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

sid.inpe.br/mtc-m21b/2017/12.01.09.17-TDI

UM AMBIENTE DE GERENCIAMENTO DE DADOS E PROCESSOS DE SIMULAÇÃO E SUA APLICAÇÃO EM SISTEMAS ESPACIAIS

Rodrigo Britto Maria

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 14 de dezembro de 2017.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34P/3Q5BQSP>

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

COMISSÃO DO CONSELHO DE EDITORAÇÃO E PRESERVAÇÃO DA PRODUÇÃO INTELECTUAL DO INPE (DE/DIR-544):
Presidente:
Maria do Carmo de Andrade Nono - Conselho de Pós-Graduação (CPG)
Membros:
Dr. Plínio Carlos Alvalá - Centro de Ciência do Sistema Terrestre (COCST)
Dr. André de Castro Milone - Coordenação-Geral de Ciências Espaciais e Atmosféricas (CGCEA)
Dra. Carina de Barros Melo - Coordenação de Laboratórios Associados (COCTE)
Dr. Evandro Marconi Rocco - Coordenação-Geral de Engenharia e Tecnologia Espacial (CGETE)
Dr. Hermann Johann Heinrich Kux - Coordenação-Geral de Observação da Terra (CGOBT)
Dr. Marley Cavalcante de Lima Moscati - Centro de Previsão de Tempo e Estudos Climáticos (CGCPT)
Silvia Castro Marcelino - Serviço de Informação e Documentação (SESID)
BIBLIOTECA DIGITAL:
Dr. Gerald Jean Francis Banon
Clayton Martins Pereira - Serviço de Informação e Documentação (SESID)
REVISÃO E NORMALIZAÇÃO DOCUMENTÁRIA:
Simone Angélica Del Ducca Barbedo - Serviço de Informação e Documentação (SESID)
Yolanda Ribeiro da Silva Souza - Serviço de Informação e Documentação (SESID)
EDITORAÇÃO ELETRÔNICA:
Marcelo de Castro Pazos - Serviço de Informação e Documentação (SESID)
André Luis Dias Fernandes - Serviço de Informação e Documentação (SESID)

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

sid.inpe.br/mtc-m21b/2017/12.01.09.17-TDI

UM AMBIENTE DE GERENCIAMENTO DE DADOS E PROCESSOS DE SIMULAÇÃO E SUA APLICAÇÃO EM SISTEMAS ESPACIAIS

Rodrigo Britto Maria

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 14 de dezembro de 2017.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34P/3Q5BQSP>

INPE
São José dos Campos
2017

Dados Internacionais de Catalogação na Publicação (CIP)

Maria, Rodrigo Britto.

M337a Um ambiente de gerenciamento de dados e processos de simulação e sua aplicação em sistemas espaciais / Rodrigo Britto Maria. – São José dos Campos : INPE, 2017.

xxx + 196 p. ; (sid.inpe.br/mtc-m21b/2017/12.01.09.17-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. Gerenciamento de dados. 2. Gerenciamento de processos. 3. Simulação. I.Título.

CDU 629.7.018

Esta obra foi licenciada sob uma Licença Creative Commons Atribuição-NãoComercial 3.0 Não Adaptada.

This work is licensed under a Creative Commons Attribution-NonCommercial 3.0 Unported License.

Aluno (a): ***Rodrigo Britto Maria***

Título: **"UM AMBIENTE DE GERENCIAMENTO DE DADOS E PROCESSOS DE SIMULAÇÃO E SUA APLICAÇÃO EM SISTEMAS ESPACIAIS".**

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. Maurício Gonçalves Vieira Ferreira

Presidente / INPE / SJCampos - SP

Dr. Marcelo Lopes de Oliveira e Souza

Orientador(a) / INPE / SJCampos - SP

Dr. Ronan Arraes Jardim Chagas

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

Dr. Fernando José de Oliveira Moreira

Convidado(a) / EMBRAER / SJCampos - SP

Este trabalho foi aprovado por:

( ) maioria simples

(X) unanimidade

São José dos Campos, 14 de dezembro de 2017

“Deem-me um ponto de apoio e moverei o mundo”.

Arquimedes

Ao Espírito universal que habita em cada um de nós.

AGRADECIMENTOS

Nesta página de agradecimentos, agradeço a todos os *stakeholders* envolvidos na conclusão épica deste trabalho:

Agradeço a Deus, ao meu anjo da guarda e a todos os espíritos de Luz que me guiam, orientam e protegem, por todas as bênçãos concedidas durante a realização deste curso.

Agradeço a meus pais por terem me permitido chegar até esta etapa do caminho.

Agradeço aos meus amigos, por terem perdoado minhas ausências em tantos finais de semana, feriados, férias, viagens e comemorações, para que este trabalho pudesse se concretizar.

Agradeço à Embraer pela disponibilização de tempo para a realização deste trabalho e aos meus colegas pelas sugestões, dicas e apoio recebido durante a elaboração do mesmo.

Agradeço ao INPE, pelo acolhimento e pela perseverança em continuar investindo em educação de qualidade, mesmo em um ambiente adverso e com limitações de recursos.

Agradeço aos meus colegas de curso, por terem me apoiado nos momentos difíceis e por todas as dicas e ajudas, de todos os tipos, recebidas durante o curso.

Agradeço a todos os professores do Curso de Pós-Graduação ETE do INPE, pelos ensinamentos transmitidos e pela disposição em dar o seu melhor para formar profissionais de destaque no mercado de trabalho.

Agradeço ao meu orientador, professor Marcelo Souza, pelo apoio, orientação e inspiração na condução deste trabalho. Tal qual um guru espiritual, muitas vezes simplesmente por estar em sua presença, as dúvidas se desfaziam e as respostas eram encontradas.

Por fim, agradeço a minha própria força interior, por me fazer acreditar em mim mesmo e seguir em frente, a despeito de todos os obstáculos no caminho.

RESUMO

Para o desenvolvimento de sistemas complexos, cada vez mais estão sendo utilizados processos de engenharia simultânea que integram times responsáveis por diferentes disciplinas de projeto. O volume de dados gerados por processos de simulação no desenvolvimento destes sistemas tem crescido cada vez mais, o que gera problemas como a falta de rastreabilidade e reuso dos dados, retrabalho e aumento de custos do projeto. Para lidar com estes problemas, foram desenvolvidos, nos últimos anos, sistemas especializados em gerenciamento de dados e processos de simulação. Este trabalho propõe e implementa um protótipo de um ambiente de gerenciamento de dados e processos de simulação válido para as fases 0 e A de projetos espaciais e o aplica em um caso de uso de simulação de sistemas espaciais. O método utilizado neste trabalho consiste em realizar uma revisão da literatura, incluindo normas da ECSS, identificar os requisitos de gerenciamento de dados e processos de simulação de sistemas espaciais sugeridos por estas normas e aplicar o ambiente proposto em casos de uso da área espacial. A solução proposta consiste de um ambiente que integra dois softwares (*Remote Component Environment e Virtual Satellite*) desenvolvidos pelo Centro Aeroespacial Alemão (DLR) e um método para usar essa integração. Embora limitada em algumas funcionalidades, esta solução cumpre o objetivo a que se propõe e pode ser considerada por organizações, como o INPE, que utilizam fortemente modelagem e simulação e precisam gerenciar dados e processos de simulação em projetos espaciais.

Palavras-chave: Gerenciamento de Dados. Gerenciamento de Processos. Simulação.

A SIMULATION PROCESS AND DATA MANAGEMENT ENVIRONMENT AND ITS APPLICATION TO SPACE SYSTEMS

ABSTRACT

For the development of complex systems, concurrent engineering processes are being progressively used to integrate teams responsible for different design disciplines. The volume of data generated by simulation processes in the development of these systems has grown increasingly, which generates problems such as lack of data traceability and reuse, rework and increased project costs. To deal with these problems, specialized systems for simulation process and data management have been developed in recent years. This work proposes and implements a prototype of a simulation process and data management environment that is valid for phases 0 and A of space projects and applies it to a space system simulation use case. The method used in this work is to perform a literature review, including ECSS standards, to identify the simulation process and data management requirements suggested by these standards, and to apply the proposed environment in use cases of space system simulations. The proposed solution consists of an environment that integrates two software (Remote Component Environment and Virtual Satellite) developed by the German Aerospace Center (DLR) and a method to use this integration. Although limited in some functionalities, this solution fulfills the objective to which it proposes itself and can be considered by organizations, such as INPE, which strongly use modeling and simulation and need to manage data and simulation processes in space projects.

Keywords: Data Management. Information Management. Simulation.

LISTA DE FIGURAS

Pág.

Figura 2.1 - Estrutura de Divisão do Produto (EDP)..... 8
Figura 2.2 – Fases, atividades e revisões do ciclo de vida de um projeto espacial típico..... 10
Figura 2.3 – Processo de *Engenharia de Sistemas.* 14
Figura 2.4 – Definições das fases de um projeto espacial e linhas de base correspondentes..... 18
Figura 2.5 – Definição de linhas de base no lado esquerdo do modelo V..... 20
Figura 2.6 - Definição de linhas de base no lado direito do modelo V..... 21
Figura 2.7 – Papel de M&S no processo de *Engenharia de Sistemas.* 24
Figura 2.8–Identificação e prevenção antecipada de defeitos. 25
Figura 2.9 - Instituto de Sistemas Espaciais do DLR em Bremen (esquerda) e *layout* do CEF (direita). 27
Figura 2.10 – Processo de engenharia simultânea do DLR. 28
Figura 2.11 – Exemplo de gerenciamento de dados de um veículo espacial... 30
Figura 2.12 – *Pedigree* de dados de um processo de simulação com três atividades. 32
Figura 2.13 – Infraestrutura típica de um sistema SPDM. 37
Figura 3.1 – Componentes da arquitetura dos recursos de simulações e testes. 46
Figura 3.2 - SMP2 *Conformance Profiles*. 48
Figura 3.3 - Implementação do gerenciamento de informações e documentações. 49
Figura 3.4 - Technical Data Package (TDP). 50
Figura 3.5 - Esquema XML da norma ECSS-M-ST-40C. 50
Figura 3.6 – Aplicação de software típica e seus componentes de dados. 52
Figura 3.7 – Cenários de troca de dados entre aplicações de software. 53
Figura 3.8 – Repositório de dados de sistemas espaciais. 54

