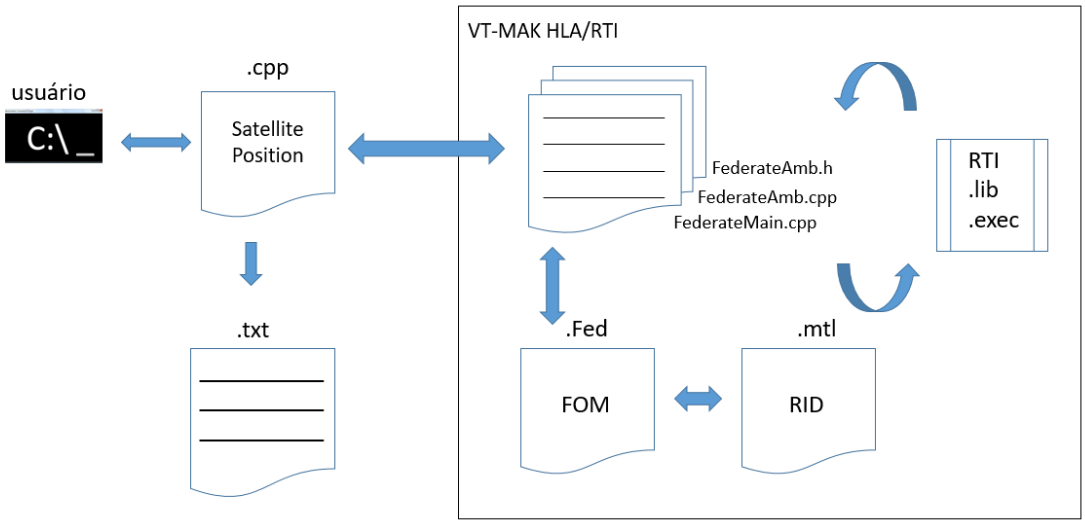
Anomalia Média (u)	Adimensional
Velocidade Orbital Média (ω_0)	[rad/s]
Período Orbital (T_0)	[s]
xo,yo e zo (posição RO)	[km]
X,Y e Z (posição RI)	[km]

O Modelo também foi verificado no MATLAB, através da simulação das equações implementadas no MATLAB e comparação das saídas do C++ (arquivo .txt) com as saídas do MATLAB.

5.5.1.4. Integração com HLA/RTI

O modelo foi integrado ao HLA/RTI, conforme diagrama apresentado na Figura 5.2 a seguir.

Figura 5.2 – Integração com HLA/RTI.

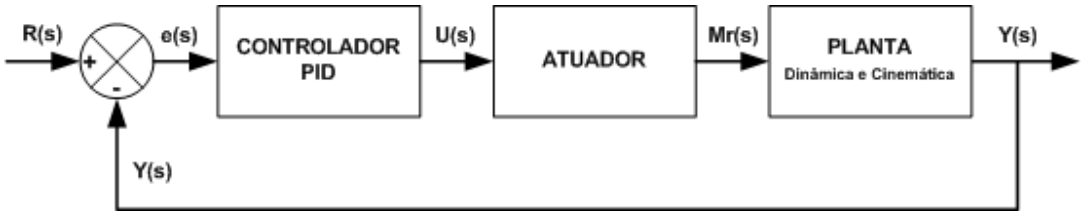


5.5.2. SCA da PMM (Amazônia-1)

Como descrito por Tagawa (2013), a mudança de atitude é obtida através de rodas de reação que variam suas velocidades angulares (ω_r) impondo um torque ao corpo do satélite (M_r). Esse torque induz uma variação na velocidade angular ω do satélite. Com base nos valores de ω, ω_re M_r é possível determinar a atitude do satélite em determinado momento em relação aos ângulos com os eixos do referencial orbital (ϕ, θ, ψ). Esses valores são comparados com valores de referência e a diferença é inserida em um controlador PID que interage com as rodas de reação fechando a malha de controle.

O sistema de controle de atitude (SCA) da PMM é representado na Figura 5.3.

Figura 5.3 - Diagrama de Blocos do SCA



O programa em C++ inicia-se com as condições iniciais apresentadas na Tabela 5.1. Em seguida, utilizando-se a modelagem das equações dinâmicas, são obtidas as acelerações angulares $\lfloor \omega_x \quad \omega_y \quad \omega_z \rfloor$.

Essas acelerações são integradas para se obter as velocidades angulares, $\lfloor \dot{\phi} \quad \dot{\theta} \quad \dot{\psi} \rfloor$.

As velocidades angulares $\lfloor \dot{\phi} \quad \dot{\theta} \quad \dot{\psi} \rfloor$ são inseridas nas equações cinemáticas e são integradas para se obter os ângulos de Euler $[\phi \quad \theta \quad \psi]$.

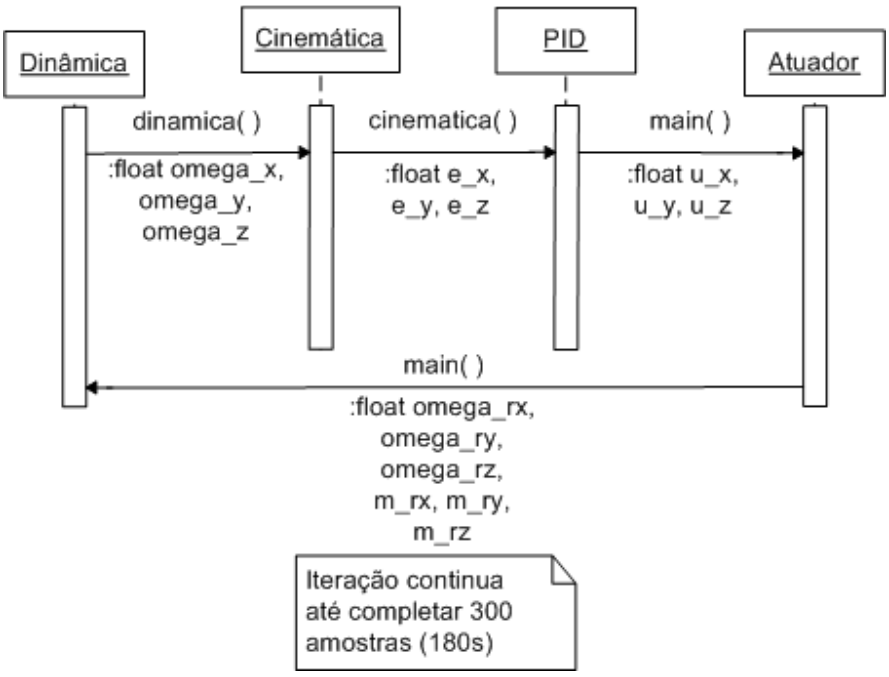
O sensor lê os ângulos $[\phi \quad \theta \quad \psi]$ e os envia para o controlador, que os compara com os de referência, quando estão diferentes, sinais de controle são enviados aos atuadores $\lfloor u_x \quad u_y \quad u_z \rfloor$.

Os atuadores responderão em acelerações angulares $\lfloor \dot{\omega}_{rx} \quad \dot{\omega}_{ry} \quad \dot{\omega}_{rz} \rfloor$, após integração, obtém-se as velocidades angulares $\lfloor \omega_{rx} \quad \omega_{ry} \quad \omega_{rz} \rfloor$ e, a partir das equações de (4.87), chega-se aos torques $\lfloor M_{rx} \quad M_{ry} \quad M_{rz} \rfloor$.

A simulação foi realizada utilizando o método *Ruge-Kutta* ODE4 com passo fixo de 0,6 s e *Stop-Time* em 300s. Os resultados obtidos são transportados para um arquivo *.txt*, que pode ser aberto e os gráficos plotados são comparados com os dados do MatLab/Simulink.

Os valores utilizados neste programa, estão apresentados nos subcapítulos 5.1, 5.2, 5.3 e 5.4.

Figura 5.4 – Sequência do SCA implementado em C++.



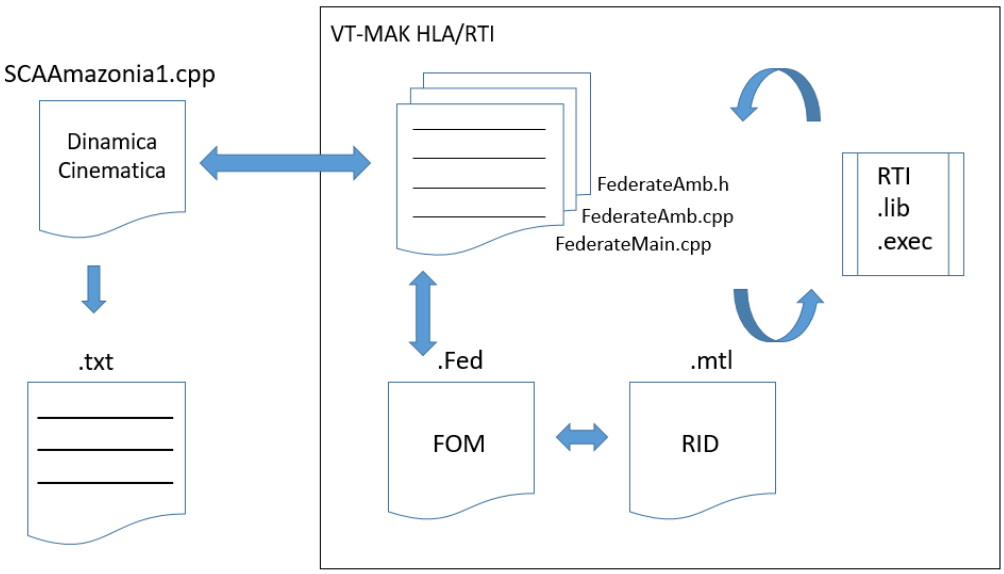
Fonte: Moraes (2017).

O modelo MatLab/Simulink foi implementado para comparação dos resultados obtidos no programa C++, conforme apresentado no capítulo 7. O modelo do SCA no MATLAB/Simulink está apresentado nas figuras do Apêndice A.

5.5.2.1. Integração com HLA/RTI

O modelo foi integrado ao HLA/RTI, conforme diagrama apresentado na Figura 5.4.

Figura 5.4 – Integração com HLA/RTI.



6 PROGRAMAS UTILIZADOS

Nesse trabalho foram utilizados três RTIs e um *software* da VT- MÄK que apresenta interface para utilização dos protocolos de simulação distribuídas DIS e HLA. Também uma análise mais superficial de mais duas RTIs *open-source* foi possível através da documentação disponível, não sendo possível a instalação e utilização devido a limitações de hardware.

Esse trabalho apresenta uma comparação entre esses *softwares* juntamente com o *software* MATLAB, que atualmente é conhecido e bastante utilizado na comunidade acadêmica, principalmente nas Engenharias. Sempre que possível, será utilizado um modelo aeronáutico.

As RTIs e *softwares* analisados foram:

- MÄK-RTI e VR-FORCES, *softwares* comerciais cedidos pela VT- MÄK, pelo período de dois anos;
- RTI da PITCH, chamado de pRTI, é um *software* comercial, com uma versão gratuita disponível no site e que limita a aprendizagem do HLA;
- RTI da PORTICO, *open-source* disponível no site da PORTICO para download;
- OpenRTI da Flightgear Project, *open-source* disponível na internet (analisando apenas a documentação);
- CERTI da ONERA (*Organisation Nationale pour la Recherche Aeronautique*) laboratório francês de aplicações aeroespaciais, *open-source* disponível na Internet (analisando apenas a documentação).

6.1. Estruturas Comuns às RTIs

As características comuns às RTIs analisadas estão apresentadas a seguir. Quando a RTI aqui utilizada não apresentar essa característica ou quando a informação não estava disponível na documentação, essa será mencionada no subcapítulo referente à RTI em questão.

6.1.1. Componentes Principais

Os *softwares* RTI são compostos dos seguintes *softwares*:

- *RtiExec* (*RTI Executive Process*): processa e administra a criação e destruição das execuções de federação. Existe um único e global *FedExec* para cada federação em execução. Administra múltiplas execuções da federação em uma rede.

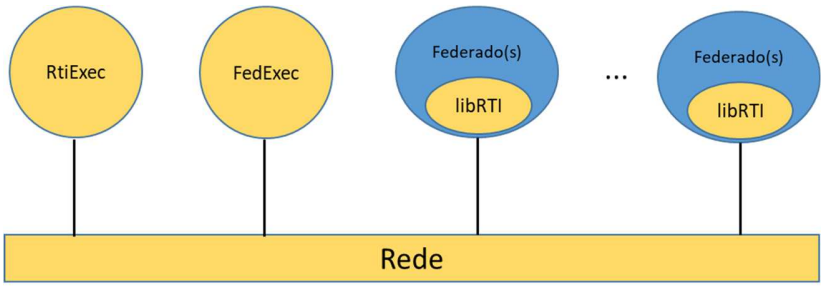
  O *RtiExec* é um processo global e conhecido. Cada aplicação se comunica com o *RtiExec* para iniciar os componentes RTI.

  A principal função do *RtiExec* é administrar a criação e destruição dos *FedExecs*. Outra função do *RtiExec* é garantir que cada *FedExec* apresenta um único nome.

- *FedExec* (*Federation Executive Process*): administra a federação, permitindo que os federados entrem e saiam, além de facilitar a troca de dados entre os federados participantes. Esse processo é criado quando o primeiro federado invoca ao serviço *createFederationExecution* para uma federação específica.

- *libRTI* (Biblioteca): é uma biblioteca em C++ que provê aos federados os serviços da especificação de interfaceamento HLA (*HLA Interface Specification*). Os federados utilizam a biblioteca *libRTI* para evocar os serviços do HLA. O *LibRTI* se comunica com o *RtiExec*, o *FedExec* e com outros federados.

Figura 6.1 – Componentes de uma RTI.



Fonte: Adaptado de DMSO (2000).

6.1.2. Estrutura das Simulações

Para a comunicação dos federados numa simulação RTI os arquivos a seguir necessariamente devem ser gerados como *.dlls* para os Federados:

- Arquivo *main* que estabelecerá o nome do Federado, cria e roda o Federado;
- Arquivo *ExampleCPPFederate* e respectivo *header* (C++);
- Arquivo *ExampleCPPFedAm* (*Ambassador*) e respectivo *header* (C++).

Estes arquivos contém as funções básicas que evocam os serviços da RTI.

Normalmente o compilador usado para gerar as *.dlls* são de C++, podendo para algumas RTIs utilizar-se compiladores de Java.

Além das *.dlls*, são necessários o *RTI.rid* e o *fom.fed*. O arquivo FOM apresenta a informação a cerca dos dados que serão transmitidos. O formato FED (*Federation Execution Data*) é o formato compilado necessário pela RTI.

Figura 6.2 – Componentes do Federado.

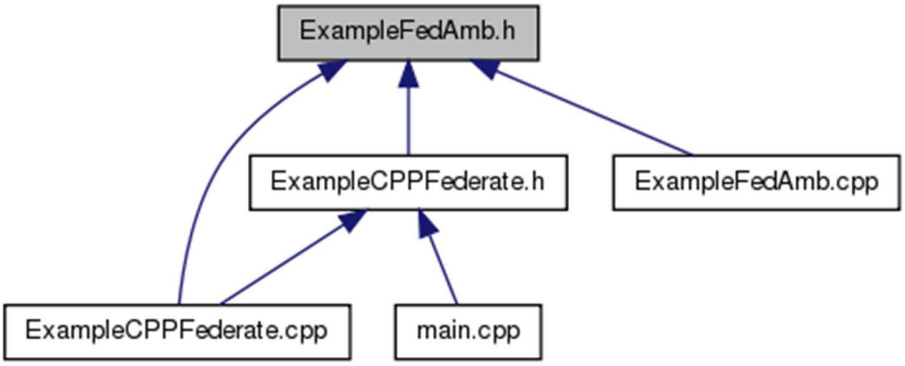
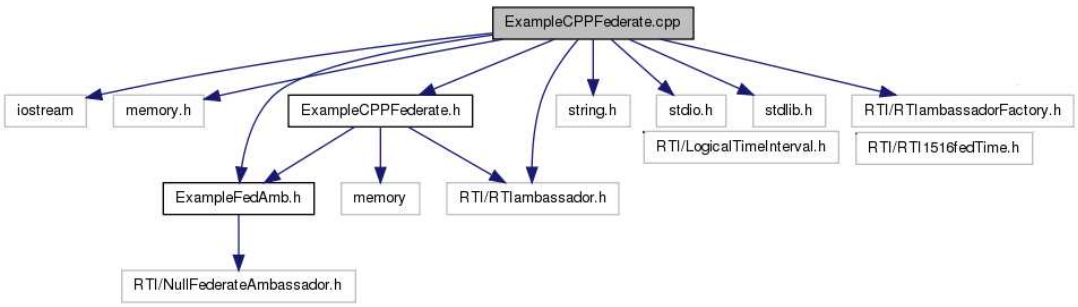


Figura 6.3 – Interações dos Componentes do Federado com os componentes da RTI.



Cada federado apresenta um SOM, que dita quais os dados estes federados devem enviar e receber. O SOM dos Federados faz parte do FOM e cada Federação deve apresentar um FOM.

6.1.3. Variáveis de Ambientes

As varáveis de ambiente que devem ser configuradas quando da utilização de uma RTI são:

- Genéricas

  $RTI_HOME: direcionando o *driver* para onde o executável da RTI se encontra;

$RTI_HOME/RTI.RID: direcionando o *driver* para onde os arquivos RID (*RTI Initialization Data*) se encontram.

Para os arquivos dos Federados, desenvolvidos em C++, deve-se incluir:

%RTI_HOME%\include: direcionando o *driver* para onde os arquivos que desempenham as funções do HLA se encontram.

%RTI_HOME%\lib: direcionando o *driver* de bibliotecas.

Algumas RTIs configuram automaticamente as variáveis de ambiente genéricas, outros requerem que os usuários os configurem. Na prática, porém, as RTIs que configuram automaticamente as variáveis de ambiente durante a instalação podem apresentar conflito caso mais de uma RTI for utilizada na máquina.

6.1.4. Serviços

A seguir estão listados os principais serviços utilizados nas simulações HLA:

a) Para criar a execução da Federação e se juntar a ela:

```
rti = RTIambassadorFactory.createRTIambassador()
rti.connect(federateAmbassador, IMMEDIATE, “MySettingsDesignator”)
rti.createFederationExecution(“MyFederation”, “MyFOM”)
rti.joinFederationExecution(“MyName”, “MyFederateType”,
“MyFederation”)
```

b) Para destruir a Federação e sair dela:

```
rti.resignFederationExecution(CANCEL_THEN_DELETE_THEN_DIVEST)
rti.destroyFederationExecution()
rti.disconnect()
```

Além dos serviços mínimos acima listados, os códigos dos Federados devem conter as seguintes variáveis de inicialização:

- *SyncPhase*: é uma varável do tipo *string*, usada pelo Federado para se sincronizar com a RTI;
- *Fomfile*: nome do arquivo que tem o modelo do objeto;

- *FederationName*: nome da Federação (deve ser o mesmo em todos os Federados);
- *FederateName*: nome do Federado;
- *RtiTimeResolution*: o tamanho do passo de tempo requerido pelo modelo. Quando não estabelecido o padrão (*default*) utilizado é 1.0 segundo, pode-se, no entanto, utilizar valores de milisegundos;
- *Object Name*: nome do objeto que está sendo compartilhado;
- *Object Attributes*: é uma propriedade do objeto que está sendo compartilhado;
- *Object Handle*: identificador único do objeto usado pelo Federado;
- *Attribute Handle*: Identificador único do Federado para o Atributo.
- *Attribute Value*: O valor que o atributo apresenta no momento.

6.1.5. Execução dos Federados

Em todos os casos estudados neste trabalho, os Federados são *.dlls* que podem ser chamadas através da barra de ferramenta do Windows ou podem ser rodadas através do duplo click sobre a *dll*.

Normalmente, as RTIs não apresentam interface gráfica, sendo as mesmas um conjunto de bibliotecas e arquivos chamados como variáveis de ambiente pelos arquivos do Federado.

Figura 6.4 – Federado Bounce (da VT-MÄK) chamado através da barra de Ferramentas do Windows.

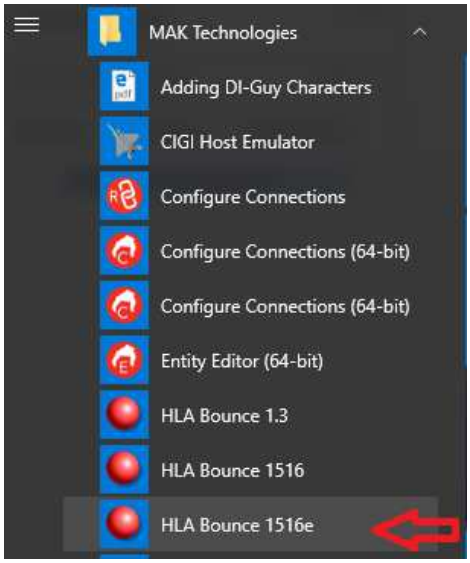
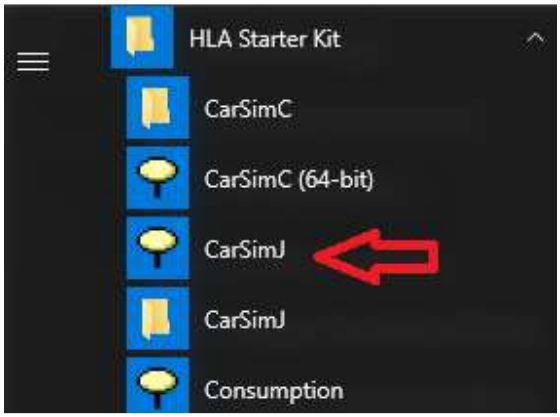


Figura 6.5 – Federado CarSimJ (da Pitch) chamado através da barra de Ferramentas do Windows.



6.2. Projetos de Código Aberto (*Open-Sources*)

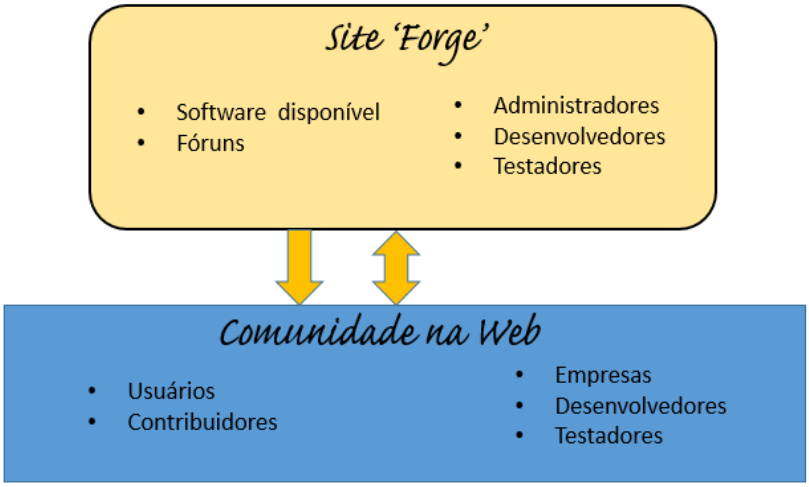
Um projeto *open-source* é estruturado conforme a comunidade de usuários que podem ser, além de simples usuários, desenvolvedores e contribuidores desse *software*.

Diferentemente dos *softwares* comerciais desenvolvidos por empresas, os usuários acompanham a evolução do *software* através de contribuições, seja por relatórios de problemas ou sugestões de melhorias ou até mesmo alterando o código, com o intuito de melhorá-lo.

Um projeto *open-source* normalmente não se limita a um país e pode ser usado na comunidade, inclusive por empresas.

A maioria dos *softwares open-source* estão disponíveis na *web* hospedados em um portal desenvolvido para o projeto chamado de *forge*. Nesse portal os usuários podem baixar os *softwares*, ferramentas e API relacionados, podem também fazer upload de códigos modificados. Para *upload* dos códigos, normalmente é necessário um cadastro incluindo informações pessoas, tais como nome e e-mail. Os códigos alterados pelos usuários serão avaliados pelos administradores da página *forge* e, usualmente o administrado do *forge* inclui o usuário como desenvolvedor. A Figura 6.6 mostra como funciona o processo de manutenção dos projetos de *software open-source*:

Figura 6.6 – Projeto *Open-Source*.



Os projetos *open-source* também disponibilizam fóruns em sites para que os usuários se ajudem e reportem bugs encontrados.

A contribuição dos usuários não é obrigatória, podendo apenas utilizar o *software*, sem que necessitem contribuir com o código.

Conforme Noulard et al (2006), um projeto *open-source* apresentará os seguintes *stakeholders*:

- Administradores do Projeto: pessoas ligadas diretamente ao projeto e que executam atividades administrativas neste. Estas atividades administrativas são, por exemplo: adicionar membros, remover membros, conceder privilégios a membros com respeito à utilização de ferramentas, e moderar e-*mails* e fóruns.

- Desenvolvedores: pessoas as quais tem acesso para escrever o código de um ou mais componentes de um *software* do projeto. Eles podem adicionar, remover ou modificar o *software*, corrigir ou reportar *bugs*, cuidar da documentação de liberação do *software* (*release*). Podem responder voluntariamente a *e-mails*. Num projeto, existe pelo menos um

desenvolvedor responsável pela liberação dos componentes de *software*, usualmente interno e ligado diretamente ao projeto.

- Contribuidores: são pessoas que utilizam os componentes do *software* e podem também contribuir concertando *bugs*, adicionando novas funcionalidades, pode auxiliar com a documentação e com traduções.

- Usuários: são pessoas que usam qualquer componente do projeto. Usuários fazem perguntas via e-mails ou fóruns. Podem se tornar desenvolvedores, quando demonstram interesse, após análise do administrador e do desenvolvedor responsáveis no projeto.

Os projetos *open-source* apresentam muitos benefícios, principalmente para a comunidade acadêmica, quando o objetivo principal é o aprendizado e normalmente não se dispõem de recursos financeiros para se manter os *softwares* comerciais. Porém a desvantagem é de não apresentarem suporte rápido, quando são encontrados problemas; e, normalmente, não apresentam uma interface gráfica desenvolvida de forma a tornar mais fácil a utilização. A documentação também pode não ser muito detalhada. *Softwares* comerciais normalmente apresentarão manuais para o usuário, bem como documentos detalhando os chamados *releases* (liberações) do *software*. Porém, o intuito de projetos *open-source* não é concorrer com os *softwares* comerciais, mas sim contribuir de alguma forma com a comunidade científica e acadêmica.

Neste trabalho foram usados cinco ambientes, sendo que um deles pertence a um projeto de código aberto.

6.3. *Softwares* da VT- MÄK

A empresa VT-MÄK forneceu os *softwares* MÄK-RTI e VR-FORCES para utilização acadêmica por dois anos.

6.3.1. RTI da VT- MÄK (MÄK-RTI)

O software MÄK-RTI foi verificado e aprovado pelo DMSO (*Simulation Coordination Office*) como totalmente compatível com o HLA, nas versões 1,3 e IEEE1516-2010, sendo todos os serviços verificados como implementados (VT-MÄK-2012).

O MÄK-RTI pode ser baixado gratuitamente para utilização de até dois federados.

6.3.2. VR-FORCES

O *software* VR-FORCES tem dupla funcionalidade, pois permite a simulação de diversos objetos modelados matematicamente além da visualização gráfica destes em ambientes também modelados.

A característica do VR-FORCES de implementar cálculos de engenharia, o classifica como um CGF (*Computer Generated Forces*), além dessa característica, este *software* apresenta uma interface de visualização gráfica (GUI) muito rica, que possibilita a observação da interação entre os objetos simulados e o ambiente.

Figura 6.7 – Visualização Gráfica de Aeronaves Simuladas em solo.

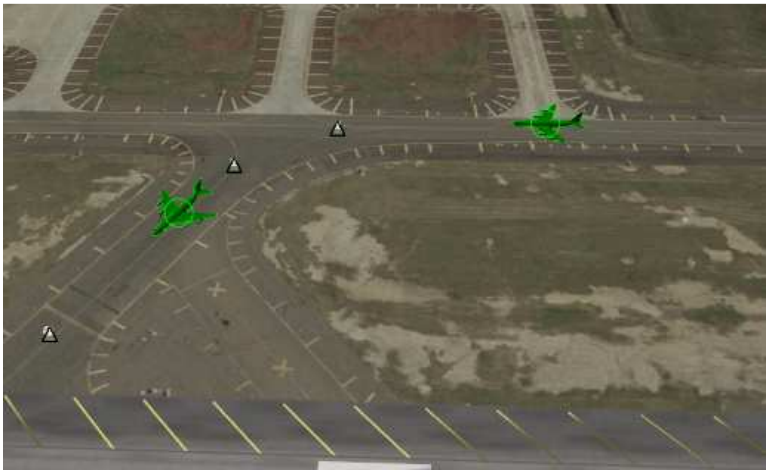
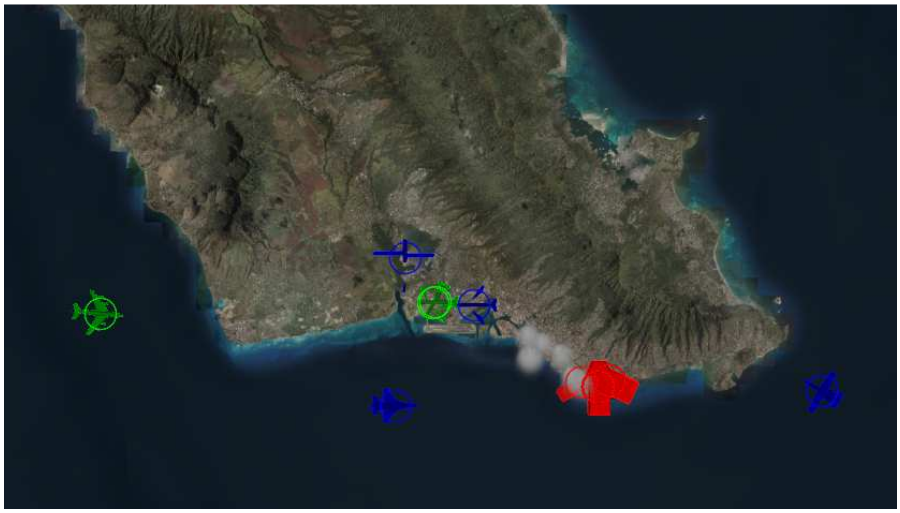


Figura 6.8 – Visualização Gráfica de Aeronaves Simuladas em voo.

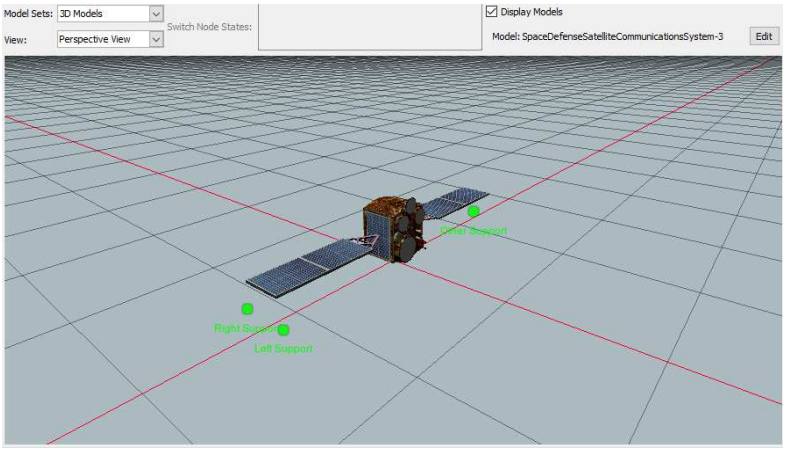


As simulações rodadas no VR-FORCES podem ser no padrão DIS ou HLA, bastando para isso que o usuário selecione o padrão que deseja utilizar no menu de inicialização do programa.

O protocolo DIS já é implementado nos objetos que iram interagir; já para as simulações em HLA, se faz necessário a utilização da RTI.

Uma terceira funcionalidade o torna mais completo. O VR-FORCES permite que usuários insiram seus próprios modelos através de um API, escrito em C++, ou simplesmente alterando as características através da utilização de dois aplicativos que vem junto com o VR-FORCES, chamados de *Simulation Object Editor* e *OPD Editor*.

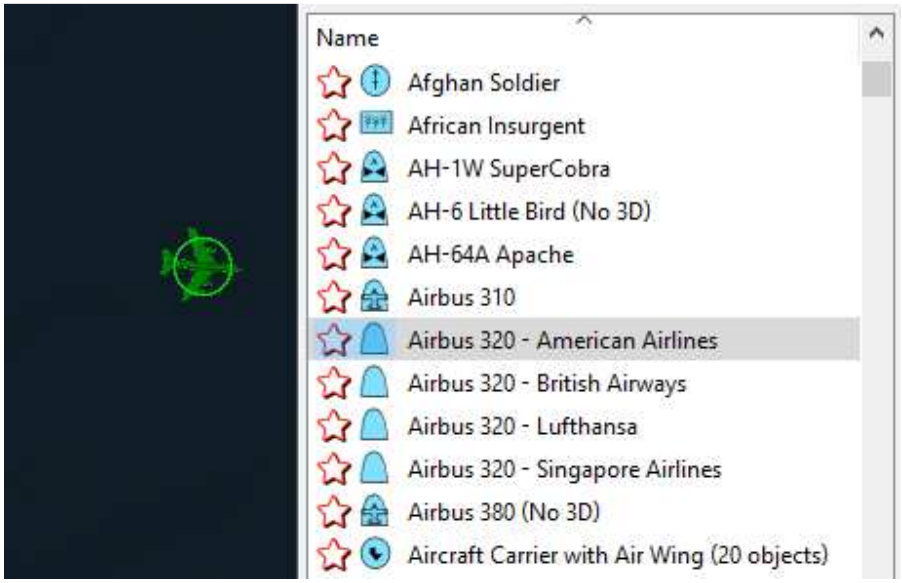
Figura 6.9 – Customização de um satélite via o *software Simulation Object Editor*.



O VR-FORCES é muito utilizado para missões táticas militares e apresenta muitos objetos já modelados, tais como aeronaves, helicópteros, submarinos destroieres. Esses modelos foram feitos a partir de informações públicas.

Por exemplo, a aeronave A-320 da fabricante Airbus é modelada com dados de dimensões, velocidade de cruzeiro, altitude de cruzeiro, MACH máximo entre outras informações a partir de dados públicos, disponíveis em manuais operacionais.

Figura 6.10 – A Aeronave A-320 apresenta quatro possíveis modelos.



6.3.3. Instalação

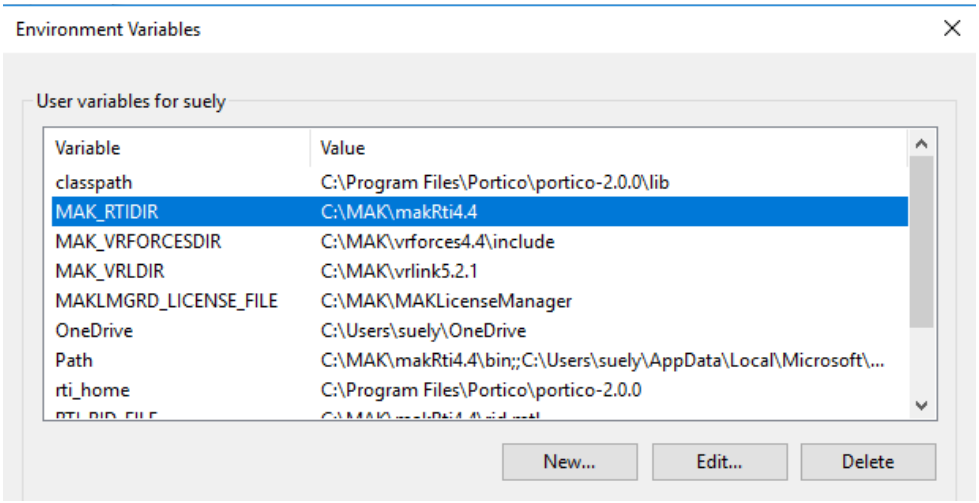
Para instalação das versões comerciais dos *softwares* da VT-MÄK, é necessária uma série de procedimentos.

Deve-se enviar à VT-MÄK um arquivo que contém informações do usuário e da máquina que será utilizada. Solicitam-se o *mach number*, *hostname*, *ethernet address* da máquina para controle.

A VT-MÄK então envia primeiramente um arquivo *.lic* para controle das licenças; este contém a data em que a licença deve expirar. A instalação de um *software* chamado de VT-MÄK *LicenseManager* é necessária para controle das licenças. É enviado então o *link* para baixar os *softwares*.

Deve-se criar e configurar as variáveis de ambiente para os programas, com os respectivos caminhos de *driver*.

Figura 6.11 – Configuração das Variáveis de Ambiente.



6.3.4. Limitações Computacionais

Segundo a documentação da VT-MÄK, não existem limitações de sistemas operacionais para Windows e Linux.

Tanto a RTI quanto o VR-FORCES foram testados no Windows 10, sem apresentarem problemas.

Porém, para implementação de APIs no VR-FORCES versão 4.4 existem as seguintes limitações:

- Visual Studio C++ deve ser especificamente o 2010 ou 2013;
- Não aceita programações em Java;
- O *software* VR-Link, também da VT-MÄK deve estar instalado e deve ser versão 5.2 para o VR-FORCES.

O VR-FORCES também requer placa gráfica NVIDIA *High Performance* para que a visualização das simulações em 3-D ou 2-D sejam adequadas.

6.3.5. Interface DIS

O *software* VR-FORCES apresenta interface DIS para os modelos existentes e esta pode ser customizada para outros modelos através dos *softwares* de customização do VR-FORCES ou da implementação de um API.

6.3.6. Componentes da MÄK-RTI

A MÄK-RTI é composta dos seguintes componentes, conforme composição básica das RTIs:

- RTI *Library*: consistente com o *libRTI* (Biblioteca) em formato de DLL.
- *Rtiexec*: provê os serviços de HLA para a execução da federação.

Além desses, o RTI da VT-MÄK provê outros *softwares*:

- RTI *Forwarder*: administra conexões TCP, gerenciando mensagens numa rede WAN.
- RTI *Assistant*: é uma interface gráfica para auxiliar os usuários a configurar as federações e visualizar o estado destas.