Figura 3.9 – Adoção do modelo de dados conceitual global da ECSS para um projeto. 57
Figura 3.10 – Árvore de produto típica de um sistema espacial. 58
Figura 3.11 – Exemplo de tipos de objetos e entidades de um elemento de sistema. 59
Figura 3.12 – Dados de um elemento de sistema ao longo de seu ciclo de vida. 59
Figura 3.13 – Vista de topo do SEIM. 61
Figura 3.14 - Tipos de objetos definidos no SEIM. 62
Figura 3.15 - Tipos de objetos definidos no SEIM e seus relacionamentos. 62
Figura 3.16 - Decomposição do sistema e modos associados no SEIM. 64
Figura 3.17 - Representação dos parâmetros de projeto simultâneo no SEIM. 65
Figura 4.1 - Interação do RCE com Eclipse RCP/OSGi e Aplicação Aeroespacial 74
Figura 4.2 - Componentes do RCE e suas dependências. 75
Figura 4.3 - Exemplo de fluxo colaborativo de ferramentas e dados. 77
Figura 4.4 – Arquivo de configuração padrão do RCE. 79
Figura 4.5 – Exemplo de configuração de rede RCE. 81
Figura 4.6 – Interface gráfica do RCE com exemplos de visualizações e editores. 82
Figura 4.7 – Janela do *Connection Editor* do RCE. 83
Figura 4.8 – Janela do *Network View* do RCE. 84
Figura 4.9 – Janela do *Workflow Data Browser* do RCE. 85
Figura 4.10 – Janela Properties do RCE. 85
Figura 4.11 – Janela Workflow Console do RCE. 86
Figura 4.12 – Conceito de integração de ferramentas no RCE. 88
Figura 4.13 – Exemplo de edição de *workflow* no RCE. 89
Figura 4.14 – Conexões de dados em um *workflow* do RCE. 90
Figura 4.15 - Vista hierárquica do *Integrated Design Model* (IDM) da ESA. 95
Figura 4.16 - Arquitetura do *Virtual Satellite.* 98
Figura 4.17 – Modelo de dados conceitual do VirSat. 99

Figura 4.18 – Arquitetura do modelo de dados do VirSat. 100
Figura 4.19 – Interface gráfica do VirSat com o conjunto de visualizações *PhaseA.* 102
Figura 4.20 – Aba *References* na visualização de componentes do VirSat. .. 103
Figura 4.21 – Aba *Calculation* na visualização de componentes do VirSat.... 104
Figura 4.22 – Aba Excel na visualização de componentes do VirSat. 105
Figura 4.23 – Janela de edição de Cálculos entre parâmetros do VirSat. 106
Figura 4.24 – Exemplo de arquivo de configuração do VirSat. 107
Figura 4.25 – Visualização do *Study Navigator* no VirSat. 108
Figura 4.26 – Visualização do *Role Management* no VirSat. 108
Figura 4.27 – Menu de associação de disciplinas a componentes no VirSat. 109
Figura 4.28 – Exemplos de casos de uso de modelamento de sistemas. 110
Figura 5.1 – Modelo de dados conceitual da proposta. 116
Figura 5.2 – Repositório de dados de sistemas espaciais da proposta. 117
Figura 5.3 – Conexões de rede entre os computadores na rede RCE no ambiente 119
Figura 5.4–Conexões de rede entre os computadores utilizando o VirSat. 121
Figura 5.5 – Identificação única do arquivo Excel com parâmetros do VirSat.123
Figura 5.6–Processo de operação do ambiente proposto. 126
Figura 5.7 – Processo de arquivamento de dados. 127
Figura 5.8 – Diagrama de fluxo de dados do ambiente proposto. 128
Figura 6.1 – Arquitetura da plataforma multimissão. 131
Figura 6.2 - Subsistema de Controle de Atitude e Órbita do satélite Amazônia-1. 133
Figura 6.3 – Diagrama de blocos do SCAO. 134
Figura 6.4 – Repositório SVN. 138
Figura 6.5 – Configuração de acesso ao repositório SVN. 138
Figura 6.6 – Arquivo de propriedades do VirSat para o estudo de caso espacial. 139
Figura 6.7 – Configuração do repositório SVN no VirSat. 139
Figura 6.8 – *Study Navigator* mostrando o novo estudo INPE_PMM. 140

Figura 6.9 – Pasta com nome do estudo criada no repositório SVN. 140
Figura 6.10 – Criação do modelo de estrutura de dados. 141
Figura 6.11 – Estrutura de dados do satélite Amazônia-1 no VirSat. 141
Figura 6.12 – *Role Management* no VirSat. 142
Figura 6.13 – Parâmetros utilizados na simulação do SCAO. 143
Figura 6.14–Mapeamento de parâmetros de entrada do VirSat para simulação RCE. 144
Figura 6.15 - Mapeamento de parâmetros de saída do VirSat para simulação RCE. 145
Figura 6.16 – Estrutura de diretórios de trabalho do RCE. 148
Figura 6.17 – *Workflow* de simulação no RCE. 151
Figura 6.18 – *Connection Editor* do RCE. 152
Figura 6.19 – Conexões entre dados de entrada e saída de componentes do *workflow.* 152
Figura 6.20 – Resultado de uma simulação no Simulink. 155
Figura 6.21 – Janela do *Workflow Data Browser* do RCE. 156
Figura 6.22 – Visualização detalhada do *Workflow Data Browser* do RCE. 157
Figura 6.23 – Exportação de *workflows* para arquivos ZIP no RCE. 159
Figura 6.24 – *Workflow* RCE para publicação de dados no VirSat. 159
Figura 6.25 – *Project Explorer* do RCE com a identificação da execução da simulação. 159
Figura 6.26 – Identificação do *workflow* RCE no VirSat. 160
Figura 7.1 – Exemplo da estrutura de dados do CPACS. 170
Figura 7.2 – Exemplo de representação de nervuras de asa no CPACS. 170
Figura 7.3 – Interface do *VAMPzero initializer*. 172
Figura 7.4 – Interface do *TiGL Viewer*. 172
Figura 7.5 – *Workflow* de simulação de decolagem. 175
Figura 7.6 – Conexões entre componentes do *workflow Takeoff.* 176
Figura 7.7 – *Workflow Data Browser* com dados de saída do *workflow Takeoff.* 177

LISTA DE TABELAS

Pág.

Tabela 2.1 – Fases do ciclo de vida de projetos espaciais. 9
Tabela 2.2 - Descrição das revisões de projeto. 10
Tabela 2.3 – Termos de M&S utilizados neste trabalho. 22
Tabela 2.4 – Simulação de sistemas no ciclo de vida de sistemas espaciais. . 22
Tabela 2.5 – Desafios operacionais em SPDM. 36
Tabela 5.1 – Função de cada computador no exemplo de arquitetura. 118
Tabela 6.1–Dados e condições iniciais do SCAO. 134
Tabela 6.2 – Configuração das entradas e saídas do componente *DataPrep.* 147
Tabela 6.3 - Configuração das entradas e saídas do componente MATLAB. 149
Tabela 7.1 – Parâmetros da simulação de decolagem. 173

LISTA DE SIGLAS E ABREVIATURAS

AC	Advisory Circular
ACDH	Attitude Control and Data Handling
ACE	Attitude Control Electronics
AOCS	Attitude and Orbit Control System
API	Application programming interface
AR	Acceptance Review
AWDT	Subsistema Transmissor de Dados da Câmera
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CDF	Concurrent Design Facilities
CDR	Critical Design Review
CEF	Concurrent Engineering Facility
ConOps	Concept of Operations
CPACS	Common Parametric Aircraft Configuration Scheme
CPRIME	Centro de Projeto Integrado de Missões Espaciais
CPU	Computer Processing Unit
CRR	Commissioning Results Review
DB	Design Baseline
DCB	Development Configuration Baseline
DDR	Digital Data Recorder
DLR	Deutsches Zentrum für Luft- und Raumfahrt
DMZ	DeMilitarized Zone
ECSS	European Cooperation for Space Standardization

ELR	End of Life Review
ESA	European Space Agency
FAA	Federal Aviation Administration
FAR	Federal Aviation Regulations
FCB	Functional Configuration Baseline
FRR	Flight Readiness Review
HPC	High Performance Computing
IDM	Integrated Design Model
IGU	Interface Gráfica com Usuário
INCOSE	International Council on Systems Engineering
INPE	Instituto Nacional de Pesquisas Espaciais
JSON	JavaScript Object Notation
LRR	Launch Readiness Review
M&S	Modelagem e Simulação
MBSE	Model-Based Systems Engineering
MCR	Mission Closeout Review
MDR	Mission Design Review
MOB	Mission Objective Baseline
NAFEMS	National Agency for Finite Element Methods and Standards
OBDH	On Board Data Handling
ODBC	Open Database Connectivity
ORR	Operational Readiness Review
OSGi	Open Service Gateway Initiative
PCB	Product Configuration Baseline

PDM	Product Data Management
PDR	Preliminary Design Review
PLM	Product Lifecycle Management
PMM	Plataforma Multimissão
PRR	Preliminay Requirements Review
QR	Qualification Review
RCE	Remote Component Environment
RCP	Rich Client Platform
RMI	Remote Method Invocation
RTU	Remote Telemetry Unit
SCAO	Sistema de Controle de Atitude e Órbita
SCS	Simulador de Conceito do Sistema
SDM	Simulation Data Management
SDM	Simulador de Desempenho da Missão
SE	Systems Engineering
SEIM	Systems Engineering Information Model
SFE	Simulador Funcional de Engenharia
SMP	Simulation Model Portability
SOAP	Simple Object Access Protocol
SPDM	Simulation Process and Data Management
SQL	Structured Query Language
SRR	Systems Requirements Review
SVF	Simulador de Validação Funcional
SVS	Simulador de Validação de Software

TDP	Technical Data Package
UML	Unified Modeling Language
UUID	Universally Unique IDentifier
VirSat	Virtual Satellite
XMI	XML Metadata Interchange
XML	eXtensible Markup Language
XSD	XML Schema Definition

LISTA DE SÍMBOLOS

h	=	Passo da simulação (s)
Ti	=	Ganho Integral de um PID
Kp	=	Ganho Proporcional de um PID
Td	=	Ganho Derivativo de um PID
Ti x	=	Ganho Integral de um PID em x
Kp x	=	Ganho Proporcional de um PID em x
Td x	=	Ganho Derivativo de um PID em x
Ti y	=	Ganho Integral de um PID em y
Kp y	=	Ganho Proporcional de um PID em y
Td y	=	Ganho Derivativo de um PID em y
Ti z	=	Ganho Integral de um PID em z
Kp z	=	Ganho Proporcional de um PID em z
Td z	=	Ganho Derivativo de um PID em z
Isx	=	Momento de Inércia do Satélite em torno do eixo x (kg.m2)
Isy	=	Momento de Inércia do Satélite em torno do eixo y (kg.m2)
Isz	=	Momento de Inércia do Satélite em torno do eixo z (kg.m2)
ϕ	=	Ângulo de rolamento (radianos)
θ	=	Ângulo de arfagem (radianos)
Ψ	=	Ângulo de guinada (radianos)
Xϕ ref	=	Posicionamento desejado ao longo do eixo x (radianos)
Yθ ref	=	Posicionamento desejado ao longo do eixo y (radianos)
ZΨ ref	=	Posicionamento desejado ao longo do eixo z (radianos)