6.3.7. Arquivos necessários para os Federados

Para implementação dos Federados, os arquivos necessários são os mesmos listados no subcapítulo 6.1.2.

6.3.8. Implementação HLA

A documentação e os exemplos pertencentes e disponíveis na RTI da VT-MÄK demonstraram que a RTI é compatível com o padrão HLA. As características que devem ser contempladas conforme apresentado nos subcapítulos 6.1.3 e 6.1.4 estão implementadas.

6.3.9. Documentação

A VT-MÄK apresenta uma vasta documentação incluindo manuais para o usuário, guia de usuário e exemplos para simulação.

Porém, a implementação da API no VR-FORCES não está muito detalhada, sendo necessário o suporte da VT-MÄK para se avançar no processo.

O processo de instalação também não é tão trivial quanto outros *software*s utilizados neste trabalho.

A documentação da VT-MÄK, apesar de vasta, é limitada aos usuários que são clientes. Não foi possível encontrar na Internet muita documentação sobre a utilização dos *softwares* de VT-MÄK.

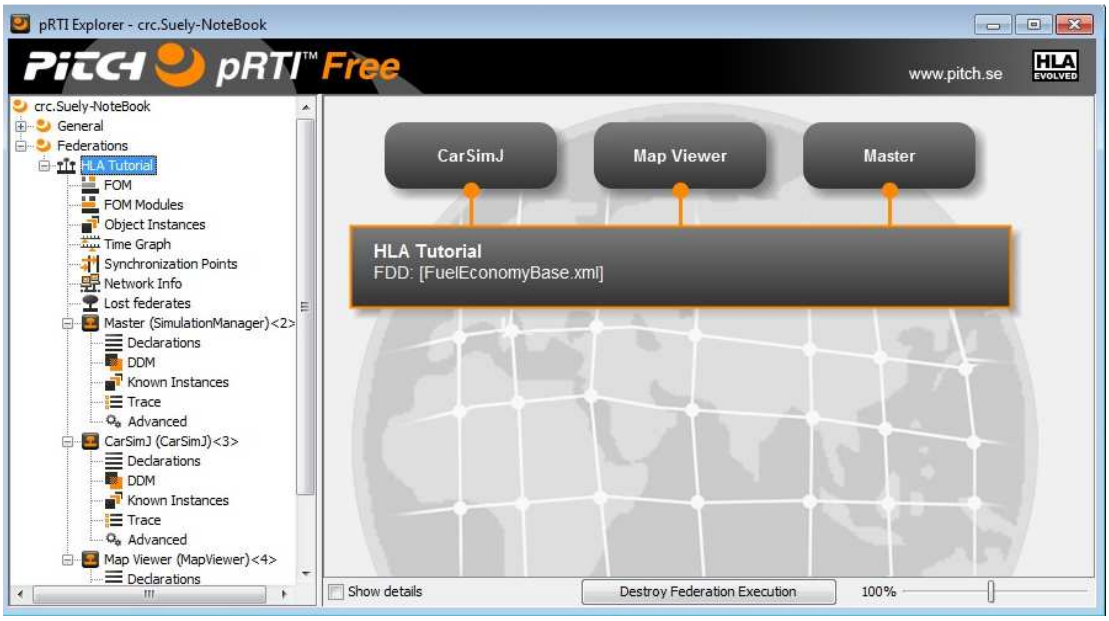
6.4. RTI da PITCH (pRTI)

A PITCH apresenta uma página na *web* para *download* de um kit para estudos do HLA. Este kit é composto de três *softwares*:

- pRTI *free* – uma versão limitada e gratuita do RTI da PITCH;
- Visual OMT *free* – uma versão limitada e gratuita de uma interface gráfica para visualizar as simulações em HLA;
- Um HLA *Starter kit* – *kit* composto de exemplos para utilização de simulações HLA.

A RTI da PITCH é a que apresenta melhor interface gráfica, sendo possível visualizar em um ambiente os Federados de uma Federação e o FOM.

Figura 6.12 – Visualização gráfica da pRTI.



Nota: Na Figura 6.12 acima, os objetos *CarSimJ*, *Map Viewer* e *Master*, são os Federados.

6.4.1. Instalação

Foi possível instalar os três *softwares* utilizando a documentação disponibilizada no *site*.

Figura 6.13 – Instalação da pRTI.

Os programas apresentam interface para auxiliar o usuário durante o processo.

As configurações do *software* são selecionadas durante a instalação, não sendo necessária a inclusão de variáveis de ambiente, pois é esta pode ser feita automaticamente.

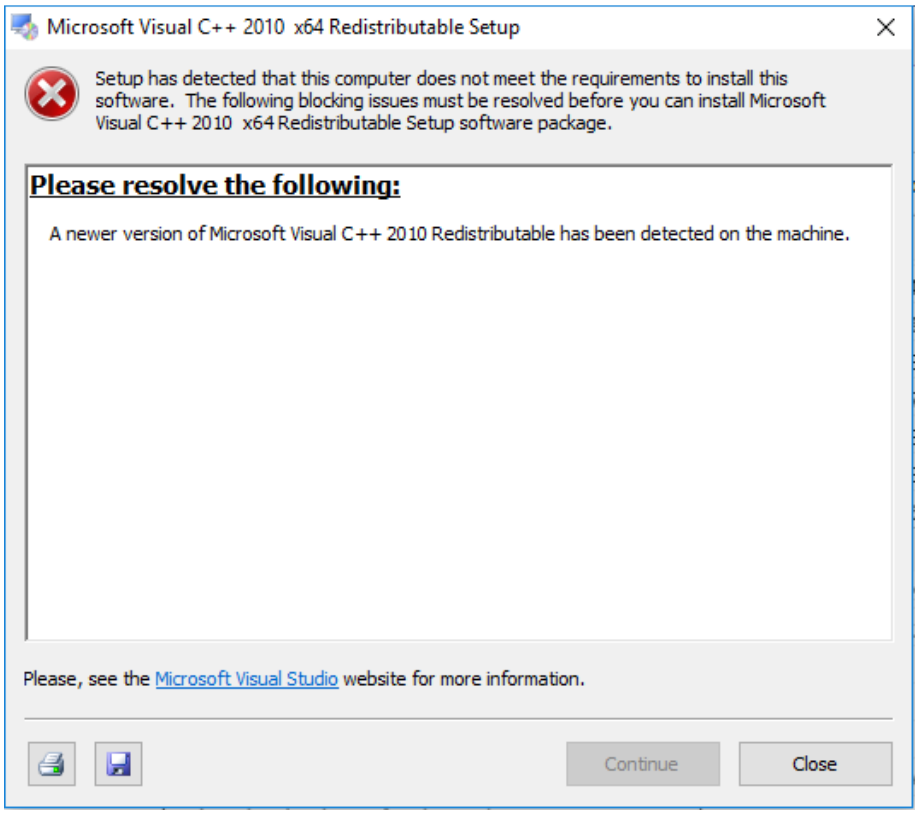
6.4.2. Limitações Computacionais

Segundo a documentação da PITCH, não existem limitações de sistemas operacionais para Windows e Linux.

Os *softwares* foram rodados em duas máquinas, uma com Windows 10 e outra com Windows 7.

Porém a RTI é compatível com Visual Studio C++ 2010. A principal limitação deste *software* é que se houver versão de Visual Studio C++ posteriores a 2010 instaladas na mesma máquina, a utilização do *software* fica impossibilitada.

Figura 6.14 – Erro durante a instalação da pRTI.



É possível também desenvolvimento utilizando-se Java.

6.4.3. Interface DIS

Os *softwares* da PITCH utilizados neste trabalho, não apresentam interface DIS.

6.4.4. Componentes da pRTI

A RTI da PITCH é composta de:

- Um *LRC*, que corresponde a uma biblioteca local (*libRTI*), instalada em cada computador com um Federado. Para federados em C++, corresponde

a arquivos de extensão *dll* ou *so*. Para Federados escritos em Java, corresponde a arquivos de extensão *jar*.

- Um componente central (*CRC*), que corresponde a uma aplicação para coordenar a execução da Federação e a entrada de Federados. O CRC é um programa que precisa ser inicializado em um computador com um endereço conhecido. É possível a utilização de vários Federados em um único computador rodando junto com o CRC.

6.4.5. Arquivos necessários para os Federados

Para implementação dos Federados, os arquivos necessários são os mesmos listados no subcapítulo 6.1.2.

6.4.6. Implementação HLA

A documentação e os exemplos pertencentes e disponíveis na pRTI demonstraram que a RTI é compatível com o padrão HLA. As características que devem ser contempladas conforme apresentado nos subcapítulos 6.1.3 e 6.1.4 estão implementadas.

6.4.7. Documentação

É possível encontrar documentação sobre a RTI da PITCH na Internet, porém essa documentação é limitada à utilização da RTI gratuito.

6.5. RTI da PORTICO

A RTI da PORTICO faz parte de um projeto de código aberto (*open-source*), podendo ser baixado gratuitamente e pode ser redistribuído. O usuário também pode modificar o código e submetê-lo para análise.

A PORTICO apresenta as implementações necessárias ao padrão HLA e foi especialmente desenvolvido para a utilização em simulações e treinamento.

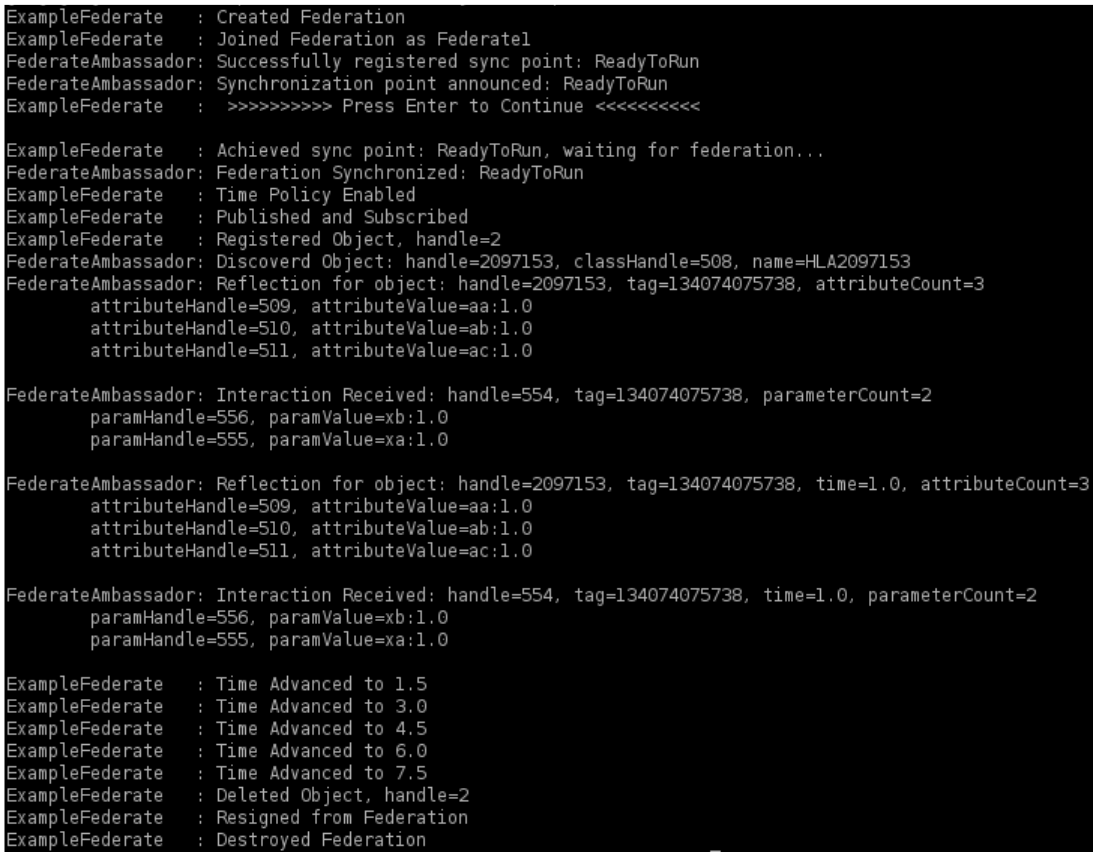
O site *forge* da PORTICO se encontra no *link*:

http://timpokorny.github.io/public/index.html

A PORTICO não apresenta uma interface gráfica com o usuário, sendo que a utilização do mesmo é feita através da chamada das variáveis de ambientes nos arquivos *cpp* (C ou C++) durante a compilação.

A saída de dados da PORTICO é visualizada através do CMD. A figura a seguir mostra a saída de dados após a compilação de um exemplo usando PORTICO. O nome do Federado é Federado1.

Figura 6.15 – Compilação do Federado1.



6.5.1. Instalação

Para instalação da PORTICO, basta o usuário acessar a página onde se encontra o código fonte e baixá-lo. O código virá com o executável para instalação.

6.5.2. Limitações Computacionais

Segundo a documentação do *software*, o PORTICO é compatível com os seguintes sistemas operações e compiladores:

- Windows 7, 8 e 10 (32 bits e 64 bits);
- *Visual Studio* 8, 9 e 10;
- Java 8u66;
- Linux: Ubutun 12.04/14.04 e CentOS 6.5 (64 bits);
- MAC OSX: OSX 10.10+ (64bits);

6.5.3. Interface DIS

A PORTICO não apresenta implementações para o padrão DIS.

6.5.4. Componentes da PORTICO

A PORTICO apresenta o executável e o conjunto de bibliotecas semelhante ao apresentado no subcapítulo 6.1.1.

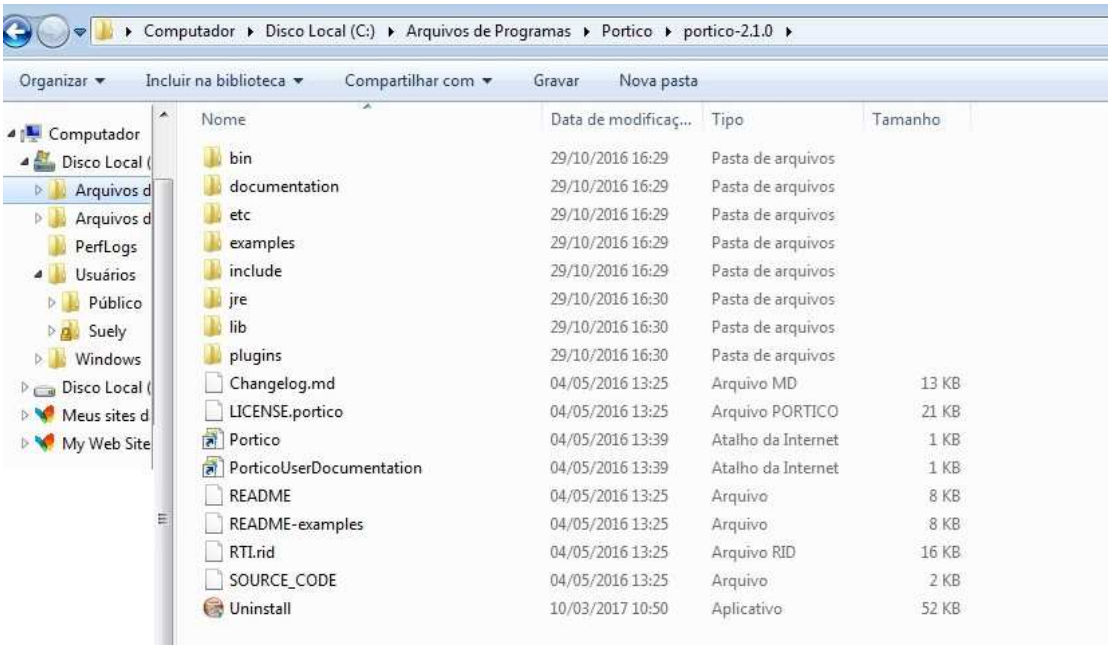
6.5.5. Arquivos necessários para os Federados

Para implementação dos Federados, os arquivos necessários são os mesmos listados no subcapítulo 6.1.2.

6.5.6. Implementação HLA

A documentação e os exemplos pertencentes e disponíveis, demonstraram que a RTI é compatível com o padrão HLA. As características que devem ser contempladas conforme apresentado nos subcapítulos 6.1.3 e 6.1.4 estão implementadas.

Figura 6.16 – Estrutura de Pastas do PORTICO.



6.5.7. Documentação

A documentação é bastante limitada. Não existem manuais para usuários, mas apenas páginas na Internet orientando o usuário a como utilizar a RTI e como implementar o Federado.

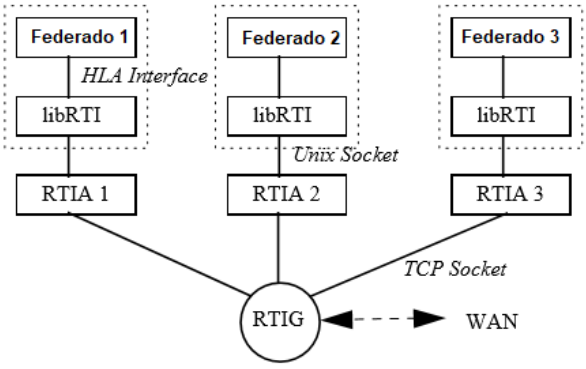
A PORTICO vem com um exemplo bem simples e útil que facilita na primeira implementação de uma simulação utilizando HLA.

6.6. RTI CERTI da ONERA

A RTI CERTI faz parte de um projeto de código aberto, desenvolvido pela ONERA, que é um Laboratório Francês de Tecnologias Aeroespaciais.

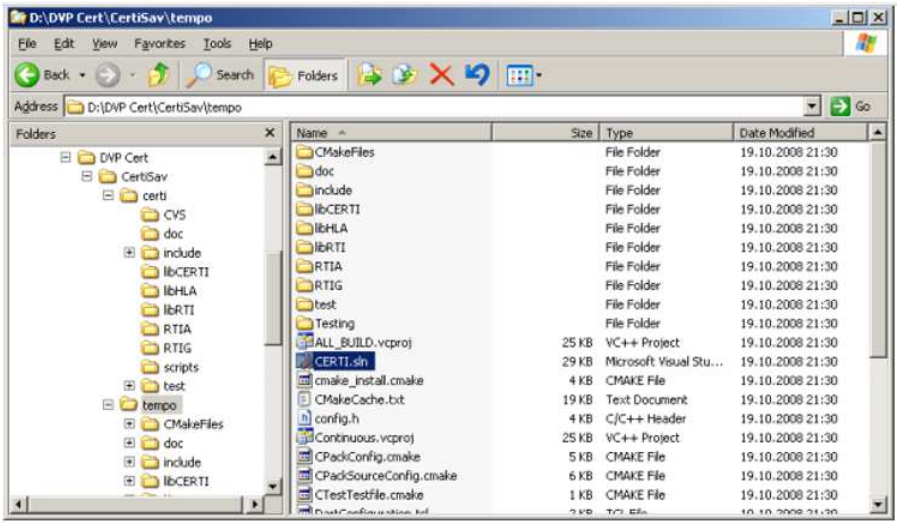
A CERTI apresenta estrutura muito semelhante as das outras RTIs aqui apresentados, sendo compatível para desenvolvimento com C++ ou Java.

Figura 6.17 – Estrutura do CERTI.