$X\phi$	=	Posicionamento real ao longo do eixo x (radianos)
$Y\theta$	=	Posicionamento real ao longo do eixo y (radianos)
$Z\Psi$	=	Posicionamento realao longo do eixo z (radianos)
ω	=	Velocidade angular do satélite (rpm)
ωx	=	Velocidade angular torno do eixo x do satélite (rpm)
ωy	=	Velocidade angular torno do eixo y do satélite (rpm)
ωz	=	Velocidade angular torno do eixo z do satélite (rpm)
Mx	=	Torque do Satélite em torno do eixo x ($kg.m^2$)
My	=	Torque do Satélite em torno do eixo y ($kg.m^2$)
Mz	=	Torque do Satélite em torno do eixo z ($kg.m^2$)
ωr	=	Velocidade angular da roda (rpm)
ωrx	=	Velocidade angular da roda torno do eixo x (rpm)
ωry	=	Velocidade angular da roda torno do eixo y (rpm)
ωrz	=	Velocidade angular da roda torno do eixo z (rpm)
Mr	=	Torque da roda ($kg.m^2$)
Mrx	=	Torque da roda em torno do eixo x (N.m)
Mry	=	Torque da roda em torno do eixo y (N.m)
Mrz	=	Torque da roda em torno do eixo z (N.m)
ωo	=	Velocidade orbital média (rad/s)
Iw	=	Momento de Inércia das rodas ($kg.m^2$)
$\omega rmax$	=	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

1 INTRODUÇÃO 1
1.1. Contexto e motivação 1
1.2. Objetivo e relevância do trabalho 3
1.3. Metodologia aplicada ao trabalho 4
1.4. Organização deste Trabalho 5
2 CONCEITOS BÁSICOS E REVISÃO DA LITERATURA 7
2.1. Projeto de sistemas espaciais 7
2.2. Engenharia de Sistemas 11
2.2.1. Processo da Engenharia de Sistemas 12
2.3. Gerenciamento de linhas de base 17
2.4. Modelagem e Simulação de sistemas espaciais 21
2.4.1. Termos utilizados neste trabalho 21
2.4.2. Modelagem e Simulação no ciclo de vida de sistemas espaciais . 22
2.5. Engenharia Simultânea 25
2.5.1. Concurrent Design Facilities (CDFs) 26
2.5.2. Processo de Engenharia Simultânea 28
2.6. Simulation Process and Data Management (SPDM) 29
2.6.1. Conceitos básicos 29
2.6.2. Principais desafios em SPDM 33
2.6.3. Principais funcionalidades de sistemas SPDM 36
2.6.4. Infraestrutura típica de sistemas SPDM 37
3 NORMAS APLICÁVEIS A SPDM 41
3.1. Introdução 41
3.2. ECSS-E-TM-10-21A 42
3.2.1. Tipos de simuladores 43
3.2.2. Componentes de uma simulação 44
3.3. ECSS-E-TM-40-07 45
3.4. ECSS-M-ST-40 48
3.5. ECSS-E-TM-10-23A 51
3.5.1. Motivação 51

3.5.2. Aplicações de software no suporte ao ciclo de vida de sistemas espaciais 52
3.5.3. Modelo de dados conceitual global de sistemas espaciais 56
3.6. ECSS-E-TM-10-25A 60
3.6.1. System Engineering Information Model (SEIM) 60
4 AMBIENTES DE GERENCIAMENTO DE DADOS E PROCESSOS DE SIMULAÇÃO 69
4.1. DEFINIÇÕES 69
4.2. SOFTWARE *FRAMEWORK* RCE 70
4.2.1. Histórico 70
4.2.2. Requisitos 70
4.2.3. Arquitetura do sistema 73
4.2.4. Componentes 74
4.2.5. Configuração da rede RCE 78
4.2.6. Interface gráfica 82
4.2.7. Integração e distribuição de ferramentas 86
4.2.8. Workflows 88
4.2.9. Gerenciamento de dados 91
4.3. CHAMALEON 92
4.4. VIRTUAL SATELLITE 93
4.4.1. Histórico 93
4.4.2. Arquitetura do sistema 97
4.4.3. Interface gráfica 101
4.4.4. Configuração do sistema 106
4.4.5. Casos de uso 109
4.5. REQUISITOS DO AMBIENTE DE GERENCIAMENTO DE DADOS E PROCESSOS DE SIMULAÇÃO A SER USADO NO INPE 110
5 PROPOSTA DO AMBIENTE DE GERENCIAMENTO DE DADOS E PROCESSOS DE SIMULAÇÃO PARA O INPE 113
5.1. Introdução 113
5.2. Modelo de dados conceitual 114
5.3. Repositório de dados de sistemas espaciais 116

5.4. Arquitetura do ambiente 117
5.5. Configuração da integração entre RCE e VirSat 122
5.6. Procedimentos de operação do ambiente 124
6 ESTUDO DE CASO ESPACIAL: IMPLEMENTAÇÃO E AVALIAÇÃO ... 129
6.1. Plataforma Multimissão 129
6.2. Satélite Amazônia-1 130
6.3. Sistema de controle de atitude e órbita 132
6.4. Definição do estudo de caso espacial 136
6.5. Configuração da simulação do SCAO no ambiente proposto 137
6.5.1. Configuração do VirSat 137
6.5.2. Configuração do RCE 144
6.6. Execução da simulação do SCAO no ambiente proposto 153
6.7. Publicação do *workflow* no VirSat 157
6.8. Resultados do estudo de caso espacial 160
7 ESTUDO DE CASO AERONÁUTICO: IMPLEMENTAÇÃO E AVALIAÇÃO 163
7.1. Introdução 163
7.2. Normas de gerenciamento de dados aeronáuticos 164
7.3. Sinergias identificadas entre as normas espaciais e aeronáuticas 166
7.4. Adaptação do ambiente para o estudo de caso aeronáutico 168
7.5. Definição do estudo de caso aeronáutico 173
7.6. Configuração da simulação de decolagem no RCE 175
7.7. Resultados do estudo de caso aeronáutico 176
8 CONCLUSÕES, LIÇÕES APRENDIDAS E TRABALHOS FUTUROS .. 179
8.1. Sugestões para trabalhos futuros 184
REFERÊNCIAS BIBLIOGRÁFICAS 187

1 INTRODUÇÃO

1.1. Contexto e motivação

Com o progressivo aumento da capacidade computacional disponível, que tem dobrado a cada 18 meses nos últimos anos segundo a chamada “Lei de Moore” (MOORE, 2006), engenheiros da área aeroespacial têm intensificado o uso de simulações multidisciplinares de diversos tipos de sistemas durante a concepção, desenvolvimento e certificação de produtos. O número e o volume de dados gerados têm crescido na mesma proporção. Empresas que fazem uso intensivo de simulações de engenharia reportam a criação de um número da ordem de 200.000 arquivos por dia (NORRIS, 2012), totalizando um volume de dados da ordem de *Terabytes* por dia (NORRIS, 2010). Muitos dos dados resultantes destas simulações são usados para demonstrar o cumprimento de requisitos a agências reguladoras, e devem estar disponíveis e rastreáveis durante toda a vida útil dos produtos (FAA, 2010).

Entretanto, não havia, até o final dos anos 1990, tecnologias de gerenciamento de dados e processos de simulação, tornando difícil o rastreamento das informações e mesmo a captura e gestão do conhecimento gerado durante este processo. Sem este tipo de controle, os dados e os processos usados no projeto de produtos podem ser facilmente perdidos, impossibilitando o reuso destas informações, o que onera o desenvolvimento de novos produtos (NORRIS, 2012). Estudos sobre produtividade no ambiente empresarial realizados em empresas que desenvolvem produtos complexos reportam que engenheiros passam de 15% a 25% do seu tempo procurando por informações necessárias para a realização de seus trabalhos. Uma perda ainda maior de tempo ocorre quando estas informações não são encontradas, resultando em retrabalho (FELDMAN; SHERMAN, 2001).

Para resolver este problema, empresas tradicionais na área de *Computer Aided Engineering* (CAE) lançaram, a partir de 2004, sistemas de *Simulation Data Management* (SDM), sendo o primeiro deles o software SimManager,

desenvolvido pela MSC Software (JACKSON, 2012). O NAFEMS (anteriormente conhecido como *National Agency for Finite Element Methods and Standards*), uma associação internacional cuja missão é fornecer conhecimento, colaboração internacional e oportunidades educacionais para o uso e validação de simulações de engenharia (NAFEMS, 2017a), foi uma das primeiras organizações a perceber a importância do SDM. Congressos dedicados ao SDM têm sido organizados pelo NAFEMS desde 2007 (NAFEMS, 2017b). A partir de 2013, ao reconhecer a importância não apenas do gerenciamento de dados, mas também dos processos que os geram, o NAFEMS ampliou a denominação da tecnologia para *Simulation Process and Data Management* (SPDM) e lançou a primeira conferência internacional de SPDM, sigla que foi amplamente adotada pelos fabricantes de software e usuários da tecnologia (NAFEMS, 2013).

Segundo Seider et al. (2012), o Centro Espacial Alemão (DLR) iniciou, em 2005, o desenvolvimento de um software *framework* chamado RCE (*Remote Component Environment*) com o objetivo de fornecer aos engenheiros de diferentes disciplinas um ambiente comum de colaboração, permitindo que eles compartilhem dados e ferramentas de simulação de forma gerenciada. Em 2010, o DLR tornou o RCE um software de código aberto, para permitir a criação de uma comunidade de usuários que não apenas usa o RCE, mas também contribui com o seu desenvolvimento. Membros desta comunidade podem provir de organizações com orçamento limitado para investimento em licenças de software, que de outra forma não conseguiriam se beneficiar da utilização de um software com as funcionalidades do RCE.

O RCE permite a construção de *workflows* colaborativos onde cada engenheiro pode ser responsável por uma etapa do processo de simulação. Construído com o reuso como um de seus requisitos principais, o software *framework* RCE foi utilizado como base para dois outros sistemas desenvolvidos pelo DLR: o *Chameleon*, um ambiente para projeto preliminar de aeronaves, e o *Virtual Satellite* (VirSat), uma aplicação de apoio para sessões de engenharia

simultânea durante as fases 0 e A de projetos de desenvolvimento de sistemas espaciais (SEIDER et al., 2012).

Dado que o Instituto Nacional de Pesquisas Espaciais (INPE) e o DLR são organizações que possuem alguns objetivos em comum, entre eles a concepção e especificação de sistemas espaciais (DLR, 2017a; BRASIL, 2016) e dado ainda que o INPE não dispõe de um sistema de gerenciamento de dados e processos de simulação de sistemas espaciais, foi realizada uma proposta de trabalho de mestrado para investigar uma possível utilização do RCE e do *VirSat* como base de um ambiente de gerenciamento de dados e processos de simulação a ser utilizado no INPE nas fases 0 e A de projetos espaciais. O Centro de Projeto Integrado de Missões Espaciais (CPRIME), localizado no INPE, especializado em análises das fases iniciais de missões espaciais, poderia se beneficiar com a introdução do ambiente proposto neste trabalho.