Fonte: adaptado de Noulard et al, 2010.

Figura 6.18 – Estrutura de arquivos da CERTI.



Fonte: Manual de Instalação, CERTI, 2014.

6.7. OPEN HLA (oh-la)

A RTI OPEN HLA pertence também a um projeto de código aberto. Sua documentação é bastante limitada, sendo bem menos utilizado que a PORTICO.

A última atualização do *software* foi realizada em 2007.

O *software* pode ser baixado do link: https://sourceforge.net/projects/ohla/

A RTI OPEN HLA não foi analisada neste trabalho, sendo a sua documentação disponível não suficiente para análise.

7 SIMULAÇÕES

Os subcapítulos a seguir, apresentam as configurações das simulações, os objetos simulados e comentários acerca dos resultados.

7.1. Cenário com Aeronaves

Tabela 7.1 – Configuração da Simulação de Aeronaves.

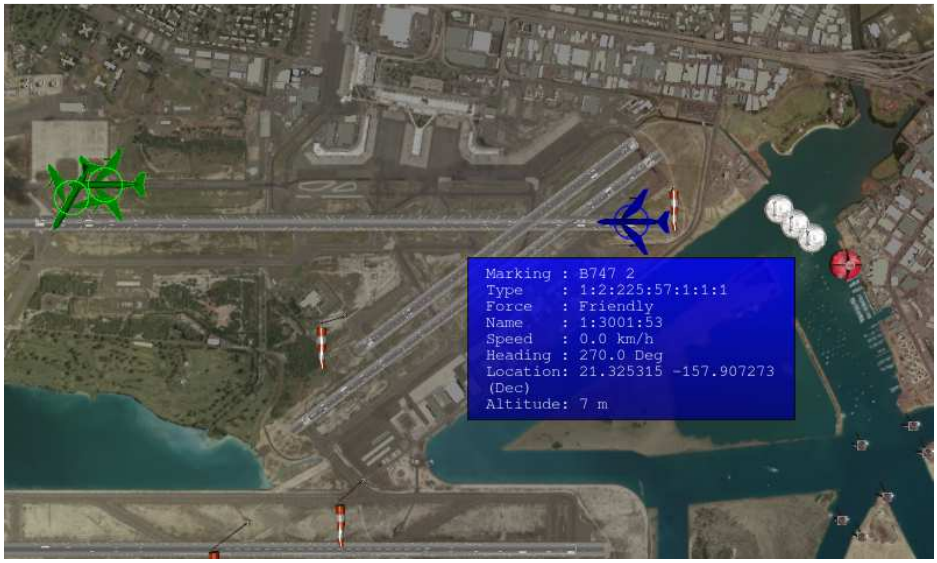
Hardwares	*Notebook Samsung Electronics*. Processador: Intel® Core™ i7-4510U CPU@2.00GHz. Memória Instalada: 8.00GHz.
Softwares	MÄK -RTI VR-FORCES
Sistema Operacional	Windows 10
Padrão	HLA 1516

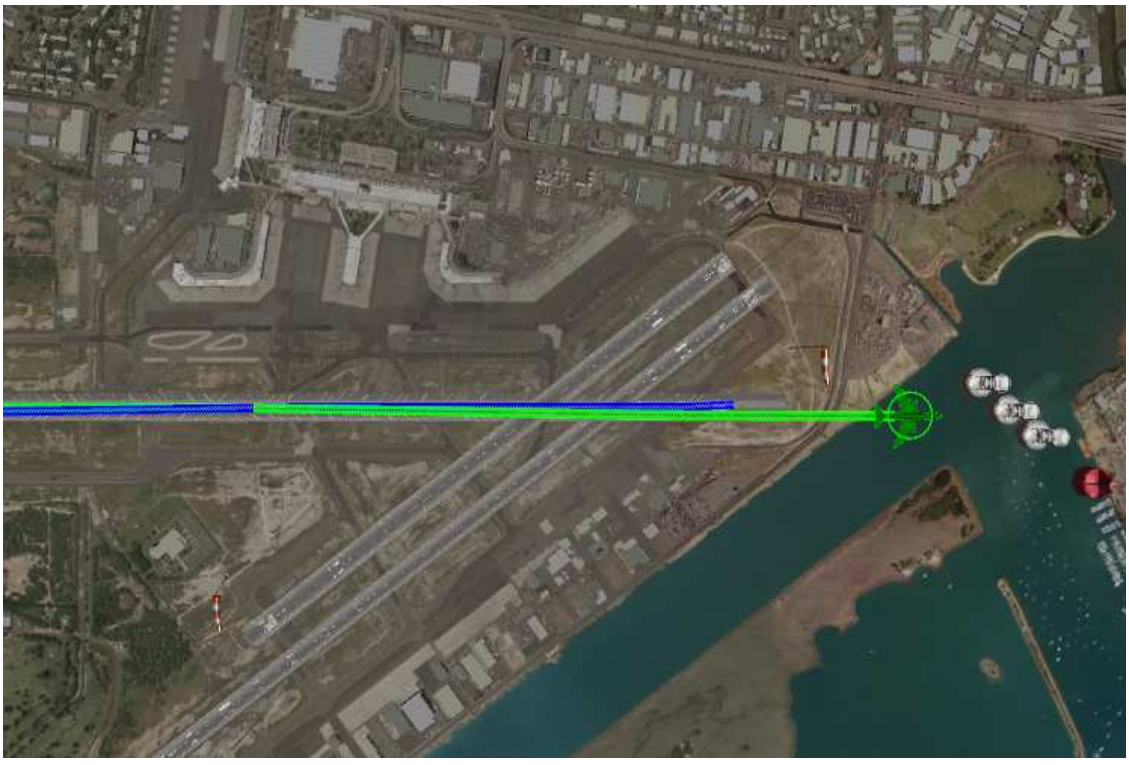
Descrição: O cenário contém três aeronaves no aeroporto de uma ilha no Havaí, as aeronaves são dois Boeing B747 e um Boeing B707. As aeronaves devem decolar sem que haja colisão, para isso elas seguem um plano estabelecido e os modelos dessas aeronaves.

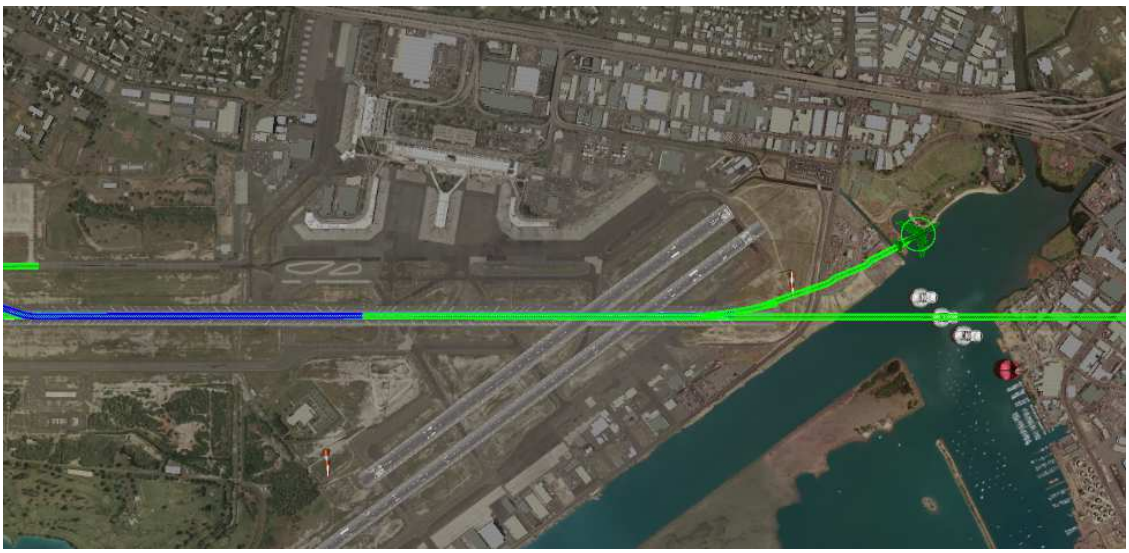
Figura 7.1 – Aeronaves B747 e B707.

Resultados: A primeira aeronave a decolar é o Boeing 707 (azul ou aeronave #3), a mesma utiliza a pista 8 (à frente desta). Em seguida a aeronave Boeing 747 número 2 decola e, por fim, a aeronave B747 decola, conforme mostra a sequência de Figuras 7.2 a seguir.

Figura 7.2 – Sequência das Aeronaves B747 e B707 Taxiando e Decolando.







7.2. Cenário com Veículos Militares

Tabela 7.2 – Configuração da Simulação de veículos militares.

Hardwares	*Notebook Samsung Electronics.* Processador: Intel® Core™ i7-4510U CPU@2.00GHz. Memória Instalada: 8.00GHz.
Softwares	MÄK -RTI VR-FORCES
Sistema Operacional	Windows 10
Padrão	DIS

Descrição: Cinco veículos militares seguem um plano de seguir em uma rodovia com velocidade constante. Três caminhões do tipo: M1028 FMTV (*Cargo Truck*) e dois tanques do tipo: *Leopard*.

Resultados: Os veículos seguem as rotas, conforme planejado.

Nota: O objetivo desta simulação é apenas verificar uma utilização com o protocolo DIS.

Figura 7.3 – Simulação Utilizando DIS.

7.3. Simulação de um Satélite em Órbita

Tabela 7.3 – Configuração da Simulação de Satélite em Órbita.

Hardwares	*Notebook Samsung Electronics.* Processador: Intel® Core™ i7-4510U CPU@2.00GHz. Memória Instalada: 8.00GHz.
Softwares	MÄK -RTI VR-FORCES
Sistema Operacional	Windows 10
Padrão	HLA 1516

Descrição: Nesta simulação foi utilizado um Satélite que deve seguir uma Órbita geoestacionária (GEO). Para tanto, foi traçada uma órbita localizada a 35,786 km de altura no Plano do Equador, seguindo a direção de rotação da Terra. Essa órbita tem um período de 86.164 segundos (24 horas), a velocidade na órbita é de 3 km/s.

Para essa simulação foi estabelecido um plano ao satélite contendo as tarefas:

- Seguir uma rota, nesse caso a órbita;
- Manter a velocidade (3 km/s);
- E enviar uma mensagem com o texto “Bateria Carregando” durante os dois primeiros segundos de simulação.

Esse modelo foi apresentado no SAE-2016 (Romeiro e Souza, 2016).

Resultados: O Satélite seguiu o plano, seguiu a rota e enviou mensagens.

Figura 7.4 – Satélite e Órbita GEO.

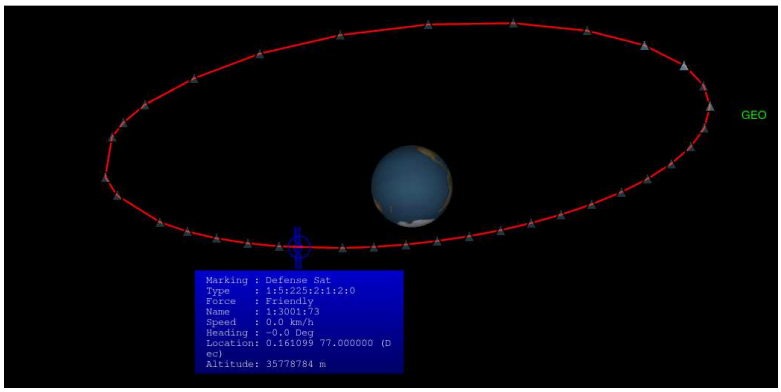
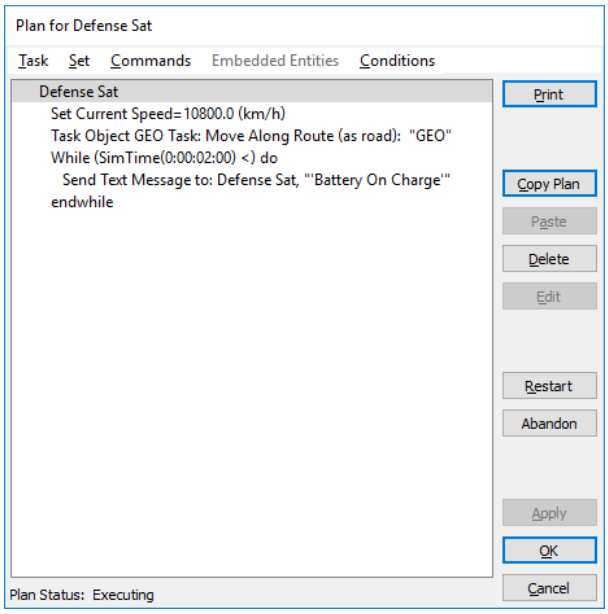


Figura 7.5 – Tarefas atribuídas ao Satélite.



7.4. Simulação de Consumo de Combustível para Três automóveis

Tabela 7.4 – Configuração da Simulação de Consumo de Combustível.

Hardwares	*Notebook Samsung Electronics.* Processador: Intel® CeleronB800 @ 1.50GHz. Memória Instalada: 2.00GHz.
Softwares	pRTI
Sistema Operacional	Windows 7
Padrão	HLA 1516

Descrição: O exemplo rodado utilizando-se a pRTI apresenta um cenário com três Federados:

- *Master*: é o Federado que gerencia a inicialização da simulação e o fim (*stop*).
- *Mapper Viewer*: é o Federado responsável por apresentar os Federados do CarJ sobre o mapa.
- *CarJ*: consiste de um conjunto de modelos de consumo de combustível, contento quatro modelos. Cada carro apresenta um modelo próprio de consumo, dependente do tipo de carro e do tipo de combustível. O modelo dos carros está apresentado na Tabela 7.5.

Tabela 7.5 – Modelo de Carros do Federado CarJ.

Carro	Combustível
Falcon	Etanol Flexível
Lightning McQ	Gás Natural
Monza 77	Gasolina
Quantum	Diesel

Resultados: A simulação começa com a inicialização do Federado.

Figura 7.6 – Inicialização do Federado *Master*.

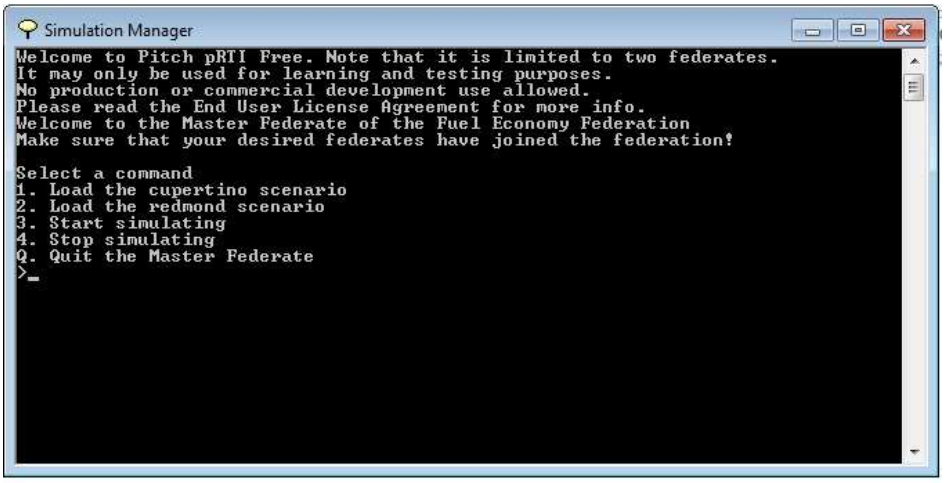
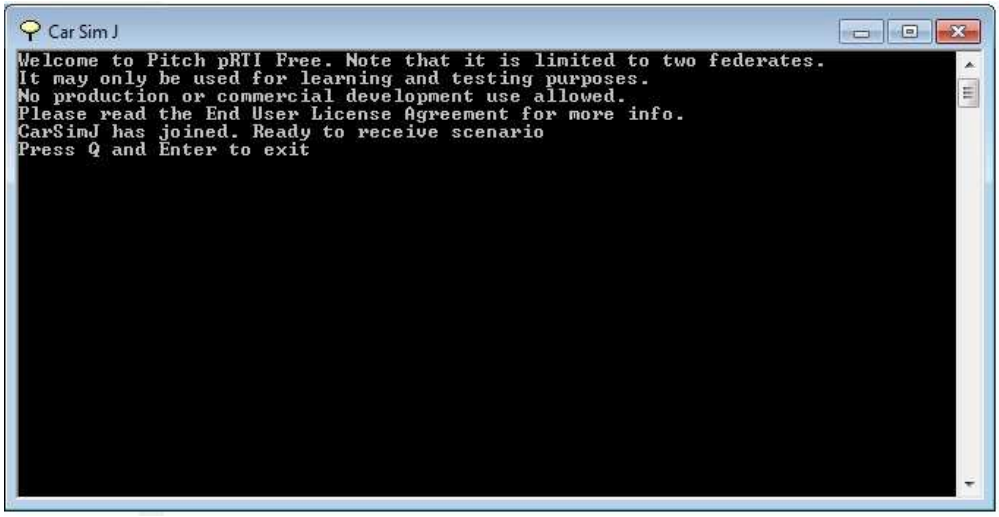


Figura 7.7 – Inicialização do Federado *CarJ*.



Ao escolher o cenário no *Master*, o usuário inicializa o Federado *Mapper*.
O usuário então deve entrar com o valor de combustível inicial dos carros.
E ao selecionar a opção 3 (*start*), o usuário inicia a simulação.

Os carros percorrem o cenário num caminho predeterminado (uma reta até pacifica), consumindo o combustível, conforme o modelo de cada carro. O modelo de consumo de combustível utilizado por cada carro é bem simples, sendo considerado um valor constante de consumo, dependente do modelo do carro e do tipo de combustível.

Figura 7.8 – Simulação do Modelo na pRTI.

7.5. Simulação do *Satellite Position* (Modelo C++ e MATLAB .m)

Tabela 7.6 – Configuração da Simulação *Satellite Position*.

Hardwares	*Notebook Samsung Electronics.* Processador: Intel® Core™ i7-4510U CPU@2.00GHz. Memória Instalada: 8.00GHz.
Softwares	MÄK -RTI MATLAB
Sistema Operacional	Windows 10
Padrão	HLA 1.3

Descrição: O modelo apresenta o desenvolvimento das equações cinemáticas para cálculo da posição de um satélite, a partir de certos parâmetros de entrada, conforme descrito no subcapítulo 5.5.1. O federado consiste do programa escrito em C++, chamado de SATPOS. Quando o programa é rodado, ele chama a RTI que executa os serviços da simulação Federação.

Resultados: Na simulação em questão, o usuário entra com os seguintes dados apresentados na Tabela 7.7.