1.2. **Objetivo e relevância do trabalho**

Este trabalho de mestrado tem como objetivo propor e implementar um protótipo de um ambiente de gerenciamento de dados e processos de simulação para ser usado no INPE, tomando como base o software *framework* RCE desenvolvido pelo DLR, e aplicá-lo em casos de uso da área aeroespacial.

Dado um cenário em que o conhecimento produzido durante a concepção e desenvolvimento de sistemas espaciais no INPE pode ser perdido devido a fatores como a migração dos profissionais responsáveis para outras organizações, ou mesmo por ocasião de sua aposentadoria, é de suma importância a existência de um ambiente computacional onde estes profissionais possam registrar o conhecimento gerado e compartilhá-lo com os demais envolvidos.

Embora o RCE já tenha sido utilizado com sucesso em projetos no DLR (SEIDER et al., 2012), este trabalho contribui com a análise deste software *framework* utilizando casos de uso de outra organização (INPE), reconhecendo seus pontos fortes e identificando oportunidades de melhoria que podem dar origem a outros trabalhos de pesquisa no INPE, visto que o RCE é um software *open source* e pode ser desenvolvido em parceria com o DLR.

1.3. **Metodologia aplicada ao trabalho**

Para atingir o objetivo proposto neste trabalho, foi utilizado um método composto das seguintes etapas:

a) realizar uma revisão da literatura para obter o conhecimento do estado da arte do tema em questão, incluindo a identificação de melhores práticas adotadas por instituições de pesquisa e pela indústria, e normas afins da *European Cooperation for Space Standardization* (ECSS);

b) identificar os requisitos de gerenciamento de dados e processos de simulação de sistemas espaciais sugeridos por uma destas normas que devem ser atendidos pelo software *framework* que será utilizado como base do ambiente de gerenciamento de dados e processos de simulação;

c) propor e implementar um ambiente de gerenciamento de dados e processos de simulação de sistemas espaciais tomando como base o software *framework* RCE e compatibilizá-lo com uma das normas identificadas na etapa b);

d) selecionar e executar um estudo de caso de simulação de sistemas da área espacial, utilizando o ambiente proposto e implementado na etapa c). O objetivo da realização deste estudo é a verificação e validação do ambiente proposto;

e) selecionar e executar um estudo de caso adicional de simulação de sistemas da área aeronáutica, utilizando o ambiente proposto e implementado na etapa c), com as adaptações que se fizerem necessárias. O objetivo da realização deste estudo é a verificação de possíveis sinergias entre as áreas espacial e aeronáutica, no que tange ao gerenciamento de dados e processos de simulação.

1.4. **Organização deste Trabalho**

Este trabalho está organizado da seguinte forma:

Capítulo 1: INTRODUÇÃO

Apresenta uma visão inicial das dificuldades atuais em gerenciamento de dados e processos de simulação e da tecnologia que foi desenvolvida para resolvê-las. São introduzidas também as soluções desenvolvidas pelo DLR e a motivação, objetivos e métodos usados na condução deste trabalho.

Capítulo 2: CONCEITOS BÁSICOS E REVISÃO DA LITERATURA

Apresenta os conceitos básicos de Modelagem e Simulação (M&S) e SPDM, contextualizando-os dentro do processo de *Engenharia de Sistemas* utilizado no desenvolvimento de sistemas espaciais.

Capítulo 3: NORMAS APLICÁVEIS A SPDM

Apresenta as normas e recomendações relacionadas a M&S e gerenciamento de dados de sistemas espaciais.

Capítulo 4: AMBIENTES DE GERENCIAMENTO DE DADOS E PROCESSOS DE SIMULAÇÃO

Apresenta o conceito de “ambiente” e detalha as funcionalidades dos software *frameworks* selecionados como base do ambiente proposto: RCE e VirSat.

Capítulo 5: PROPOSTA DO AMBIENTE DE GERENCIAMENTO DE DADOS E PROCESSOS DE SIMULAÇÃO PARA O INPE

Apresenta a arquitetura do ambiente de gerenciamento de dados e processos de simulação proposto para ser utilizado no INPE.

Capítulo 6: ESTUDO DE CASO ESPACIAL: IMPLEMENTAÇÃO E AVALIAÇÃO

Apresenta o estudo de caso espacial utilizado para validação do ambiente proposto. Este estudo consiste no gerenciamento dos dados e do processo de simulação do sistema de controle de altitude e órbita do satélite Amazônia-1 do INPE.

Capítulo 7: ESTUDO DE CASO AERONÁUTICO: IMPLEMENTAÇÃO E AVALIAÇÃO

Apresenta o estudo de caso aeronáutico, que consiste na aplicação de conceitos das normas ECSS na especificação de um ambiente de gerenciamento de dados e processos de simulação que pode ser utilizado na área aeronáutica.

Capítulo 8: CONCLUSÕES, LIÇÕES APRENDIDAS E TRABALHOS FUTUROS

Apresenta as conclusões obtidas com os estudos de caso realizados, assim como as lições aprendidas e sugestões de trabalhos futuros que podem ser realizados no contexto do ambiente proposto.

2 CONCEITOS BÁSICOS E REVISÃO DA LITERATURA

"*If I had an hour to solve a problem and my life depended on it, I would use the first 55 minutes determining the proper question to ask, for once I know the proper question, I could solve the problem in less than five minutes*."

Albert Einstein.

Neste capítulo, o processo de desenvolvimento de sistemas espaciais é apresentado sob a ótica de *Engenharia de Sistemas*. São mostradas as fases deste processo onde se aplicam as atividades de Modelagem e Simulação (M&S), assim como a importância do gerenciamento dos dados gerados por estas atividades e do reuso de modelos no ciclo de vida de desenvolvimento de produtos. Comenta-se também sobre os benefícios de se antecipar o início das atividades de M&S para as fases iniciais do processo de desenvolvimento.

É apresentado o conceito de *Engenharia Simultânea* e a sua aplicação nas chamadas *Concurrent Design Facilities* (CDF), que motivaram a criação do software *VirSat*. Por fim, são mostrados os conceitos básicos da tecnologia de SPDM e as principais funcionalidades oferecidas por softwares de SPDM.

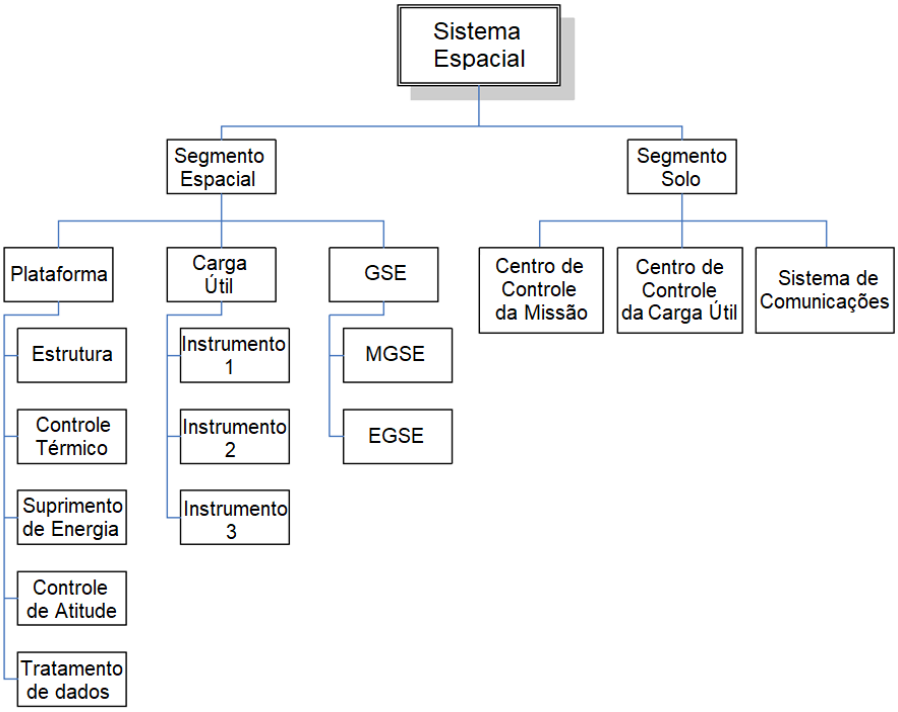
2.1. **Projeto de sistemas espaciais**

Segundo a norma ECSS-M-ST-10C, intitulada *Space project management - Project planning and implementation* (ECSS, 2009c), projetos espaciais são iniciados por governos, agências espaciais, comunidade científica e empresas privadas com propósito e objetivos definidos em uma declaração de missão. Por exemplo, o governo brasileiro pode ter a necessidade de monitorar o desmatamento da floresta amazônica. Considerando-se um conceito de operações para o atendimento desta necessidade envolvendo a área espacial, isto dá origem a uma missão espacial, que será desenvolvida por uma organização responsável como, por exemplo, o INPE.

Projetos espaciais geralmente são compostos de um Segmento Espacial e um Segmento Solo, que são implementados em paralelo. Estes segmentos se apoiam e têm interfaces com um segmento de lançamento. Em conjunto, os três segmentos mencionados constituem um sistema espacial (ECSS, 2009c).

Para um melhor gerenciamento de um projeto espacial, é realizada uma divisão do mesmo em elementos menores e mais fáceis de gerenciar, estruturados hierarquicamente, em formato de árvore. Assim, são criadas estruturas como a Estrutura de Divisão do Produto (do Inglês *Product Breakdown Structure*), a Estrutura de Divisão do Trabalho/Estrutura Analítica de Projetos (do Inglês *Work Breakdown Structure*) e a Estrutura de Divisão da Organização/Estrutura Analítica da Organização (do Inglês *Organizational Breakdown Structure*). A Estrutura de Divisão do Produto (EDP), mostrada na Figura 2.1, apresenta um exemplo de divisão do sistema espacial em dois segmentos, que são subdivididos em seus sistemas principais e subsistemas.

Figura 2.1 - Estrutura de Divisão do Produto (EDP).



Fonte: Adaptado de ECSS (2009c). (tradução nossa).

A execução de um projeto espacial envolve diversas atividades, distribuídas ao longo de fases que fazem parte do ciclo de vida do projeto espacial, mostradas na Tabela 2.1. De acordo com a ECSS (2009c), usualmente as fases 0, A e B são focadas na elaboração de requisitos técnicos e funcionais do sistema, identificação dos conceitos de sistemas aplicáveis à declaração de missão, identificação de atividades e recursos a serem usados no desenvolvimento dos sistemas, análises preliminares de risco e início das atividades de pré-desenvolvimento.