Os resultados calculados no programa C++ e no MATLAB, estão plotados nas Figura 7.10, Figura 7.11,

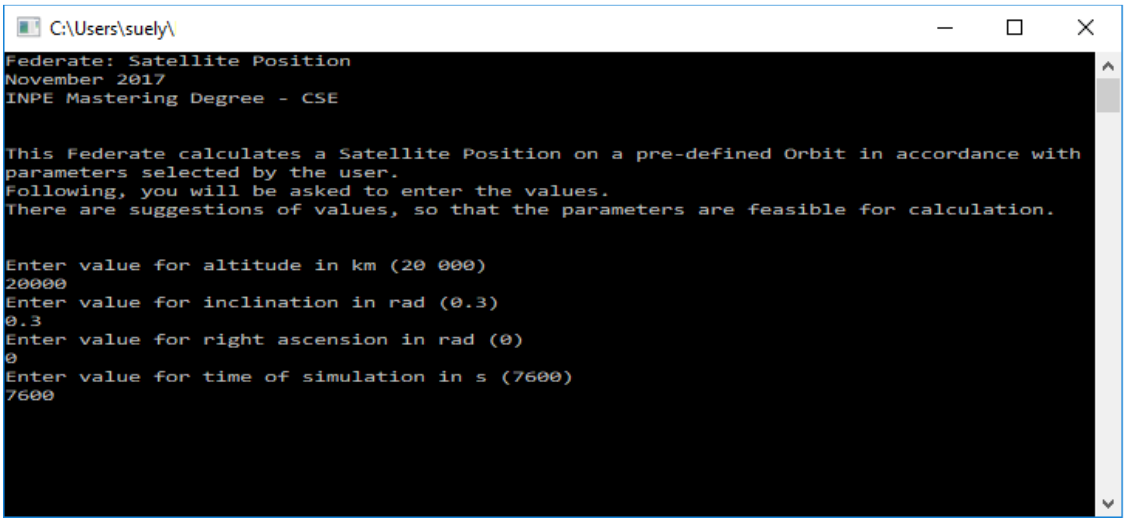
Figura 7.12 e Figura 7.13.

Nota: A utilização do programa MATLAB foi apenas para verificação dos dados, a integração com a RTI foi feita através do arquivo C++.

Tabela 7.7 – Dados de Entrada.

Parâmetro	Valor
Altitude (h)	2000 [km]
Inclinação (i)	0.3 [rad] (17º)
Ascensão direta (omega=Ω)	0 [rad]
Tempo de simulação (t)	7600 [s]

Figura 7.9 – Inicialização do Federado *Satellite Position*.



A saída em *txt* é lida pelo *.m* (arquivo MATLAB) e são plotados para verificação dos dados.

Figura 7.10 – Variação da Posição do Satélite em X, ao longo do Tempo.

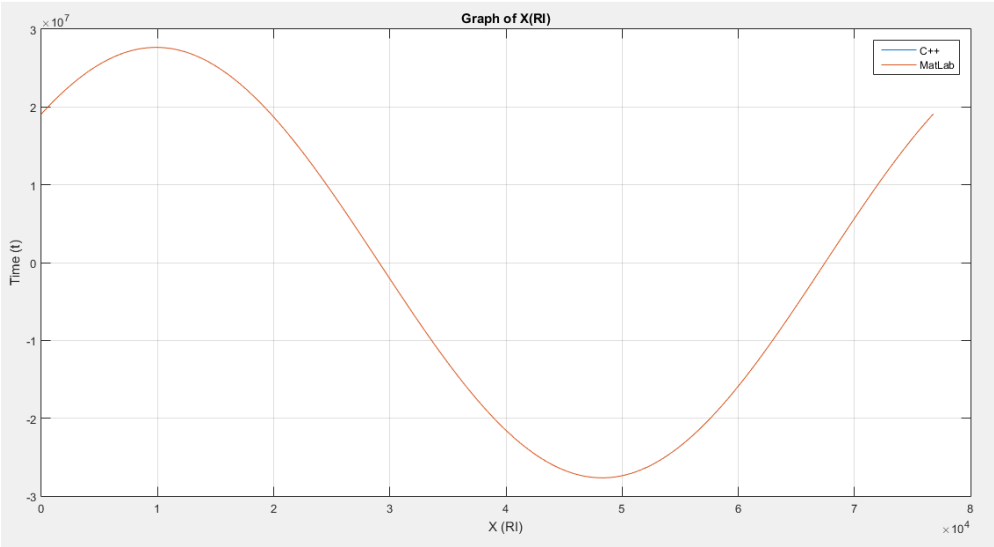


Figura 7.11 – Variação da Posição do Satélite em Y, ao longo do Tempo.

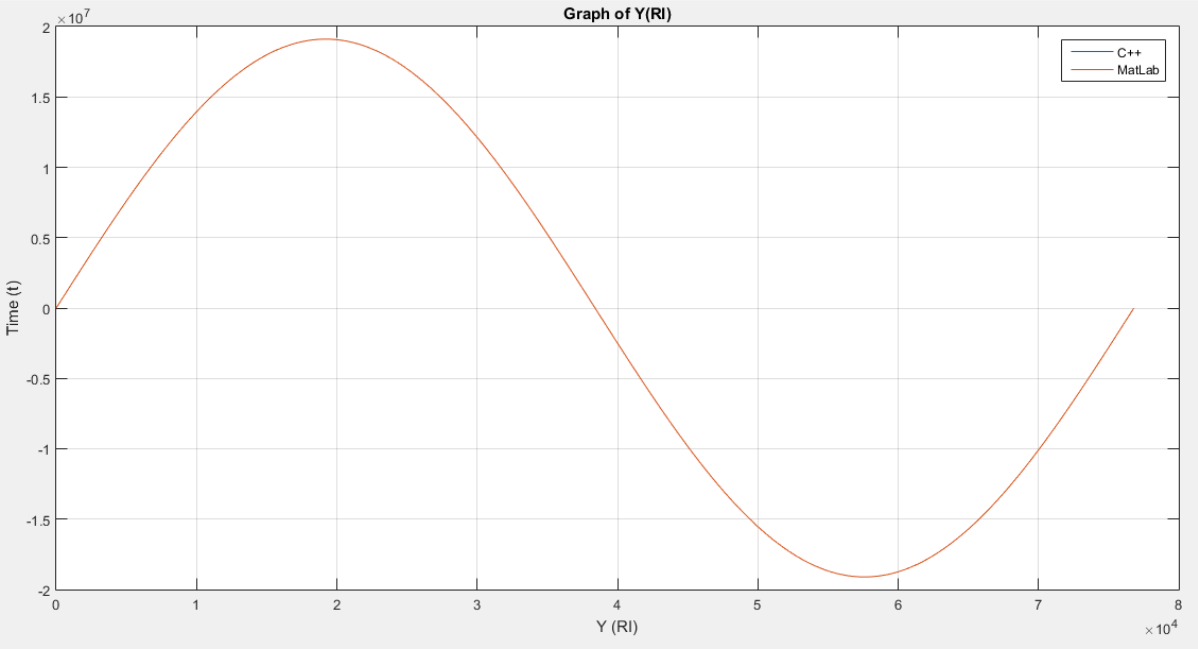
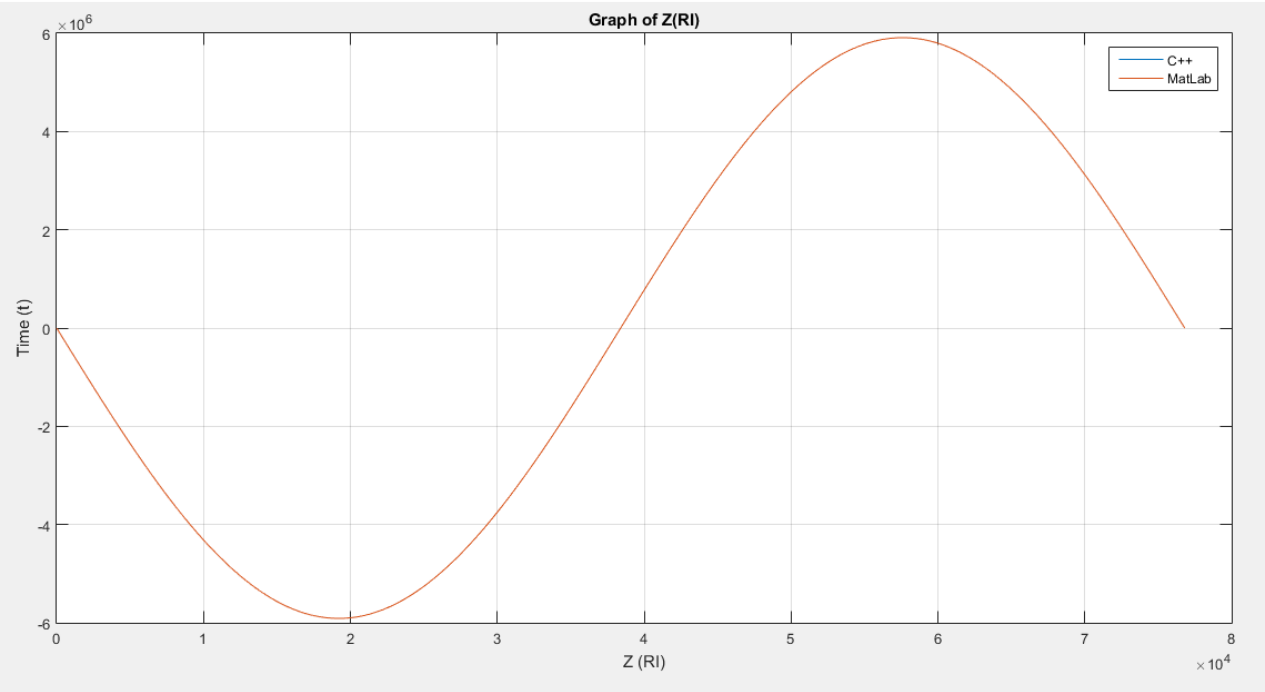
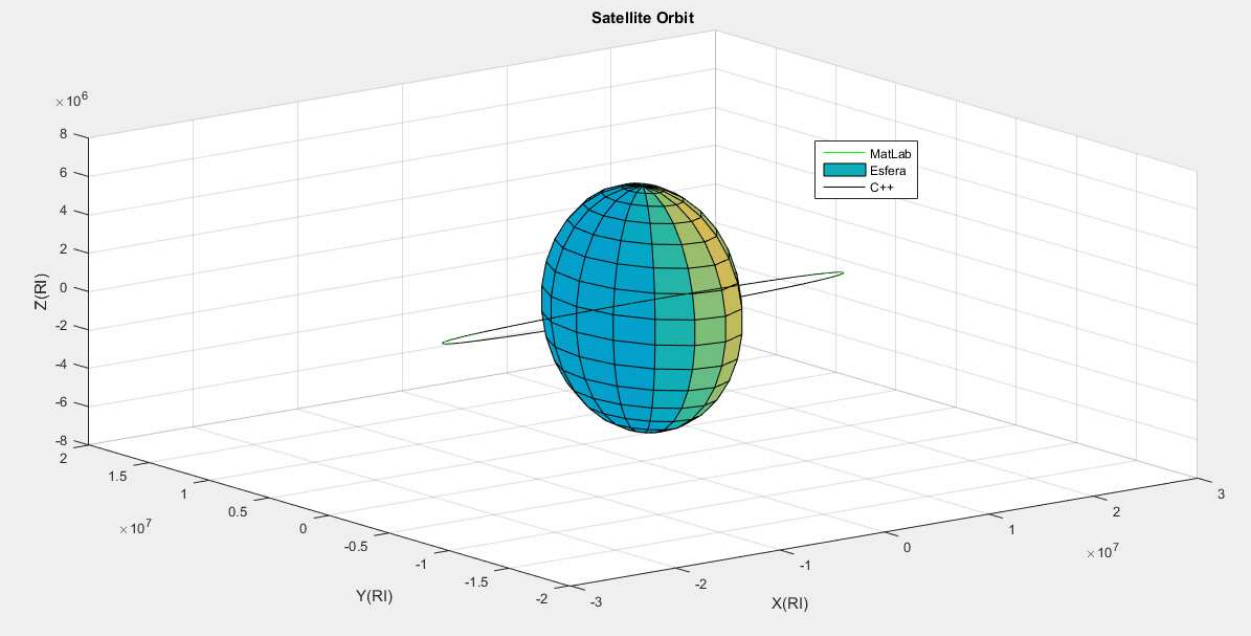


Figura 7.12 – Variação da Posição do Satélite em Z, ao longo do Tempo.



Nota: Os valores calculados usando o C++ e MATLAB, não apresentaram diferenças, pois os cálculos foram feitos do mesmo modo, ou seja, utilizando equações trigonométricas e Ângulos de Euler.

Figura 7.13 – Gráfico 3-D Plotado no MATLAB.



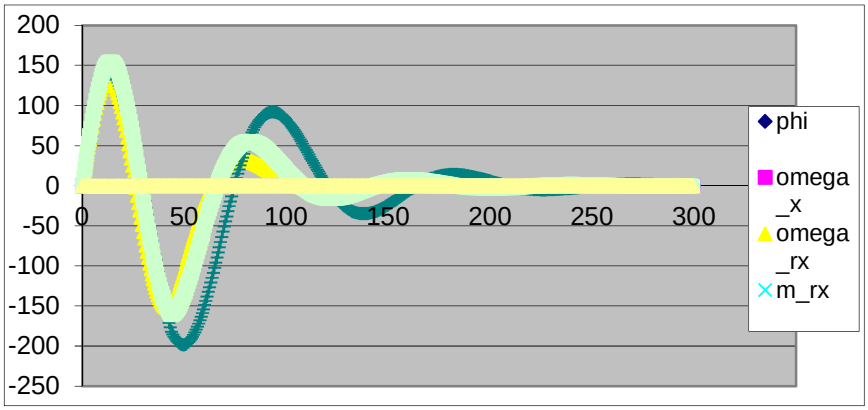
7.6. Simulação do SCA (Modelo C++)

Hardwares	*Notebook Samsung Electronics.* Processador: Intel® Core™ i7-4510U CPU@2.00GHz. Memória Instalada: 8.00GHz.
Softwares	MÄK -RTI MATLAB/ Simulink
Sistema Operacional	Windows 10
Padrão	HLA 1.3

Descrição: O modelo do SCA do Amazônia-1 feito C++ está implementado conforme descrito no capítulo 5. Também neste capítulo, está descrito a integração com a HLA/RTI.

Resultados: Após a simulação, a saída do modelo é um arquivo .*txt* com as grandezas apresentadas na Figura 7.14.

Figura 7.14 – Saída da Simulação do SCA do Amazônia-1, implementado em C++.



7.7. Simulação do SCA (Modelo MATLAB/Simulink)

Hardwares	*Notebook Samsung Electronics.* Processador: Intel® Core™ i7-4510U CPU@2.00GHz. Memória Instalada: 8.00GHz.
Softwares	MATLAB
Sistema Operacional	Windows 10
Padrão	HLA 1516

Descrição: O modelo do SCA do Amazônia-1 feito no MATLAB/Simulink está apresentado nas figuras do Anexo A. Os dados usados na simulação são os apresentados no Capítulo 5 e as equações, cinemática e dinâmica, estão apresentadas no Capítulo 4.

Resultados: O Modelo apresenta saídas para osciloscópios do MATLAB/Simulink, mas também exporta as variáveis para um arquivo em Excel. Os dados do programa em C++, exportados para o Excel, foram comparados com os dados do MATLAB/Simulink.

A Figura 7.15 apresenta graficamente os resultados obtidos e a comparação dos mesmos. Pelos gráficos, pode-se verificar, que os erros foram muito pequenos ficando na ordem de 0.01. Esses erros se devem às diferenças entres os métodos de cálculos utilizados no arquivo em C++ e no MATLAB/Simulink.

Figura 7.15 - Comparação dos resultados das simulações - SCA - C++

7.8. Resumo sobre a Utilização dos *Softwares*

A Tabela 7.8 apresenta, resumidamente, as características das ferramentas utilizadas aqui neste trabalho. Em resumo, os *softwares* utilizados se demonstraram adequados e de fácil utilização.

Tabela 7.8 – Resumo sobre as Ferramentas Utilizadas.

Software	Padrão	Interface com o usuário	Conhecimento prévio do Padrão HLA	Conhecimento prévio de programação	Linguagem para Implementação	Dificuldade de Implementação do Modelo	Dificuldade de Implementação da Interface RTI	Maturidade (Popularidade)	Limitações Computacionais	Documentação	Suporte
VR-FORCES	HLA e DIS	Sim, apresenta GUI	Sim.	Sim.	C++	Média	Média.	Alta.	Sim: compiladores e *softwares* de suporte.	Vasta, mas não gratuita.	Comercial.
MÄK -RTI	HLA	Sim, apresenta GUI	Sim.	Sim.	C++	Média	Média.	Alta.	Sim: compiladores e *softwares* de suporte.	Vasta, mas não gratuita.	Comercial.
pRTI	HLA	Sim, apresenta GUI	Sim.	Sim.	C++ e Java	Média.	Média.	Alta.	Sim, compiladores.	Vasta, mas não gratuita.	Comercial.
PORTICO	HLA	Não.	Sim.	Sim.	C++ e Java	Média.	Média.	Alta.	Sim, compiladores e sistemas operacionais.	Limitada e gratuita.	Gratuito.
CERTI	HLA	Não.	Sim.	Sim.	C++ e Java	Não avaliado	Não avaliado.	Média.	Sim, compiladores e sistemas operacionais.	Limitada e gratuita.	Gratuito.
Open HLA (oh-la)	HLA	Não.	Sim.	Sim.	C++ e Java	Não avaliado	Não avaliado.	Baixa.	Sim, compiladores e sistemas operacionais.	Pouca e gratuita.	Gratuito.

8 CONCLUSÕES, COMENTÁRIOS E SUGESTÕES PARA TRABALHOS FUTUROS

8.1. Conclusões

A principal proposta desse trabalho é apresentar o estudo e desenvolvimento de simulações que apresentem interface com a padronização de simulações distribuída, em especial o HLA.

Para se atingir esse objetivo foram utilizados quatro *softwares*, que cumprem com os requisitos do HLA, sendo três comerciais e um de projeto de código aberto.

Utilizando-se estes *softwares* foram executadas diversas simulações, incluindo a simulação do modelo do SCAO da PMM do INPE, em seu modo normal de operação.

Deste estudo desenvolvido podem-se extrair as seguintes conclusões:

- O Padrão HLA para simulações pode ser perfeitamente aplicado às simulações de sistemas espaciais, especialmente com o intuito de se reutilizar modelos e conectar outras simulações;
- Os *softwares* utilizados demonstraram apresentar padronização com o HLA conforme proposto e interface muito parecidas, no que diz respeito às RTIs;
- Observa-se maturidade nos *softwares* aqui estudados e utilizados, principalmente através da literatura e publicações científicas, conforme apresentado nos capítulos 1 e 2.

8.2. Comentários a Respeito das RTIs Utilizadas, Dificuldades e Características

Os *softwares* utilizados neste trabalho demonstraram muitas semelhanças no que diz respeito à implementação da padronização do HLA.

A diferença entre eles está mais relacionada às limitações computacionais e interface com outros *softwares* de suporte. Foi justamente devido a este aspecto que surgiram as dificuldades durante o desenvolvimento do trabalho.

Por exemplo, todas as RTIs apresentam limitação com relação à utilização de compiladores. A maioria é compatível com o Visual Studio C++ 2010; porém, se outra versão de Visual Studio já foi instalada anteriormente na máquina, não é possível a compilação do modelo, sendo gerados erros de compilação durante o processo.

Em alguns casos, mesmo removendo-se a versão anterior, a compilação ou instalação ainda não é possível, sendo possível somente após uma formatação na máquina. A PORTICO e o pRTI apresentaram este problema, sendo que o pRTI apresentou mensagem durante a sua instalação. A PORTICO apenas apresentava erros de compilação sem definição muito clara do problema.