Tabela 2.1 – Fases do ciclo de vida de projetos espaciais.

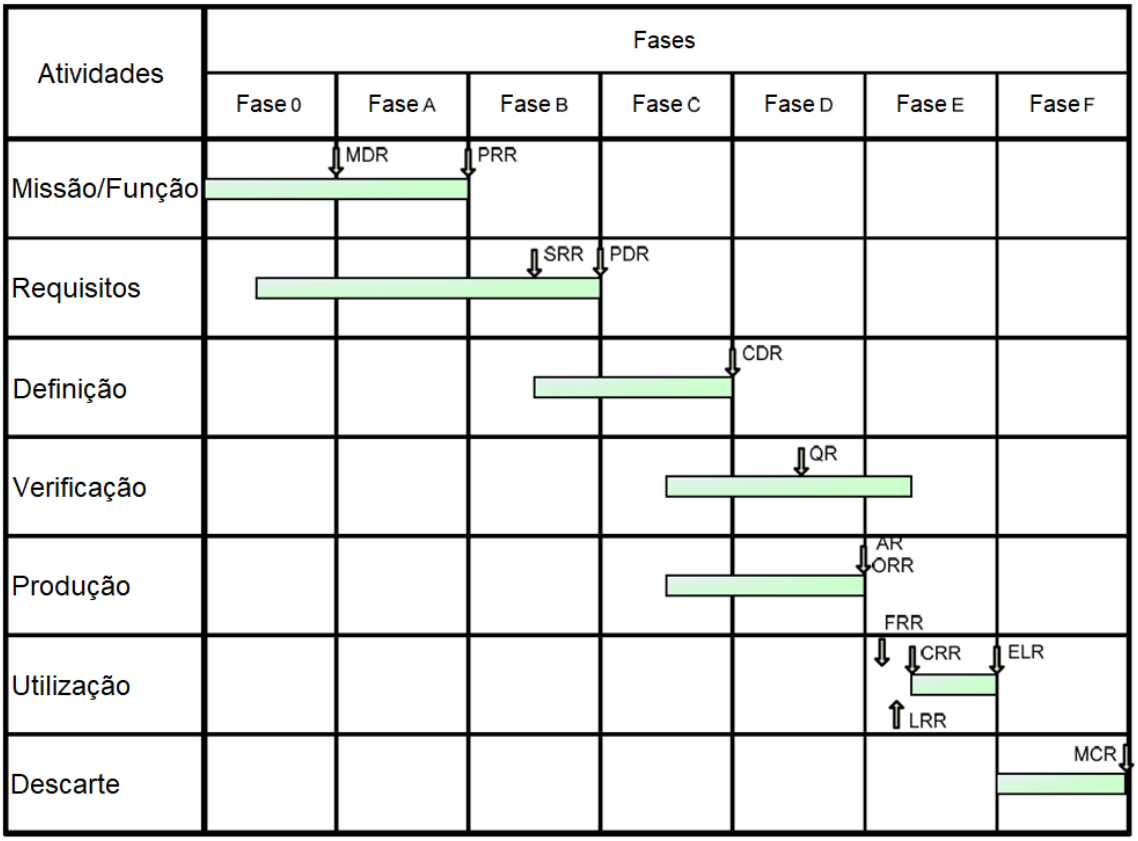
Fase do Projeto	Descrição
Fase 0	Análise da missão / Identificação de necessidades
Fase A	Viabilidade
Fase B	Definições preliminares
Fase C	Definições detalhadas
Fase D	Qualificação e Produção
Fase E	Utilização
Fase F	Descarte

Fonte: ECSS (2009c) (tradução nossa).

Nas fases C e D são executadas as atividades de desenvolvimento detalhado, produção e qualificação dos segmentos espacial e de solo. A fase E é constituída das atividades de lançamento, comissionamento, utilização, manutenção dos elementos orbitais do segmento espacial e utilização e manutenção do segmento solo correspondente. Na fase F, os segmentos espacial e de solo são descartados de forma segura (ECSS, 2009c).

Cada fase contém ainda marcos em forma de revisões de projeto cujo resultado determina a prontidão do projeto em prosseguir para a próxima fase. Nestas revisões de projeto são também registradas as linhas de base que contêm as configurações oficiais do sistema que servirão de referência para a próxima fase (ECSS, 2009c). A Figura 2.2 mostra a distribuição das revisões de projeto ao longo de suas fases, para um projeto típico. A Tabela 2.2 mostra a descrição destas revisões.

Figura 2.2 – Fases, atividades e revisões do ciclo de vida de um projeto espacial típico.



Fonte: Adaptado de ECSS (2009c). (tradução nossa).

Tabela 2.2 - Descrição das revisões de projeto.

Revisão	Descrição
MDR	Revisão de projeto da missão
PRR	Revisão de requisitos preliminar
SRR	Revisão de requisitos de sistema
PDR	Revisão de projeto preliminar
CDR	Revisão de projeto crítica
QR	Revisão de qualificação
AR	Revisão de aceitação
FRR	Revisão de prontidão para voo
ORR	Revisão de prontidão operacional
LRR	Revisão de prontidão para lançamento
CRR	Revisão de resultados de comissionamento
ELR	Revisão de fim de vida
MCR	Revisão de encerramento da missão

Fonte: ECSS (2009c). (tradução nossa).

Conforme descrito nesta seção, um projeto espacial é intrinsicamente multidisciplinar e sua coordenação exige um esforço interdisciplinar, que pode ser alcançado com processos de *Engenharia de Sistemas*.

2.2. **Engenharia de Sistemas**

Segundo a norma ECSS-E-ST-10C-Rev.1, intitulada *Space engineering - System engineering general requirements* (ECSS, 2017), *Engenharia de Sistemas* é “uma abordagem interdisciplinar que governa o esforço técnico total para transformar requisitos em uma solução de sistema”, sendo sistema definido como “um conjunto de elementos integrados para atingir um objetivo definido” (tradução nossa).

Segundo o *International Council on Systems Engineering* (INCOSE, 2016), *Engenharia de Sistemas* é “uma abordagem interdisciplinar e um meio para permitir a realização de sistemas bem-sucedidos.” Ele se concentra na definição das necessidades do interessado e das funcionalidades necessárias no início do ciclo de desenvolvimento, documentando os requisitos, prosseguindo com a síntese do projeto e a validação do sistema, considerando o problema completo, incluindo operações, custo, planejamento, desempenho, treinamento, suporte, teste, fabricação e retirada de serviço.

Ainda segundo o INCOSE (2016), *Engenharia de Sistemas* integra todas as disciplinas e grupos de especialidades em um esforço de equipe formando um processo de desenvolvimento estruturado que passa do conceito para a produção e em seguida para a operação. *Engenharia de Sistemas* considera as necessidades técnicas e de negócios de todos os interessados com o objetivo de fornecer um produto de qualidade, que atenda às necessidades do usuário.

De acordo com Schaus et al. (2010), no contexto da engenharia aeroespacial, satélites e veículos espaciais são sistemas complexos de projetar e operar. Várias disciplinas, como engenharia, gerenciamento de projetos e logística são envolvidas. Geralmente, projetos espaciais são conduzidos por vários parceiros

de diferentes nacionalidades, situados em diferentes países. Isto cria necessidades adicionais no gerenciamento de times, ferramentas e processos de trabalho. As várias disciplinas envolvidas no projeto de sistemas espaciais possuem interdependências. A mudança de um parâmetro de projeto em uma disciplina pode ter fortes efeitos em outras disciplinas.

Assim, torna-se importante a existência de processos e ferramentas que ajudem a gerenciar os dados, metadados, parâmetros de projeto e processos produzidos pelas diversas disciplinas durante o desenvolvimento de sistemas espaciais.

2.2.1. Processo da Engenharia de Sistemas

A norma ECSS-E-ST-10C-Rev.1 (ECSS, 2017) apresenta uma visão abrangente do processo de *Engenharia de Sistemas* no projeto de sistemas espaciais e define os requisitos que norteiam seu desenvolvimento. Segundo esta norma, o processo de *Engenharia de Sistemas* visa o atendimento das necessidades de um interessado com a entrega de um produto que satisfaça condições pré-determinadas de custo, prazo e qualidade. Este processo é composto de uma série de atividades, distribuídas ao longo das fases do ciclo de vida do produto espacial.

A relação entre o ciclo de vida de projetos e a disciplina de *Engenharia de Sistemas* é analisada no trabalho de Forsberg e Mooz (1991). Segundo estes autores, o ciclo de desenvolvimento para projetos geralmente se inicia com as necessidades de um usuário, que são traduzidas em um conjunto viável de requisitos de sistema. Estes requisitos são progressivamente decompostos em requisitos de subsistemas e componentes, até o nível mais baixo de detalhe, considerando hardware e software. Após a decomposição, vêm o projeto, o desenvolvimento, os testes, e a aceitação dos componentes. Após estas, vêm a montagem, a integração e os testes dos componentes em subsistemas que, por sua vez, são integrados, verificados e validados no sistema principal, até

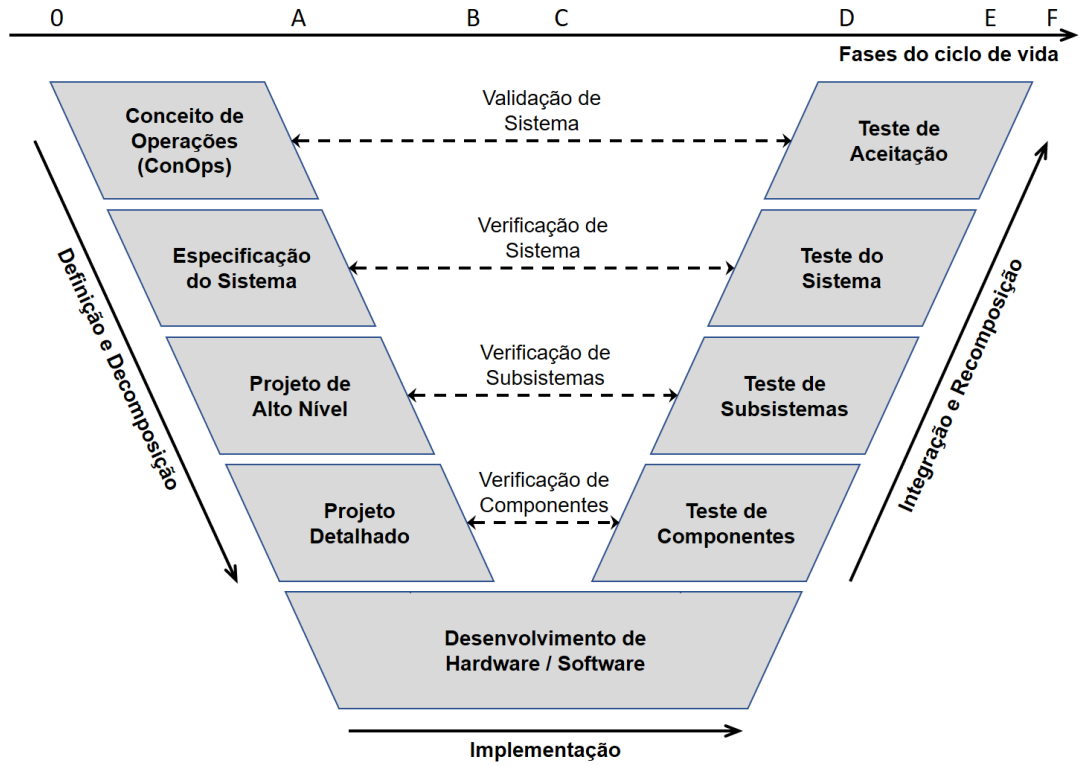
que o processo de integração esteja completo e evidenciado por um sistema validado e funcional.

Segundo o INCOSE (2015), modelos de ciclo de vida são úteis para definir o início, fim e as atividades de processo apropriadas para cada fase do ciclo de vida. Entretanto, modelos de ciclo de vida tradicionais, como o *Waterfall* (ROYCE, 1970) e o *Spiral* (BOEHM, 1986), não conseguem representar adequadamente o papel do processo de *Engenharia de Sistemas* e a participação do processo da *Engenharia Simultânea*. Para resolver este problema, Forsberg e Mooz (1991) propuseram a criação do modelo V, um método usado para visualizar várias áreas-chave de *Engenharia de Sistemas*, especialmente durante as fases de definições conceituais e desenvolvimento. O modelo V evidencia a necessidade de validação contínua com os interessados, a necessidade de definir planos de verificação durante o desenvolvimento de requisitos e a importância da contínua avaliação de riscos e oportunidades.

A Figura 2.3 ilustra o processo de *Engenharia de Sistemas* representado pelo modelo V. Neste modelo, tempo e maturidade do produto fluem da esquerda para a direita, com o processo iniciando na parte superior esquerda do V, com as atividades de definição do Conceito de Operações. Nesta etapa, são analisadas e documentadas as necessidades do interessado e formalizados os seus requisitos, com uma visão geral de papéis, responsabilidades e capacidades gerais do sistema (SHAMIEH, 2012). Um plano de validação é criado para que, na fase final do projeto, o desempenho do sistema possa ser adequadamente validado, respondendo-se à pergunta: “o desempenho verificado do sistema atende aos requisitos do interessado?” (FORSBERG; MOOZ,1991) (tradução nossa). Nota-se que, muitas vezes, apenas análises são possíveis, pois não é viável testar os sistemas nas condições reais de operação, como os satélites.

Na etapa de Especificação do Sistema, vários conceitos são propostos, testados e comparados entre si, até que um deles seja selecionado e sejam mapeados os requisitos de sistema que atendam aos requisitos de interessado identificados na etapa anterior. De acordo com Forsberg e Mooz (1991), é muito importante nesta etapa a aplicação da *Engenharia Simultânea* para envolver especialistas de todas as disciplinas relevantes ao projeto para a verificação dos requisitos que, se não for realizada adequadamente, pode dar origem a um produto que, futuramente, possa ter dificuldades de ser manufaturado, inspecionado ou mantido.

Figura 2.3 – Processo de *Engenharia de Sistemas.*



Fonte: Adaptado de Shamieh (2012). (tradução nossa).

É recomendado o uso de M&S nesta etapa, tanto para a verificação dos requisitos, quanto para a proposição e análise de conceitos de sistemas para o atendimento das necessidades do interessado. Em assim se procedendo, antecipa-se o atingimento da maturidade do produto e reduz-se a necessidade

de revisões de requisitos e mudanças futuras. Isto diminui também os custos do projeto, pois o custo de mudanças no produto é cada vez maior, à medida que o projeto avança. Nesta etapa é criado também um plano de verificação do sistema, que será utilizado em uma etapa posterior, para responder à pergunta: “o desempenho verificado do sistema atende às especificações do sistema?” (FORSBERG; MOOZ, 1991) (tradução nossa).

Na etapa de Projeto de Alto Nível, é criada uma arquitetura de alto nível que descreve o sistema como um conjunto de subsistemas. É realizada uma decomposição funcional multidisciplinar dos requisitos de sistema para a criação dos requisitos dos subsistemas (ECSS, 2017). Um plano de verificação dos subsistemas é criado para que seja possível responder, em uma etapa posterior, ao questionamento: “o desempenho verificado dos subsistemas atende às especificações dos subsistemas?” (FORSBERG; MOOZ, 1991) (tradução nossa). Nota-se que, muitas vezes, apenas análises são possíveis, pois não é viável testar os sistemas nas condições reais de operação, como os satélites.

Na etapa de Projeto Detalhado, os subsistemas são decompostos em seus componentes mais elementares, com a criação de requisitos de componentes coerentes com os requisitos dos subsistemas. Um plano de verificação dos componentes é criado para que seja possível responder, em uma etapa posterior, ao questionamento: “o desempenho verificado dos componentes atende às especificações dos componentes?” (FORSBERG; MOOZ, 1991) (tradução nossa).

O lado esquerdo do V, da primeira etapa até a de Projeto Detalhado, é denominado “Definição e Decomposição” de acordo com a Figura 2.3. Ao longo de cada uma das etapas deste lado do V, é realizado um progressivo desdobramento de requisitos, iniciando-se com os requisitos do interessado, passando pelos requisitos de sistema, subsistemas e seus componentes. A responsabilidade pela execução destas etapas é da função de *Engenharia de*

Sistemas, com o apoio da Engenharia de Desenvolvimento. A função de *Engenharia de Sistemas* pode ou não ser formalmente exercida por um departamento da organização responsável pelo projeto espacial ou por um de seus fornecedores (ECSS, 2017). Nenhum desenvolvimento propriamente dito de sistema é realizado neste lado do V. Apenas são detalhados todos os requisitos necessários para o desenvolvimento do produto e descritas as formas de verificação e validação destes requisitos.

Na base do V tem-se a etapa de Desenvolvimento de Hardware e Software, sob responsabilidade da Engenharia de Desenvolvimento, que desenvolve o produto sob auditoria e de acordo com as especificações da função de *Engenharia de Sistemas*.

O lado direito do V, da etapa de Teste de Componentes até a etapa final, é chamado de “Integração e Recomposição”, de acordo com a Figura 2.3. Ao longo das etapas deste lado do V, o sistema é testado desde seus componentes mais básicos, passando pelos testes das montagens dos componentes em subsistemas e dos testes da montagem dos subsistemas no sistema principal.

Na etapa de Teste de Componentes, a funcionalidade de cada componente é testada de acordo com o plano de verificação de componentes criado na etapa de Projeto Detalhado. Na etapa de Teste de Subsistemas, os componentes são integrados em subsistemas e as funcionalidades de cada subsistema são testadas de acordo com o plano de verificação de subsistemas criado na etapa de Projeto de Alto Nível. Na etapa de Teste do Sistema, os subsistemas são integrados no sistema principal e as funcionalidades do sistema são testadas de acordo com o plano de verificação do sistema criado na etapa de Especificação do Sistema. Na etapa de Teste de Aceitação, as funcionalidades do sistema são validadas de acordo com o plano de validação do sistema criado na etapa de Conceito de Operações. Nesta etapa final, é realizada a

validação que o sistema cumpre os requisitos do interessado e é efetivo no atingimento dos objetivos esperados (SHAMIEH, 2012).

Segundo Forsberg e Mooz (1991), é importante notar a diferença entre os conceitos de verificação e validação no modelo V. Segundo estes autores, “Verificação é o processo de provar que cada produto cumpre com sua especificação”, enquanto que “Validação é o processo de demonstrar (e não provar) que o produto satisfaz as necessidades do interessado, independentemente do que a especificação do produto requer” (tradução nossa). Mas, segundo Souza (2017), Verificação (=correção técnica) é a prova do atendimento das verdades/boas práticas das áreas de conhecimento envolvidas; Conformidade (=satisfação burocrática) é a prova do atendimento das especificações do sistema; Validação (=adequação jurídica) é a prova do atendimento dos requisitos dos interessados; Qualidade (=satisfação psicológica) é a prova do atendimento das necessidades dos interessados.

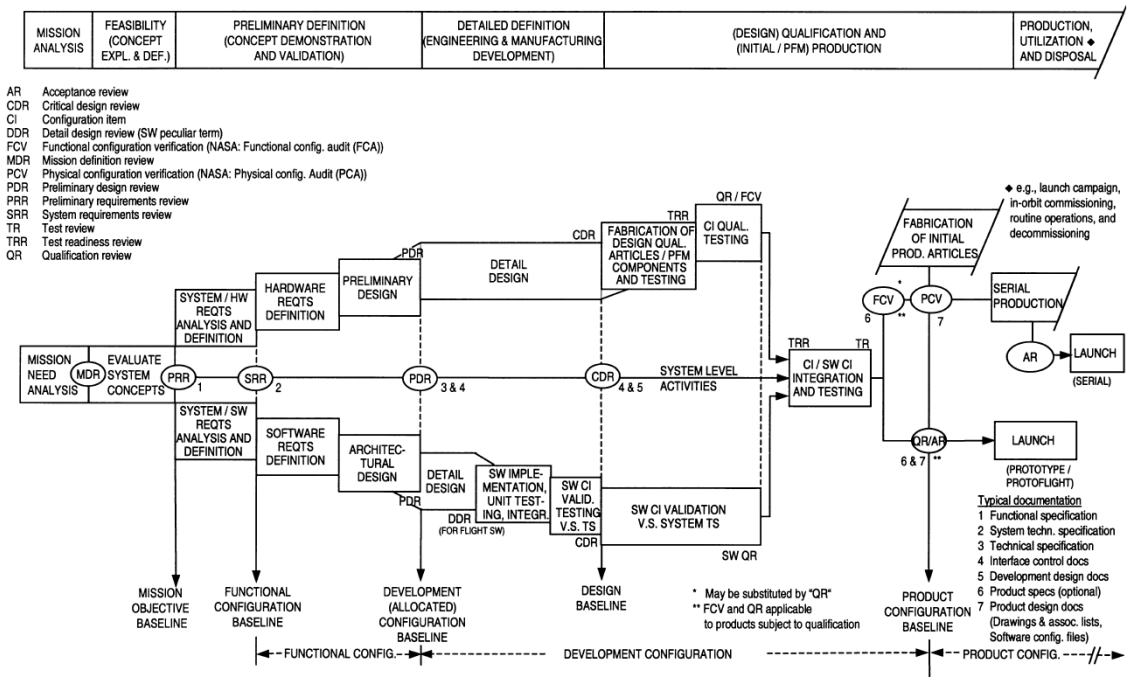
2.3. **Gerenciamento de linhas de base**

Conforme visto na Seção 2.1, ao longo das fases de um projeto espacial têm-se várias revisões de projeto onde a configuração do sistema é consolidada e registrada, tornando-se linhas de base para fases seguintes. A forma de gerenciamento da configuração de sistemas em projetos espaciais é descrita em detalhes na norma ECSS-M-ST-40C_Rev.1, intitulada “*Space project management: Configuration and information management*” (ECSS, 2009b).