Neste sentido, a MÄK-RTI foi a mais simples pois, apesar de rodar com o Visual Studio C++ 2010 ou 2013, não houve necessidade de se removerem versões posteriores, como a 2015, que também estava na mesma máquina.

Também se observa incompatibilidade para se instalar duas RTIs de desenvolvedores diferentes em uma mesma máquina. Isso se deve ao fato das configurações de variáveis de ambiente apresentarem o mesmo nome (por exemplo, *%RTI_HOME%*), o que pode ser resolvido através de configuração manual e correções.

O *software* VR-FORCES foi o que apresentou maiores dificuldades devido às excessivas limitações para os *softwares* de suporte.

Outro aspecto que pode ser avaliado é a questão de documentação. Os *softwares* comerciais apresentam maior quantidade de documentos, tais como manuais para o usuário entre outros, porém, apesar de a documentação da PORTICO ser bem reduzida, a mesma se demonstrou suficiente para uma utilização mais simples. O VR-FORCES, apesar de apresentar vastos manuais, não apresentou documentação adequada para o desenvolvimento do API.

A existência de interface gráfica para o usuário também foi um aspecto em que os *softwares* diferiram. Por exemplo, a MÄK-RTI e a pRTI, apresentavam interfaces que facilitam a visualização da conexão dos federados. A PORTICO não apresenta interface gráfica.

Essas características não necessariamente representam desvantagens, pois a escolha da ferramenta vai ser com base na necessidade do projeto ou do usuário. Por exemplo, para projetos mais complexos com vários federados conectados, um *software* comercial, que tenha suporte dos desenvolvedores, por demanda, pode ser o mais adequado. Para empresas que já tenham uma equipe de desenvolvimento de *software* integrada e experiente, um projeto de código aberto (*open-source*), como a PORTICO, pode atender, pois os próprios envolvidos poderiam realizar as modificações sem grande necessidade de suporte.

Um ponto importante a se observar, é que, quando da escolha de se utilizar a arquitetura distribuída, mais especificamente utilizando o RTI/HLA, a RTI escolhida será única e fará parte do *core* (parte central e essencial) do conjunto de simulações interconectadas; e uma transição futura para outra RTI envolverá horas de trabalho para adaptação e custos adicionais. Para uma empresa ou instituição de grande porte, que apresenta diversos modelos distribuídos em diferentes locais, a transição de uma RTI para outra apresentará custos bem elevados. Por estas razões, uma análise de qual a RTI deve ser utilizado e se é necessário utilizar HLA, deve ser feita com cautela.

Com relação aos objetivos almejados pelo DoD quando da ideia de se criar uma arquitetura padronizada e as propostas iniciais vislumbradas pelo DMSO (extinto) para o HLA, observa-se que nem todas as características são cumpridas. Por exemplo, a proposta do HLA inicial era que de que o mesmo apresentasse interfaces *plug-and-play* (Reid, 2000), porém, não é o que se observa quando se utiliza as RTIs atualmente disponíveis. Todos requerem algum desenvolvimento para interfaceamento, e a maioria não é de simples utilização, pois não apresentam interface gráfica.

Outro ponto é a comercialização das RTIs, pois inicialmente a ideia era de a RTI ser gratuita, existindo inclusive uma RTI desenvolvida pelo DMSO, que foi descontinuada. Mas com respeito a isso, ainda existem RTIs com a proposta de *softwares open-source*, dando um pouco mais de liberdade para escolha.

Sobre a questão da padronização do HLA, através da literatura observa-se que existe uma preocupação dos desenvolvedores em se manter o padrão, até porque as ferramentas são analisadas pelo DoD, para que as mesmas sejam classificadas como padrão HLA. Porém, ainda se nota a limitação para se utilizar RTIs de diferentes desenvolvedores, dificultando a transição entre as ferramentas, caso haja necessidade durante um projeto, por exemplo, de se trocar de RTI.

E, por fim, o critério de o HLA permitir interconectar modelos distribuídos geograficamente, o que se infere da literatura, esse requisito parece ser atendido. Porém, a literatura aqui estudada não apresentou questões acerca de atrasos inseridos na simulação devidos à utilização da RTI. As literaturas aqui estudadas apresentam resultados positivos com relação ao HLA quando utilizado em modelos distribuídos geograficamente (conforme apresentado no capítulo 2).

Contudo, apesar de alguns aspectos ainda pendentes de evolução, o HLA demonstra algumas vantagens com relação aos padrões predecessores. Sendo

por exemplo mais flexível e permitindo maior controle sobre as informações que trafegam na simulação.

8.3. Propostas para Trabalhos Futuros

A seguir estão apresentadas algumas propostas para a realização de trabalhos futuros relacionados ao HLA:

- Utilização de outra RTI de código aberto, como por exemplo, a CERTI (da ONERA), para comparação com a PORTICO;
- Utilização de uma RTI de código aberto juntamente com um compilador também de código aberto, por exemplo, o GCC;
- Desenvolvimento de uma interface que possibilite a integração do ambiente de simulação MatLab/Simulink à arquitetura HLA;
- Estudo das características de comunicação (*layers, protocols,* etc.) das RTIs, visando sua classificação (*synchronous, assynchronous, etc.*) e uso em rede (*network*).
- Estudo das características temporais (*latency, jitter,* etc.) das RTIs, visando sua classificação (*soft, firm, hard*) e uso em tempo real (*real time*).
- E por fim, com um enfoque diferente, um estudo sobre o processo de verificação e aprovação para certificação de *softwares* conformes com os padrões HLA.

REFERÊNCIAS BIBLIOGRÁFICAS

AGÊNCIA ESPACIAL BRASILEIRA (AEB). **Programa Nacional de Atividades Espaciais**: PNAE. Brasília, 2012 – 2021, 2012.

AGÊNCIA ESPACIAL BRASILEIRA (AEB). **Plataforma Multi-Missão (PMM)**. Brasília, 2012. Disponível em:<http://www.aeb.gov.br/plataforma-multimissao-pmm/>. Acesso em: 14 de Janeiro de 2016.

AMORIM TERCEIRO, F. C. **Simulação paralela de um controle discreto para a Plataforma Multi-Missão com dois processadores e um sistema operacional de tempo real.** Dissertação (Mestrado em Mecânica Espacial e Controle) – Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, SP, Brasil, 2007. Disponível em: <http://urlib.net/8JMKD3MGPBW/32HCGQ8>.

ARANTES JÚNIOR, G. **Estudo comparativo de técnicas de controle de atitude em três eixos para satélites artificiais**. 2005. (INPE-12970 TDI/1018). Dissertação (Mestrado em Engenharia e Tecnologia Espaciais / Mecânica Espacial e Controle) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2005. Disponível em: <http://urlib.net/sid.inpe.br/jeferson/2005/03.09.14.25>.

ARGÜELLO, L.;. MIRÓ, J. Distributed interactive simulation for space projects. **ESA Bulletin,** n. 102, p. 125-130, May 2000.

BOGOSSIAN O.L; SOUZA, F. L. **CSE-200-4 introduções à tecnologia de satélites.** São José dos Campos: INPE, 2015. Curso de Pós Graduação do INPE.

CHOBOTOV, V. A. **Orbital mechanics –** second edition. Reston: American Institute of Aeronautics and Astronautics, 1996. (ISBN-1-56347-179-5).

CLARK, P.; RYAN, P.; ZALCMAN, L. **Advanced distributed simulation for the Australian Defence Force.** Melbourne**:** Air Operations Division Aeronautical and Maritime Research Laboratory. Austrália. 2000.

CROSBIE, R. E.; ZENOR, J. **High level architecture, module 1 –** basic concepts, parts 1-6**.** California State University. Curso *online*. Disponível em:< http://www.ecst.csuchico.edu/~hla>. Acesso em: 20 de Fevereiro de 2017.

CURTIS, H. D. **Orbital mechanics for engineering students**. Daytona Beach, FL: Embry-Riddle Aeronautical University Daytona Beach, 2005.

DAHMAN, J.; FUJIMOTO, R. M.; WEATHERLY, R. M. The DoD high level architecture: an update. In: INTERNATIONAL WORKSHOP ON DISTRIBUTED INTERACTIVE SIMULATION AND REAL-TIME APPLICATIONS, 2., 1998, Montreal, Canada. **Proceedings…** IEEE Computer Society, 1998. ISBN 0-8186-8594-8.

DoD. **RTI 1.3-next generation programmer's guide version 3.2. high level architecture run-time infrastructure.** Alexandria, VA: Department of Defense Modeling and Simulation Office. Revision 3. 2000.

European Global Navigation Satellite Systems Agency. **Orbital and technical parameters.** Disponível em <https://www.gsc-europa.eu/system-status/orbital-and-technical-parameters>. Acesso em: 12-Fev-2017.

FALCONE, A.; GARRO, A.; LONGO, F.; SPADAFORA, F. Simulation exploration experience: a communication system and a 3d real time visualization for a moon base simulated scenario. In: IEEE/ACM INTERNATIONAL SYMPOSIUM ON DISTRIBUTED SIMULATION AND REAL TIME APPLICATIONS, 18., 2014, Toulouse, France. **Proceedings...** Aeronautics and Space Institute, 2014.

GOBATO, M. F. **Controles monovariáveis e multivariáveis aplicados a sistemas aeroespaciais fracamente ou fortemente acoplados**. 2006. (INPE 14494 TDI/1175). Dissertação (Mestrado em Engenharia e Tecnologia Espaciais / Mecânica Espacial e Controle) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2006.

HALLIDAY, R. **Física**. 4. ed. Rio de Janeiro, LTC, 1984.

HENNINGER, A. E.; CUTTS, D. et al. **Live Virtual Constructive Architecture Roadmap (LVCAR).** Institute for Defense Analysis, 2008. Final Report.

U.S. CONGRESS, OFFICE OF TECHNOLOGY ASSESSMENT. **Distributed interactive simulation of combat.** Washington, DC: U.S. Government Printing Office, Sept., 1995. OTA-BP-ISS-151.

HILL, F. **The complete DIS PDU guide**. SISO. 11 de Março de 2005.

HOLLENBACH, J. W. **Inconsistency, neglect, and confusion; a historical review of dod distributed simulation architecture policies**. SISO. 2009.

HONG, L. J. **Wireless and satellite communications**. Notas de Classe. T. Pratt, C. W. Bostian, and J. Allnutt, Satellite Communications, 2/e. Wiley, 2003.

HUDGINS, G. **Test and Training Enabling Architecture (TENA) offers range interoperability and resource reuse solutions for test and training ranges.** Jacksonville, FL: NDIA Test & Evaluation Conference,2002.

INPE. **Instalações**. Disponível em: <http://www.inpe.br/institucional/sobre_inpe/instalacoes.php>. Acesso em: 01 de fevereiro de 2018.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (INPE). **Multi-Mission Platform attitude control and data dandling (ACDH) subsystem specification.** São José dos Campos, SP, Brasil: INPE, 2001.

JENSE, G.J. KUIJPERS, N.H.L.; DUMAY, A.C.M. **DIS and HLA: connecting people, simulations and simulators in the military, space and civil domains.** New York: AIAA, 1997.

KUGA, H.K., CARRARA, V., RAO, K. R. **Introdução à mecânica orbital -** 2a edição. Sao Jose dos Campos: INPE, 2012. 67 p. IBI: <8JMKD3MGPAW/3C76K98>. (sid.inpe.br/mtc-m05/2012/06.28.14.21.24-PUD). Disponível em: <http://urlib.net/8JMKD3MGPAW/3C76K98>.

KAPLAN, M.H. **Modern spacecraft dynamics & control**. New York: John Wiley & Sons, 1976.

McCALL, M.; MURRAY, B. **IEEE 1278 Distributed Interactive Simulation (DIS).** SISO. 26/05/2010.

MÖLLER, B., MORSE, L. K.; LIGHTNER, M.; LITTLE R.; LUTZ, R. **HLA evolved –** a summary of major technical improvement. Copyright Pitch Technologies AB, Sweden. 2008.

MÖLLER, B.; SOLLIN, P.; KARLSSON, M. ANTELIUS, F. **Early experiences from migrating to the HLA evolved C++ and Java APIs.** Copyright Pitch Technologies AB, Sweden. 2010.

MÖLLER, B.; KLASSON, F.; LÖFSTRAND, B.; SOLLIN, P. **Practical experiences from four HLA evolved federations.** Copyright Pitch Technologies AB, Sweden, 2011.

MÖLLER, B. **The HLA tutorial:** a practical guide for developing distributed simulations. Copyright Pitch Technologies AB, Sweden, 2012.

MORAES, A. L. O. **Apreciação, Simulação e Implementação da Arquitetura IMA Distribuída (DIMA) e sua Aplicação a Sistemas Espaciais.** 2017. Dissertação (Mestrado em Mecânica Espacial e Controle) – Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, SP, Brasil, 2017.

MOREIRA, M. L. B. **Projeto e simulação de um controle discreto para a Plataforma Multi-missão e sua migração para um sistema operacional de tempo real.** 2006. 181 p. IBI: <6qtX3pFwXQZGivnJSY/LHnAv>. (INPE-14202-TDI/1103). Dissertação (Mestrado em Mecânica Espacial e Controle) – Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, SP, Brasil, 2006. Disponível em: <http://urlib.net/6qtX3pFwXQZGivnJSY/LHnAv>.

NOSEWORTHY, J. R. TENA- **Supporting the Decentralized Development of Distributed Applications and LVC Simulations.** 2010 IEEE/ACM 14th International Symposium on Distributed Simulation and Real Time Applications.

NOULARD, E.; ROUSSELOT, J. **CERTI, an open source RTI, why and how**. Onera, França. 2010.

OGATA, K. **Modern control engineering**. 3.ed. Prentice Hall, 1997.

OLIVA, L.L.; SOUZA, M. L. O. **Modeling and simulation of a satellite propulsion subsystem by physical and signal flows.** In: WORKSHOP ON SPACE ENGINEERING AND TECHNOLOGY, 3. (WETE)., 2012, São José dos Campos. Anais... São José dos Campos: INPE, 2012. IBI: <8JMKD3MGP7W/3CF59GL>. Disponível em: <http://urlib.net/8JMKD3MGP7W/3CF59GL>.

ONERA. **CERTI install documentation by doxygen.** Paris, 2014

ONERA. **Certi user documentation.** Paris, 2014.

ONERA. **Savannah project.** Paris, 2017. Disponível em: <https://savannah.nongnu.org/projects/certi>. Acesso em: Novembro de 2017.

PETZ, I. Communication protocols in distributed simulation. In: SCIENTIFIC CONFERENCE OF YOUNG RESEARCHERS (SCYR 2010), 10., 2010, FEI TU of Košice, Slovak Republic. **Proceedings…** Dept. of Informatics and Computer Science, FEI TU of Košice, 2010.. –

PRUDÊNCIO, S. V. **Simulação digital em tempo real de um sistema de controle de atitude magnético autônomo de um satélite.** Dissertação (Mestrado em Mecânica Espacial e Controle) – Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, SP, Brasil, 1997.

REID, M. R. **An evaluation of HLA as framework for NASA modeling and simulation.** Whashington: NASA, MD, 2000.

REIS, A. M. **Simulação distribuída em tempo real de um sistema de controle de atitude e de órbita para a Plataforma Multi-Missão utilizando a arquitetura HLA.** 2009. 196 p. IBI: <8JMKD3MGP8W/34RR5T5>. (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/8JMKD3MGP8W/34RR5T5>.

ROMEIRO, S.; SOUZA, M. L. A discussion and demonstration of some characteristics of an advanced HLA/RTI environment for aerospace applications. In: SAE BRASIL International Congress and Display, 25.,São Paulo. **Proceedings…** SAE, 2016.

ROGERSON, S. P. **Implementation of a distributed interactive simulation inteface in a sea king flight simulator.** Toronto: Graduate Department of Aerospace Science and Engineering University of Toronto, 1997.

SALIO, J.R.M., RODRIGUES, J.M.L. **A comparison of simulation and operational architectures.** NADS Madrid, Spain, 2008.

SHARING EARTH OBSERVATION RESEARCH. **Sich-2 (optical observation mission-2).** Disponível em: <https://eoportal.org/web/eoportal/satellite-missions/content/-/article/sich2> . Acesso em: 12 de October, 2017.

SILVA, D. F.; MURAOKA, I.; GARCIA, E. C. Thermal control design conception of the Amazonia-1 satellite. **Journal of Aerospace Technology and Management**, v. 6, n. 2, p. 169-176, Apr.-Jun. 2014. DOI: <10.5028/jatm.v6i2.320>. Disponível em: <http://dx.doi.org/10.5028/jatm.v6i2.320>.

SOUZA, M. L. O. **Estudo e desenvolvimento de um sistema de controle de atitude ativo em três eixos para satélites artificiais usando atuadores pneumáticos a gás frio e volantes de reação**. 1980. (INPE 2000 TDL/042). Dissertação (Mestrado em Engenharia e Tecnologia Espaciais / Mecânica Espacial e Controle) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 1980. Disponível em: <http://urlib.net/6qtX3pFwXQZ3r59YCT/GT7QC>.

TAGAWA, G. B S. **Estudo de um controlador de atitude em um simulador de aviônica modular integrada (IMA) aplicado ao Satélite Amazônia-1**. 2013.(INPE- 02.08.16.21-TDI). Dissertação (Mestrado em Engenharia e Tecnologia Espaciais / Mecânica Espacial e Controle) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2013. Disponível em: <http://urlib.net/8JMKD3MGP7W/3DGNKP2>.

TENA. **Test and Training Enabling Architecture (TENA) Website**. Disponível em: < //www.tena-sda.org/doc/5.1/InstallationGuide/ig_Planning.html>. Acesso: 19 de Setembro, 2016.

TEMIZER, S**.**The state of the art and the future of modeling and simulation systems. **Journal of Aeronautics and Space Technologies**, v. 3, n. 1, p. 41-50, 2007.

The PORTICO Project user documentation. Disponível em: < http://portico.openlvc.org/index.php?title=Getting_Started_with_Portico_Java >. Acesso em: Novembro de 2017.

TRIVELATO, G. C. **Simulação distribuída:** o que é o potencial da HLA e web Simulation. São José dos Campos: INPE, 2003. 26 p. (INPE-9664-NTC/357). Disponível em: <http://urlib.net/sid.inpe.br/jeferson/2003/07.08.08.05>.

VT-MÄK RTI. **Users guide**.USA. Copyright © 2015 VT-MÄK.

VT-MÄK VR-Forces. **Users guide.** USA. Copyright © 2016 VT-MÄK.

VT-MÄK. **VR-Forces training, basics and background.** ©MÄK Technologies, Inc. February 2016.

VT-MÄK**. VR-Forces developers guide.** ©MÄK Technologies, 2013.

WERTZ, J. R. **Spacecraft attitude determination and control**. Dordrecht, Holland: D. Reidel, 1978.

WIKIPEDIA. **Link-Trainer**. Disponível em: <https://en.wikipedia.org/wiki/Link_Trainer>. Acesso em: 01 de fevereiro de 2018.

ZALCMAN, L. **What Distributed Interactive Simulation (DIS) Protocol Data Units (PDU) should my simulator have?** Australian Defence Force. Março de 2004.