As linhas de base descritas por esta norma, mostradas na Figura 2.4, são:

a) **Linha de base do objetivo da missão (MOB):** estabelecida na Revisão de Requisitos Preliminar (PRR) com base na especificação funcional aprovada. Estabelece o propósito do sistema, suas restrições e ambientes associados, as capacidades operacionais e de desempenho para cada fase do seu ciclo de vida e a flexibilidade permitida;

Figura 2.4 – Definições das fases de um projeto espacial e linhas de base correspondentes.



Fonte: ECSS (2009b).

b) **Linha de base de configuração funcional (FCB):** estabelecida na Revisão de Requisitos de Sistema (SRR) com base na especificação técnica do sistema aprovado. Estabelece as características do sistema em termos de seus requisitos técnicos, bem como os critérios e níveis correspondentes de qualificação e aceitação;

c) **Linha de base de configuração de desenvolvimento (DCB):** estabelecida na Revisão de Projeto Preliminar (PDR) com base em especificações técnicas aprovadas. Estabelece as características do produto em termos de requisitos técnicos e restrições de projeto, bem como suas condições de verificação;

d) **Linha de base de projeto (DB):** estabelecida na Revisão de Projeto Crítica (CDR) com base na documentação de projeto aprovada;

e) **Linha de base de configuração do produto (PCB):** estabelecida com base no conjunto aprovado de documentos contendo todas as características físicas e funcionais necessárias para produção, aceitação, operação, suporte e eliminação.

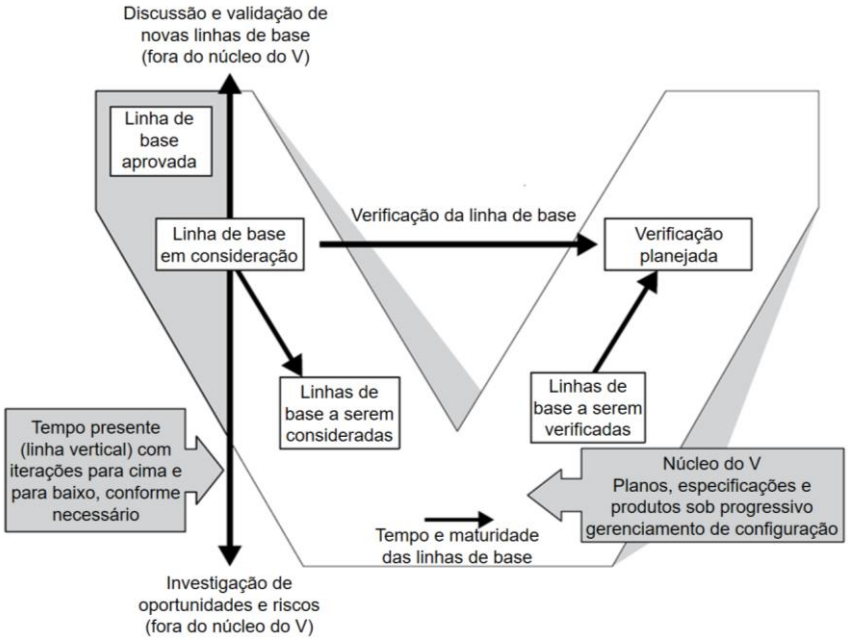
A determinação destas linhas de base é coerente com o trabalho de Forsberg e Mooz (1991), segundo o qual o modelo V do processo de *Engenharia de Sistemas* possui diversas linhas de base distribuídas ao longo das etapas do V. O INCOSE (2015) mostra como as linhas de base são determinadas em cada lado do V.

Conforme mencionado na seção 2.2.1, tempo e maturidade fluem da esquerda para a direita no modelo V. Na Figura 2.5, o núcleo do V contém uma sequência de linhas de base em grau crescente de maturidade. Em um dado instante de tempo, uma linha de base em consideração, baseada em definições de uma linha de base já aprovada (mostrada à esquerda do V, dentro da área sombreada), participa de iterações acima e abaixo, fora do núcleo do V. Nas iterações abaixo, a equipe de desenvolvimento investiga opções de solução de sistema, testando novos conceitos em níveis mais baixos de detalhe e balanceando riscos, visando minimizá-los, para que o desempenho esperado pelos interessados seja atendido. Nas iterações acima, os novos conceitos estudados são validados pelos interessados e demais influenciadores, para garantir que a linha de base proposta é aceitável. Uma vez que a linha de base seja aceita, ela é incorporada ao núcleo do V (como resultado de uma revisão de projeto, por exemplo) e o processo continua com novas linhas de base a serem consideradas, em níveis maiores de detalhe.

É importante mencionar que, segundo Forsberg e Mooz (1991), embora vários estudos sejam realizados fora do núcleo do V, somente linhas de base aprovadas e incorporadas ao núcleo do V são colocadas em controle de configuração ao final de cada revisão de projeto. Estudos, análises, modelos e

demais artefatos não diretamente associados à linha de base aprovada não são formalmente controlados.

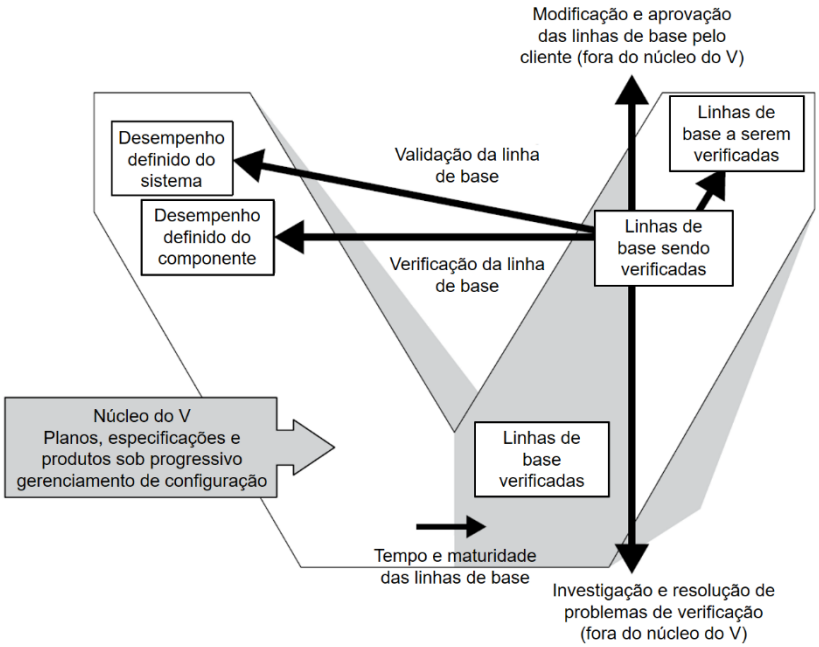
Figura 2.5 – Definição de linhas de base no lado esquerdo do modelo V.



Fonte: Adaptado de INCOSE (2015). (tradução nossa).

No lado direito do modelo V, mostrado na Figura 2.6, à medida que o sistema é integrado e verificado, as iterações para cima e para baixo continuam acontecendo para as linhas de base que estão sendo verificadas. Após a verificação do desempenho de uma determinada linha de base, possíveis problemas são investigados e resolvidos nas iterações abaixo, onde o nível de detalhe é maior, e possíveis modificações são propostas e aprovadas pelo interessado e demais influenciadores, nas iterações acima, onde o nível de detalhe é menor. Estas iterações são mostradas verticalmente pois, como o tempo e maturidade fluem da esquerda para a direita no modelo V, não é possível retroceder. Uma vez aprovada a linha de base em uma das revisões de projeto, ela é incorporada ao núcleo do V e torna-se referência para a próxima linha de base que será analisada em seguida.

Figura 2.6 - Definição de linhas de base no lado direito do modelo V.



Fonte: Adaptado de INCOSE (2015). (tradução nossa).

2.4. **Modelagem e Simulação de sistemas espaciais**

Segundo o memorando técnico ECSS-E-TM-10-21A, intitulado *Space engineering - System modeling and simulation* (ECSS, 2010a), atividades de M&S são utilizadas no projeto de sistemas espaciais para suportar os processos de definição de requisitos, projeto detalhado, verificação e operação. Tradicionalmente, M&S tem sido considerada como uma disciplina de apoio, aplicada em várias áreas do projeto espacial de forma independente e não coordenada. Entretanto, o uso de M&S de forma coerente ao longo do ciclo de vida do projeto traz ganhos significativos ao reduzir riscos e custos do projeto.

2.4.1. Termos utilizados neste trabalho

A Tabela 2.3contém as definições de termos da disciplina de M&S utilizados neste trabalho, descritos pela ECSS (2010a).

Tabela 2.3 – Termos de M&S utilizados neste trabalho.

Termo	Definição
Modelo de simulação	Por modelo de simulação entende-se tanto modelos de dados, como modelos geométricos, quanto modelos de comportamento, como algoritmos que representam o comportamento de um componente representados em uma linguagem de programação de alto nível. Modelos de simulação geralmente têm entradas, saídas, variáveis internas de estado e constantes.
Modelagem	Método usado para analisar e descrever o comportamento de um modelo. Técnicas de modelagem típicas são a modelagem física, comportamental, funcional e geométrica.
Simulador	Combinação de um ou mais modelos de simulação que são acoplados e executados em conjunto para representar o comportamento de um sistema.
Cenário	Configuração inicial particular de um simulador, representando uma parte de uma missão espacial.
Simulação	Execução de um cenário em um simulador com tempos de início e fim bem definidos.
Ambiente de simulação	Infraestrutura de software que é usada para executar os modelos de simulação.

Fonte: ECSS (2010a). (tradução nossa).

2.4.2. Modelagem e Simulação no ciclo de vida de sistemas espaciais

Segundo a ECSS (2010a), M&S de sistemas deve ser uma parte integral do processo de *Engenharia de Sistemas,* para que se extraia o máximo benefício de sua utilização. Atividades de M&S de sistemas estão distribuídas ao longo das fases do ciclo de vida de sistemas espaciais, conforme mostrado na Tabela 2.4.

Tabela 2.4 – Simulação de sistemas no ciclo de vida de sistemas espaciais.