ZANDONADI, D.; KAKIZAKI, M. **Welding and testing of propulsion subsystem of PMM-Based Satellite Qualification Model.** In: AIAA SPACE CONFERENCE AND EXPOSITION., 2013, San Diego. **Proceedings...** 2013. ISBN 9781624102394.

APÊNDICE A - MODELO SIMULINK SCA

Figura A.1 - Modelo SCA

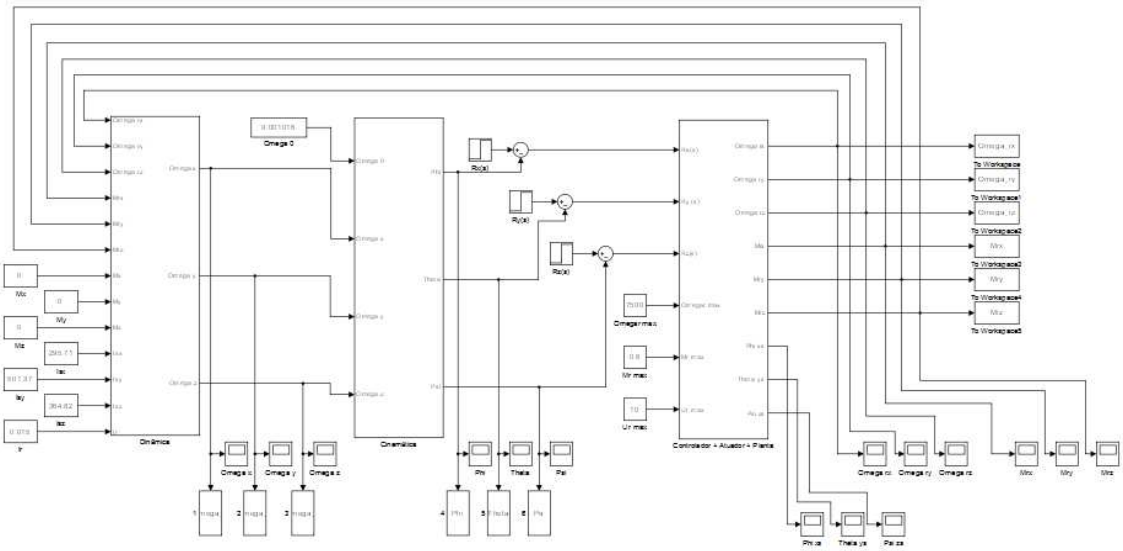


Figura A.2 - Modelo SCA - Dinâmica

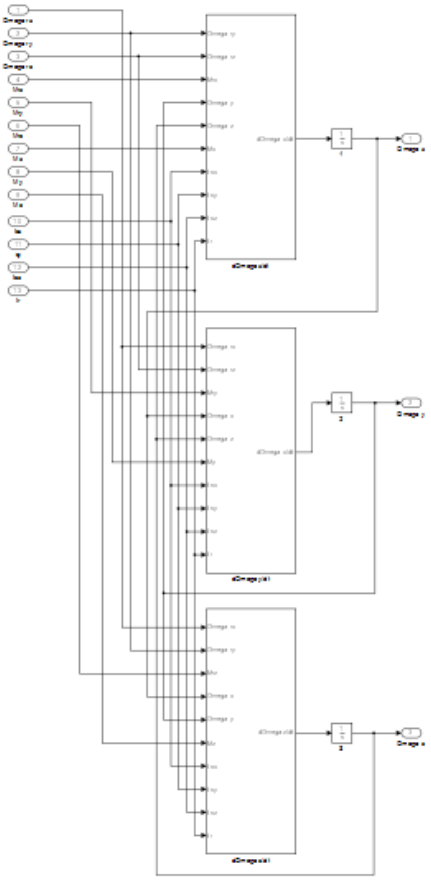


Figura A.3 - Modelo SCA - Dinâmica dOmega x/dt

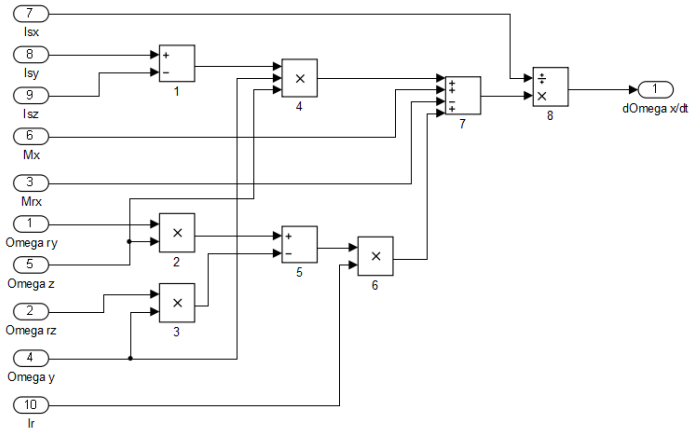


Figura A.4 - Modelo SCA - Dinâmica dOmega y/dt

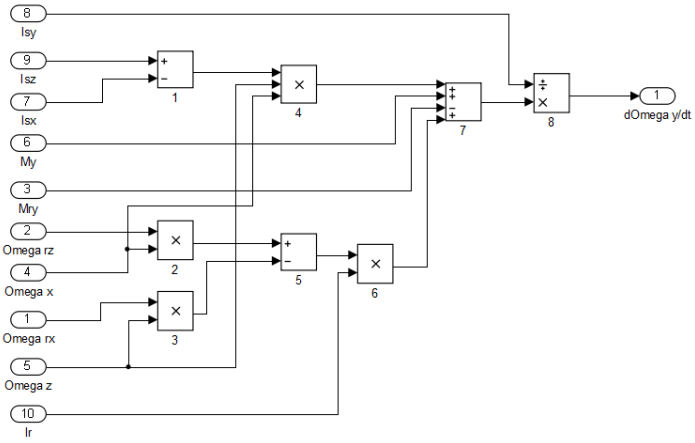


Figura A.5 - Modelo SCA - Dinâmica dOmega z/dt

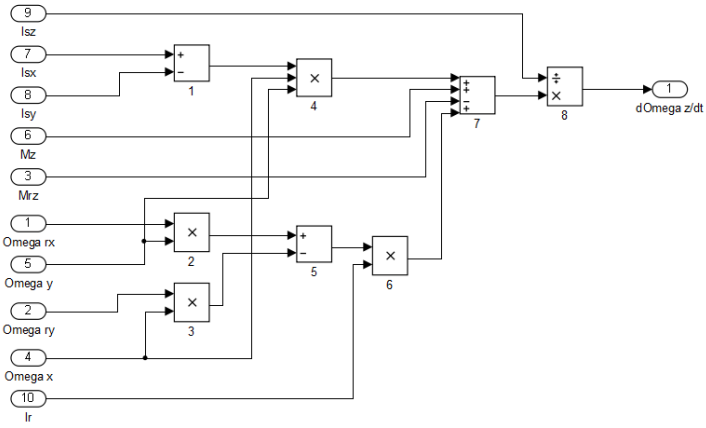


Figura A.6 - Modelo SCA - Cinemática

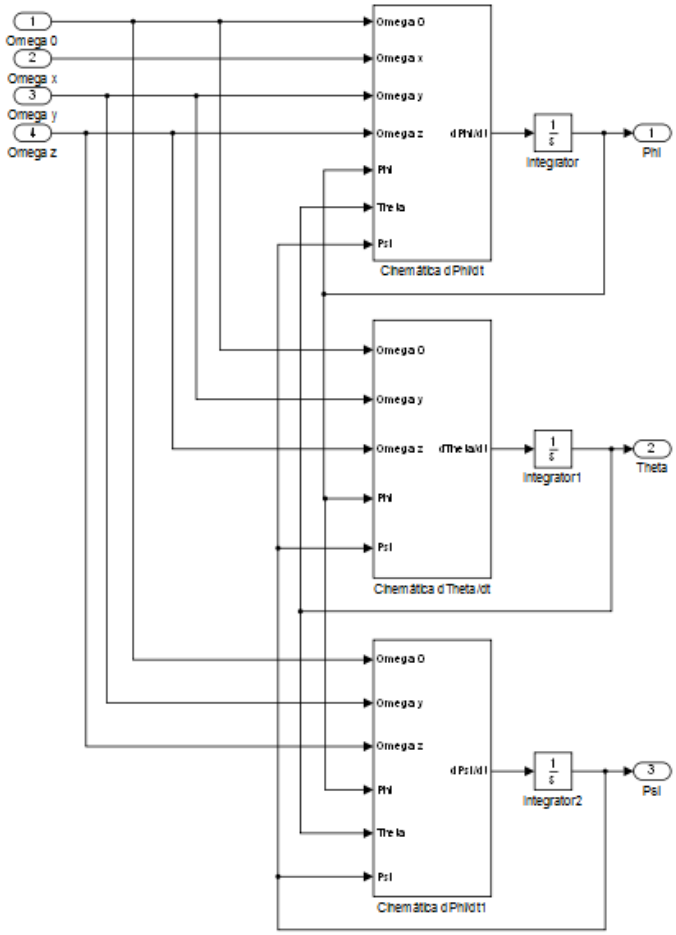


Figura A.7 - Modelo SCA - Cinemática dPhi/dt

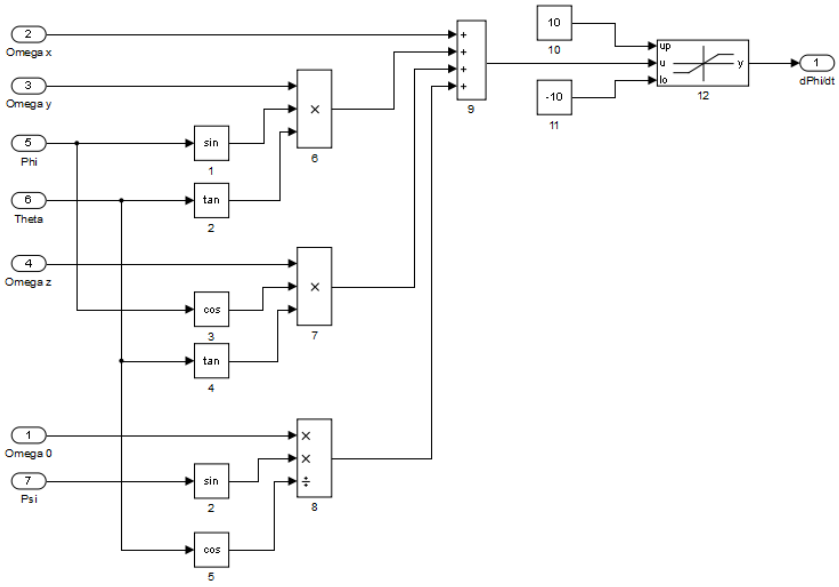


Figura A.8 - Modelo SCA - Cinemática dTheta/dt

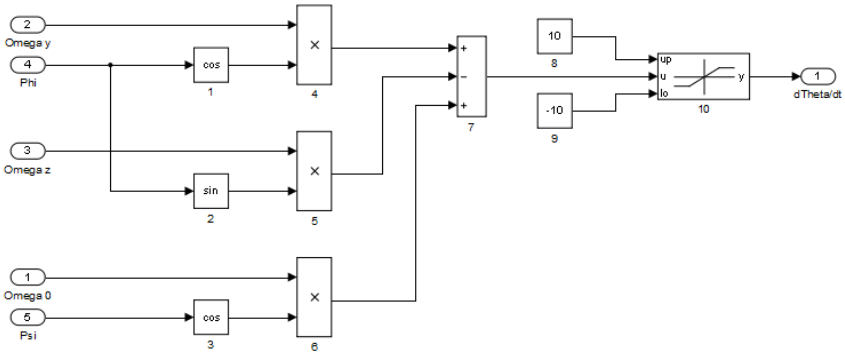


Figura A.9 - Modelo SCA - Cinemática dPsi/dt

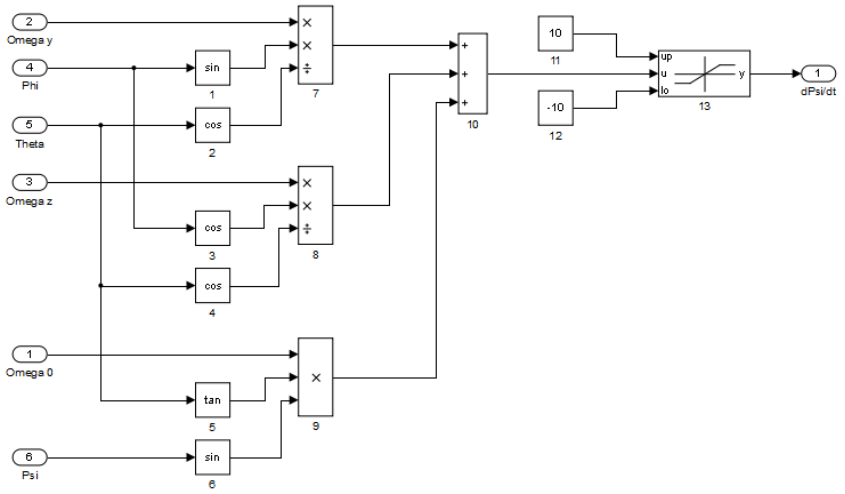


Figura A.10 - Modelo SCA - Controlador + Atuador + Planta

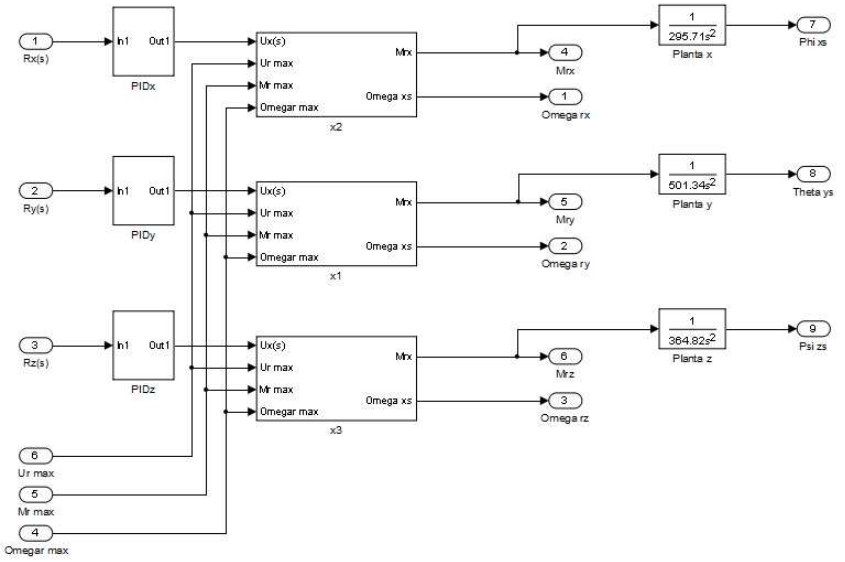


Figura A.11 - Modelo SCA - Controlador + Atuador + Planta - PIDx

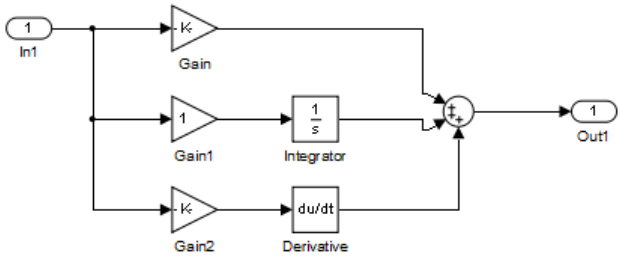


Figura A.12 - Modelo SCA - Controlador + Atuador + Planta - PIDy

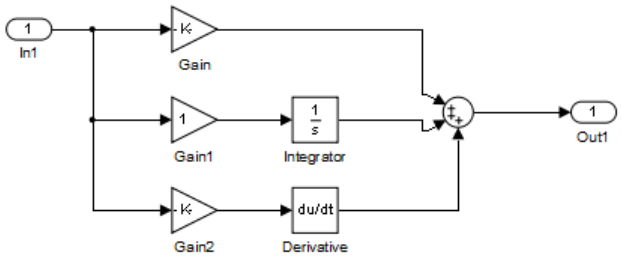


Figura A.13 - Modelo SCA - Controlador + Atuador + Planta - PIDz

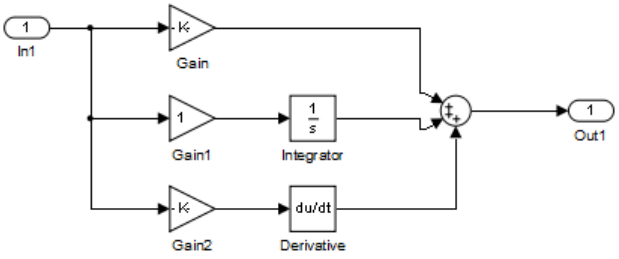


Figura A.14 - Modelo SCA - Controlador + Atuador + Planta - x1

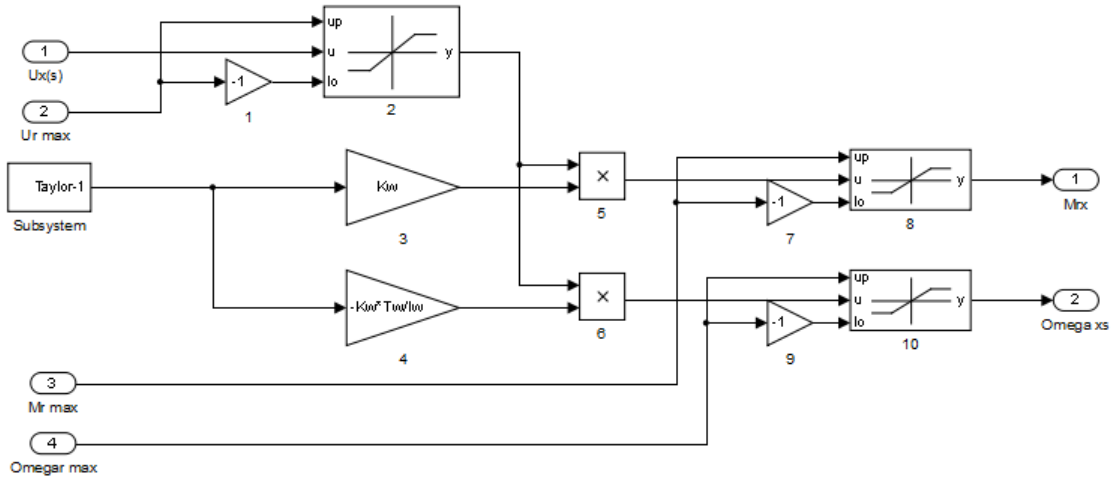


Figura A.15 - Modelo SCA - Controlador + Atuador + Planta - x2

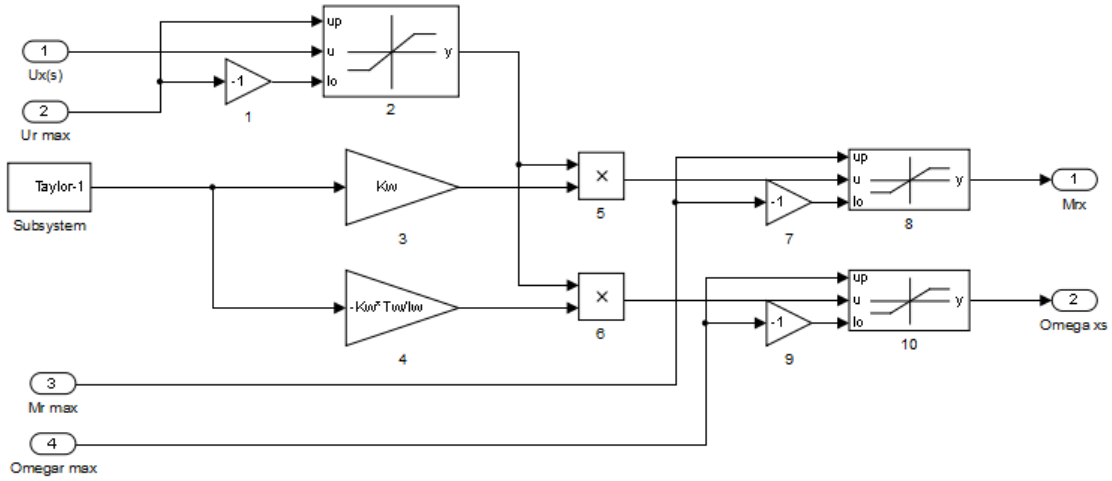
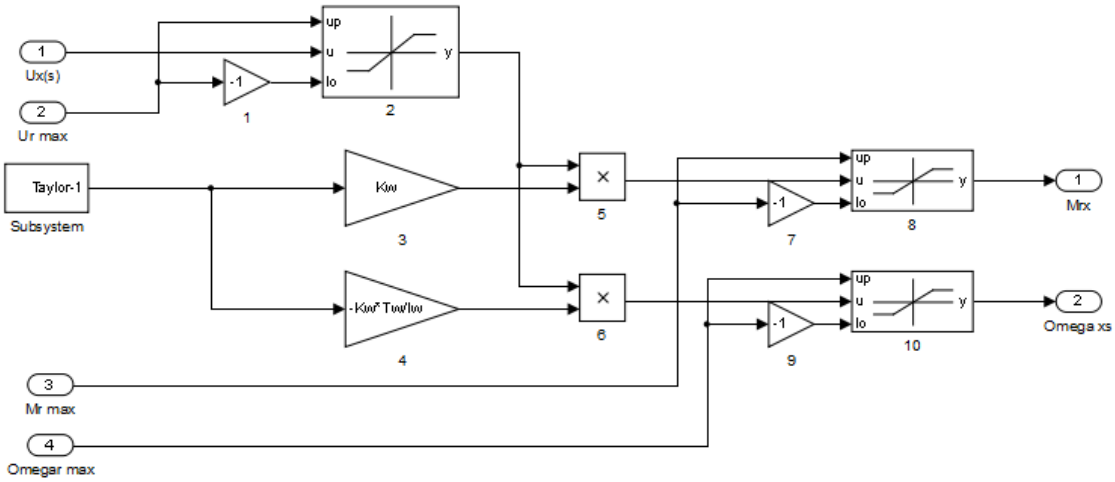


Figura A.16 - Modelo SCA - Controlador + Atuador + Planta - x3



APÊNDICE B – EXEMPLOS DE FOM

B.1. RTI da PORTICO

O FOM abaixo faz parte do exemplo fornecido junto com a instalação da RTI da PORTICO.

```
(FED
  (Federation Portico-Test)
  (FEDversion v1.3)
  (spaces
    (space TestSpace
      (dimension TestDimension)
    )
    (space OtherSpace
      (dimension OtherDimension)
    )
  )

  (objects
    (class ObjectRoot
      (attribute privilegeToDelete reliable timestamp)
      (class RTIprivate)
      (class A
        (attribute aa reliable timestamp TestSpace) ;; more
comments!
        (attribute ab reliable timestamp TestSpace)
        (attribute ac reliable timestamp TestSpace)
        (class B
             (attribute ba reliable timestamp TestSpace)
             (attribute bb reliable timestamp TestSpace)
             (attribute bc reliable timestamp TestSpace)
        )
      )
      (class BestEffortTest
        (attribute blah best_effort timestamp)
      )
      (class Manager
        (class Federation
          (attribute FederationName reliable receive)
          (attribute FederatesInFederation reliable receive)
          (attribute RTIversion reliable receive)
          (attribute FEDid reliable receive)
          (attribute LastSaveName reliable receive)
          (attribute LastSaveTime reliable receive)
          (attribute NextSaveName reliable receive)
          (attribute NextSaveTime reliable receive)
        )
        (class Federate
          (attribute FederateHandle reliable receive)
```

```
          (attribute FederateType reliable receive)
          (attribute FederateHost reliable receive)
          (attribute RTIversion reliable receive)
          (attribute FEDid reliable receive)
          (attribute TimeConstrained reliable receive)
          (attribute TimeRegulating reliable receive)
          (attribute AsynchronousDelivery reliable receive)
          (attribute FederateState reliable receive)
          (attribute TimeManagerState reliable receive)
          (attribute FederateTime reliable receive)
          (attribute Lookahead reliable receive)
          (attribute LBTS reliable receive)
          (attribute MinNextEventTime reliable receive)
          (attribute ROlength reliable receive)
          (attribute TSOlength reliable receive)
          (attribute ReflectionsReceived reliable receive)
          (attribute UpdatesSent reliable receive)
          (attribute InteractionsReceived reliable receive)
          (attribute InteractionsSent reliable receive)
          (attribute ObjectsOwned reliable receive)
          (attribute ObjectsUpdated reliable receive)
          (attribute ObjectsReflected reliable receive)
        )
      )
    )
  )
  (interactions
    (class InteractionRoot reliable timestamp
      (class RTIprivate reliable timestamp)
      (class X reliable timestamp TestSpace
        (parameter xa)
        (parameter xb)
        (parameter xc)
        (class Y reliable timestamp ;; note the absence of any
space definition
          (parameter ya)
          (parameter yb)
          (parameter yc)
          (class Z reliable timestamp TestSpace
            (parameter za)
            (parameter zb)
            (parameter zc)
          )
        )
      )
      (class DDM-Test reliable timestamp TestSpace
        (parameter ddm-parameter)
      )
      (class Manager reliable receive
        (class Federate reliable receive
          (parameter Federate)
```

```
            (class Request reliable receive
              (class RequestPublications reliable receive
              )
              (class RequestSubscriptions reliable receive
              )
              (class RequestObjectsOwned reliable receive
              )
              (class RequestObjectsUpdated reliable receive
              )
              (class RequestObjectsReflected reliable receive
              )
              (class RequestUpdatesSent reliable receive
              )
              (class RequestInteractionsSent reliable receive
              )
              (class RequestReflectionsReceived reliable receive
              )
              (class RequestInteractionsReceived reliable receive
              )
              (class RequestObjectInformation reliable receive
                (parameter ObjectInstance)
              )
            )
            (class Report reliable receive
              (class ReportObjectPublication reliable receive
                (parameter NumberOfClasses)
                (parameter ObjectClass)
                (parameter AttributeList)
              )
              (class ReportObjectSubscription reliable receive
                (parameter NumberOfClasses)
                (parameter ObjectClass)
                (parameter Active)
                (parameter AttributeList)
              )
              (class ReportInteractionPublication reliable receive
                (parameter InteractionClassList)
              )
              (class ReportInteractionSubscription reliable
receive
                (parameter InteractionClassList)
              )
              (class ReportObjectsOwned reliable receive
                (parameter ObjectCounts)
              )
              (class ReportObjectsUpdated reliable receive
                (parameter ObjectCounts)
              )
              (class ReportObjectsReflected reliable receive
                (parameter ObjectCounts)
              )
```

```
  (class ReportUpdatesSent reliable receive
    (parameter TransportationType)
    (parameter UpdateCounts)
  )
  (class ReportReflectionsReceived reliable receive
    (parameter TransportationType)
    (parameter ReflectCounts)
  )
  (class ReportInteractionsSent reliable receive
    (parameter TransportationType)
    (parameter InteractionCounts)
  )
  (class ReportInteractionsReceived reliable receive
    (parameter TransportationType)
    (parameter InteractionCounts)
  )
  (class ReportObjectInformation reliable receive
    (parameter ObjectInstance)
    (parameter OwnedAttributeList)
    (parameter RegisteredClass)
    (parameter KnownClass)
  )
  (class Alert reliable receive
    (parameter AlertSeverity)
    (parameter AlertDescription)
    (parameter AlertID)
  )
  (class ReportServiceInvocation reliable receive
    (parameter Service)
    (parameter Initiator)
    (parameter SuccessIndicator)
    (parameter SuppliedArgument1)
    (parameter SuppliedArgument2)
    (parameter SuppliedArgument3)
    (parameter SuppliedArgument4)
    (parameter SuppliedArgument5)
    (parameter ReturnedArgument)
    (parameter ExceptionDescription)
    (parameter ExceptionID)
  )
)
(class Adjust reliable receive
  (class SetTiming reliable receive
    (parameter ReportPeriod)
  )
  (class ModifyAttributeState reliable receive
    (parameter ObjectInstance)
    (parameter Attribute)
    (parameter AttributeState)
  )
  (class SetServiceReporting reliable receive
```

```
            (parameter ReportingState)
          )
          (class SetExceptionLogging reliable receive
            (parameter LoggingState)
          )
        )
        (class Service reliable receive
          (class ResignFederationExecution reliable receive
            (parameter ResignAction)
          )
          (class SynchronizationPointAchieved reliable receive
            (parameter Label)
          )
          (class FederateSaveBegun reliable receive
          )
          (class FederateSaveComplete reliable receive
            (parameter SuccessIndicator)
          )
          (class FederateRestoreComplete reliable receive
            (parameter SuccessIndicator)
          )
          (class PublishObjectClass reliable receive
            (parameter ObjectClass)
            (parameter AttributeList)
          )
          (class UnpublishObjectClass reliable receive
            (parameter ObjectClass)
          )
          (class PublishInteractionClass reliable receive
            (parameter InteractionClass)
          )
          (class UnpublishInteractionClass reliable receive
            (parameter InteractionClass)
          )
          (class SubscribeObjectClassAttributes reliable
receive
            (parameter ObjectClass)
            (parameter AttributeList)
            (parameter Active)
          )
          (class UnsubscribeObjectClass reliable receive
            (parameter ObjectClass)
          )
          (class SubscribeInteractionClass reliable receive
            (parameter InteractionClass)
            (parameter Active)
          )
          (class UnsubscribeInteractionClass reliable receive
            (parameter InteractionClass)
          )
          (class DeleteObjectInstance reliable receive
```

```
              (parameter ObjectInstance)
              (parameter Tag)
              (parameter FederationTime)
            )
            (class LocalDeleteObjectInstance reliable receive
              (parameter ObjectInstance)
            )
            (class ChangeAttributeTransportationType reliable
receive
              (parameter ObjectInstance)
              (parameter AttributeList)
              (parameter TransportationType)
            )
            (class ChangeAttributeOrderType reliable receive
              (parameter ObjectInstance)
              (parameter AttributeList)
              (parameter OrderingType)
            )
            (class ChangeInteractionTransportationType reliable
receive
              (parameter InteractionClass)
              (parameter TransportationType)
            )
            (class ChangeInteractionOrderType reliable receive
              (parameter InteractionClass)
              (parameter OrderingType)
            )
            (class UnconditionalAttributeOwnershipDivestiture
reliable receive
              (parameter ObjectInstance)
              (parameter AttributeList)
            )
            (class EnableTimeRegulation reliable receive
              (parameter FederationTime)
              (parameter Lookahead)
            )
            (class DisableTimeRegulation reliable receive
            )
            (class EnableTimeConstrained reliable receive
            )
            (class DisableTimeConstrained reliable receive
            )
            (class EnableAsynchronousDelivery reliable receive
            )
            (class DisableAsynchronousDelivery reliable receive
            )
            (class ModifyLookahead reliable receive
              (parameter Lookahead)
            )
            (class TimeAdvanceRequest reliable receive
              (parameter FederationTime)
```

```
            )
            (class TimeAdvanceRequestAvailable reliable receive
              (parameter FederationTime)
            )
            (class NextEventRequest reliable receive
              (parameter FederationTime)
            )
            (class NextEventRequestAvailable reliable receive
              (parameter FederationTime)
            )
            (class FlushQueueRequest reliable receive
              (parameter FederationTime)
            )
          )
        )
      )
    )
  )
)
```

B.2. RTI da VT-MÄK

O código a seguir foi adaptado de um FOM desenvolvido pela VT-MÄK, para a utilização da MÄK -RTI.

```
(FED
(Federation FederationName)
(FEDversion v1.3)
 (spaces
 )
 (objects
    (class ObjectRoot
       (attribute privilegeToDelete  reliable timestamp)
 (class Satellite
     (attribute Location best_effort timestamp)
     (attribute VelocityVector best_effort timestamp)


 )
    )
)
 (interactions
    (class InteractionRoot  reliable timestamp
      (class InitialCond reliable timestamp
        (parameter Altitude)
        (parameter Incl)
        (parameter Omega)
        (parameter time)

      )
      (class SatPos reliable timestamp
        (parameter PosX)
        (parameter PosY)
        (parameter PosZ)
        (parameter PosXp)
        (parameter PosYp)
        (parameter PosZp)
       )
  (class Manager reliable receive
  (class Federate reliable receive
     (parameter Federate)
   (class Request reliable receive
  (class RequestPublications reliable receive
 )

  (class RequestSubscriptions reliable receive
 )

  (class RequestObjectsOwned reliable receive
```

```
)

 (class RequestObjectsUpdated reliable receive
)

 (class RequestObjectsReflected reliable receive
)

 (class RequestUpdatesSent reliable receive
)

 (class RequestInteractionsSent reliable receive
)

 (class RequestReflectionsReceived reliable receive
)

 (class RequestInteractionsReceived reliable receive
)

 (class RequestObjectInformation reliable receive
    (parameter ObjectInstance)
 )

)

 (class Report reliable receive
 (class ReportObjectPublication reliable receive
    (parameter NumberOfClasses)
     (parameter ObjectClass)
     (parameter AttributeList)
 )

 (class ReportObjectSubscription reliable receive
    (parameter NumberOfClasses)
     (parameter ObjectClass)
     (parameter Active)
     (parameter AttributeList)
 )

 (class ReportInteractionPublication reliable receive
    (parameter InteractionClassList)
 )

 (class ReportInteractionSubscription reliable receive
    (parameter InteractionClassList)
 )

 (class ReportObjectsOwned reliable receive
    (parameter ObjectCounts)
 )
```

```
(class ReportObjectsUpdated reliable receive
   (parameter ObjectCounts)
)

(class ReportObjectsReflected reliable receive
   (parameter ObjectCounts)
)

(class ReportUpdatesSent reliable receive
   (parameter TransportationType)
    (parameter UpdateCounts)
)

(class ReportReflectionsReceived reliable receive
   (parameter TransportationType)
    (parameter ReflectCounts)
)

(class ReportInteractionsSent reliable receive
   (parameter TransportationType)
    (parameter InteractionCounts)
)

(class ReportInteractionsReceived reliable receive
   (parameter TransportationType)
    (parameter InteractionCounts)
)

(class ReportObjectInformation reliable receive
   (parameter ObjectInstance)
    (parameter OwnedAttributeList)
    (parameter RegisteredClass)
    (parameter KnownClass)
)

(class Alert reliable receive
   (parameter AlertSeverity)
    (parameter AlertDescription)
    (parameter AlertID)
)

(class ReportServiceInvocation reliable receive
   (parameter Service)
    (parameter Initiator)
    (parameter SuccessIndicator)
    (parameter SuppliedArgument1)
    (parameter SuppliedArgument2)
    (parameter SuppliedArgument3)
    (parameter SuppliedArgument4)
    (parameter SuppliedArgument5)
```

```
     (parameter ReturnedArgument)
     (parameter ExceptionDescription)
     (parameter ExceptionID)
 )

)

 (class Adjust reliable receive
 (class SetTiming reliable receive
    (parameter ReportPeriod)
 )

 (class ModifyAttributeState reliable receive
    (parameter ObjectInstance)
     (parameter Attribute)
     (parameter AttributeState)
 )

 (class SetServiceReporting reliable receive
    (parameter ReportingState)
 )

 (class SetExceptionLogging reliable receive
    (parameter LoggingState)
 )

)

 (class Service reliable receive
 (class ResignFederationExecution reliable receive
    (parameter ResignAction)
 )

 (class SynchronizationPointAchieved reliable receive
    (parameter Label)
 )

 (class FederateSaveBegun reliable receive
)

 (class FederateSaveComplete reliable receive
    (parameter SuccessIndicator)
 )

 (class FederateRestoreComplete reliable receive
    (parameter SuccessIndicator)
 )

 (class PublishObjectClass reliable receive
    (parameter ObjectClass)
     (parameter AttributeList)
```

```
)

(class UnpublishObjectClass reliable receive
   (parameter ObjectClass)
)

(class PublishInteractionClass reliable receive
   (parameter InteractionClass)
)

(class UnpublishInteractionClass reliable receive
   (parameter InteractionClass)
)

(class SubscribeObjectClassAttributes reliable receive
   (parameter ObjectClass)
    (parameter AttributeList)
    (parameter Active)
)

(class UnsubscribeObjectClass reliable receive
   (parameter ObjectClass)
)

(class SubscribeInteractionClass reliable receive
   (parameter InteractionClass)
    (parameter Active)
)

(class UnsubscribeInteractionClass reliable receive
   (parameter InteractionClass)
)

(class DeleteObjectInstance reliable receive
   (parameter ObjectInstance)
    (parameter Tag)
    (parameter FederationTime)
)

(class LocalDeleteObjectInstance reliable receive
   (parameter ObjectInstance)
)

(class ChangeAttributeTransportationType reliable receive
   (parameter ObjectInstance)
    (parameter AttributeList)
    (parameter TransportationType)
)

(class ChangeAttributeOrderType reliable receive
   (parameter ObjectInstance)
```

```
      (parameter AttributeList)
      (parameter OrderingType)
  )

  (class ChangeInteractionTransportationType reliable receive
     (parameter InteractionClass)
      (parameter TransportationType)
  )

  (class ChangeInteractionOrderType reliable receive
     (parameter InteractionClass)
      (parameter OrderingType)
  )

  (class UnconditionalAttributeOwnershipDivestiture reliable
receive
     (parameter ObjectInstance)
      (parameter AttributeList)
  )

  (class EnableTimeRegulation reliable receive
     (parameter FederationTime)
      (parameter Lookahead)
  )

  (class DisableTimeRegulation reliable receive
 )

  (class EnableTimeConstrained reliable receive
 )

  (class DisableTimeConstrained reliable receive
 )

  (class EnableAsynchronousDelivery reliable receive
 )

  (class DisableAsynchronousDelivery reliable receive
 )

  (class ModifyLookahead reliable receive
     (parameter Lookahead)
  )

  (class TimeAdvanceRequest reliable receive
     (parameter FederationTime)
  )

  (class TimeAdvanceRequestAvailable reliable receive
     (parameter FederationTime)
  )
```

```
   (class NextEventRequest reliable receive
      (parameter FederationTime)
   )

   (class NextEventRequestAvailable reliable receive
      (parameter FederationTime)
   )

   (class FlushQueueRequest reliable receive
      (parameter FederationTime)
   )

  )

  )

  )

     )
  )
 )
```