	Ciclo de vida do sistema espacial						
Atividades de engenharia e operações	**Fase 0**	**Fase A**	**Fase B**	**Fase C**	**Fase D**	**Fase E**	**Fase F**
Análises de viabilidade e desempenho	Atividades de engenharia simulânea						
Especificação de Requisitos	Atividades de engenharia simulânea	Análise do sistema e da missão					
Verificação de Projeto		Interfaces de sistema e *trade-offs* de projeto					
Verificação de		Interfaces de sistema e					

sistemas e desempenho da missão		*trade-offs* de projeto					
Verificação e Validação funcional (subsistemas)			Interfaces	Integração e Verificação da Montagem. Software de bordo			
Qualificação e Aceitação do veículo espacial				Integração e Verificação Virtuais da Montagem	Integração e Verificação da Montagem. Testes de Validação do sistema		
Qualificação e Aceitação do segmento solo				Teste do sistema de controle da missão	Validação do procedimento de operações. Testes de Validação do sistema		
Qualificação e Aceitação do sistema					Integração e Verificação da Montagem. Testes de Validação do sistema	Manutenção do sistema	
Treinamento e Operações					Treinamento do time de controle da missão	Treinamento continuado do time de controle da missão. Procedimento de validação de alterações no software de bordo. Investigação e resolução de anomalias	Investigação de opções para descarte

Fonte: Adaptado de ECSS (2010a). (tradução nossa).

Dentre as atividades mostradas na Tabela 2.4, destacam-se, no contexto deste trabalho, para as fases 0 e A:

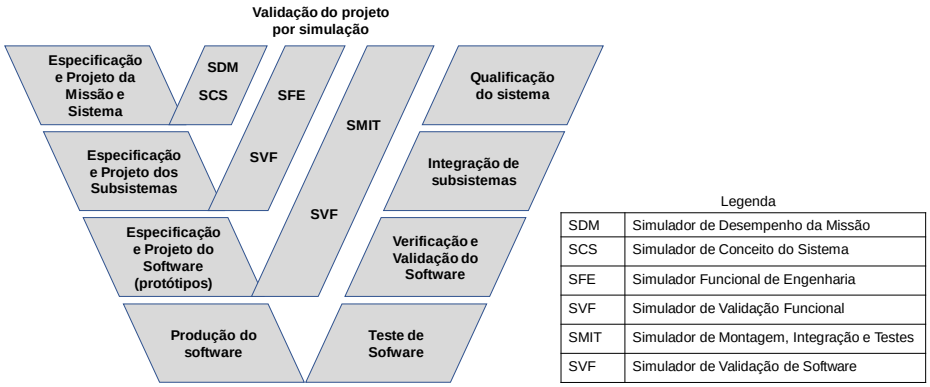
a) Análises de viabilidade e desempenho: atividades de engenharia simultânea;

b) Especificação de requisitos: atividades de engenharia simultânea e análise do sistema e da missão;

c) Verificação do projeto, sistemas e desempenho da missão: análise de interfaces de sistema e *trade-offs* de projeto.

Segundo a ECSS (2010a), atividades de M&S de sistemas podem ser utilizadas para antecipar os resultados dos processos de verificação de sistema, subsistemas e componentes. Estes processos geralmente ocorrem no lado direito do modelo V, quando os componentes, subsistemas e sistema já foram desenvolvidos e protótipos foram fabricados. Entretanto, o comportamento do sistema como um todo e o de suas partes podem ser modelados e os planos de verificação e validação podem ser executados por meio de simulação destes modelos. Assim, podem ser descobertas não-conformidades em etapas anteriores ao desenvolvimento detalhado do sistema.

Figura 2.7 – Papel de M&S no processo de *Engenharia de Sistemas.*

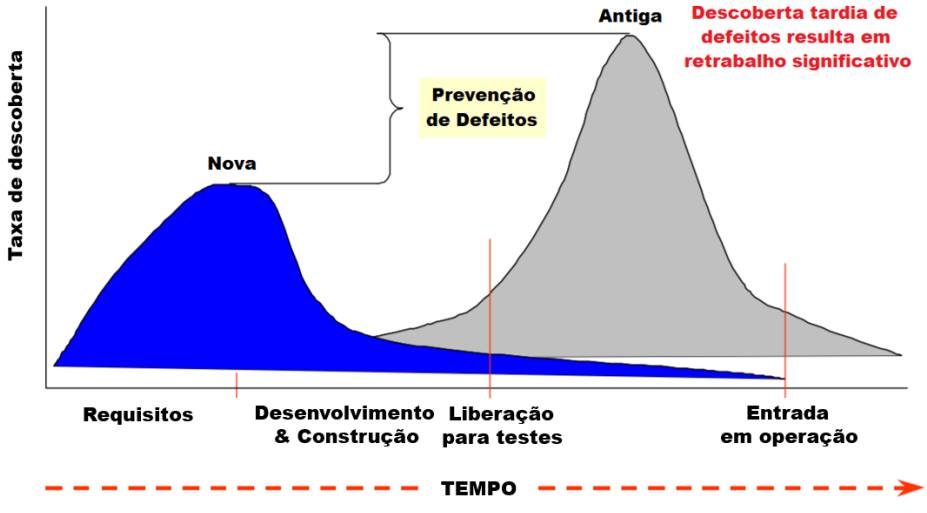


Fonte: Adaptado de ECSS (2010a). (tradução nossa).

Graficamente, este processo pode ser representado no modelo V por prolongamentos do lado esquerdo do V, em diferentes níveis de maturidade, paralelos ao seu lado direito, como mostrado na Figura 2.7.Segundo Blackburn et al. (2001), o uso de M&S nas fases iniciais de projeto, onde os requisitos estão sendo definidos, acelera a taxa de descoberta de defeitos e isto é benéfico, pois nestas fases o custo de alterações de projeto é menor. Assim, após a fase de desenvolvimento detalhado e início dos testes físicos, o produto apresenta bem menos defeitos e permite uma entrada em operação praticamente sem defeitos a serem corrigidos, o que está representado pela curva azul da Figura 2.8. A curva cinza mostrada na mesma figura mostra o

significativo retrabalho causado pela descoberta tardia de defeitos devido à baixa utilização de M&S nas fases iniciais do projeto.

Figura 2.8–Identificação e prevenção antecipada de defeitos.



Fonte: adaptado de Blackburn et al. (2001). (tradução nossa).

2.5. **Engenharia Simultânea**

O termo *Concurrent Engineering* (Engenharia Simultânea) foi cunhado em 1988 no Relatório R-338 do *Institute for Defence Analysis:*

> A engenharia simultânea é uma abordagem sistemática para o projeto integrado e concorrente de produtos e seus processos relacionados, incluindo fabricação e suporte. Esta abordagem tem como intenção incentivar os desenvolvedores a considerar, desde as etapas iniciais do desenvolvimento de um produto, todos os elementos do seu ciclo de vida, desde a concepção até o descarte, incluindo a qualidade, o custo, o planejamento e os requisitos dos usuários (WINNER, 1988) (tradução nossa).

A ECSS, no memorando técnico ECSS-E-TM-10-25A, intitulado "*Space engineering - Engineering design model data exchange (CDF)*", apresenta a seguinte definição de Engenharia Simultânea:

> Engenharia simultânea é uma abordagem sistemática para o desenvolvimento de sistemas multidisciplinares integrados que enfatiza a resposta às expectativas dos interessados e incorpora valores de equipe de cooperação, confiança e compartilhamento de tal forma que a tomada de decisões seja por consenso, envolvendo todas as perspectivas em paralelo desde o início do ciclo de vida do sistema (ECSS, 2010b) (tradução nossa).

Segundo Schaus et al. (2010), a Engenharia Simultânea trouxe grande melhoria para o projeto inicial de missões e estudos conceituais. Ela reúne vários especialistas em um time multidisciplinar, o que permite rápidas interações e troca de dados de projeto, resultando em uma sólida concepção inicial do sistema. A colaboração faz com que todos fiquem cientes de mudanças no projeto e assegura que todos estejam trabalhando com as revisões mais recentes dos parâmetros de projeto.

2.5.1. Concurrent Design Facilities (CDFs)

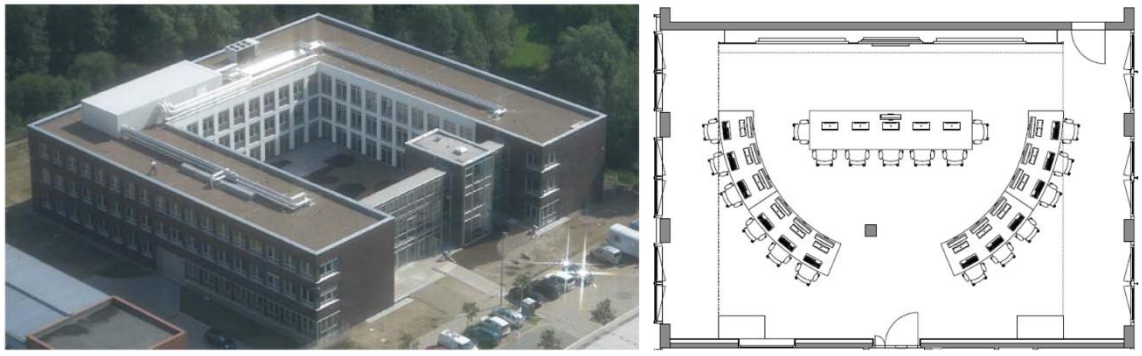
Segundo a ECSS (2010b), desde 1996 a Agência Espacial Europeia (ESA) tem utilizado o conceito de *Concurrent Design Facility* (CDF) para conduzir sessões de engenharia simultânea nas fases 0 e A do desenvolvimento de sistemas espaciais. Cada CDF é composta por um time multidisciplinar de engenheiros, distribuídos fisicamente em um ambiente propício à interação.

Para facilitar a colaboração e o compartilhamento de dados entre os membros do time, a ESA criou um modelo de dados padrão, chamado de *Integrated Design Model* (IDM). Com o sucesso do conceito de CDF, muitas organizações europeias começaram a adotá-lo. Segundo Chagas (2016), o INPE também

desenvolveu uma infraestrutura similar, denominada “Centro de Projeto Integrado de Missões Espaciais” (CPRIME). Como cada organização tem suas particularidades, naturalmente foram realizadas modificações no IDM, ou mesmo criados novos modelos de dados, para se adaptar a elas.

Segundo Seider et al. (2012), este foi o caso do DLR que, em 2008, iniciou um programa nacional de desenvolvimento de satélites e criou o *Concurrent Engineering Facility* (CEF), nos moldes do CDF. Para apoiar o trabalho do CEF, o DLR também iniciou o desenvolvimento do software *Virtual Satellite*. O CEF foi criado para facilitar a comunicação, realização de apresentações e o trabalho científico simultâneo. Para isto, conta com doze estações de trabalho dispostas em semicírculo e cinco estações adicionais no centro de forma a facilitar o contato visual e a interação entre os membros da equipe, o que é importante nos estágios iniciais de discussão de sistemas espaciais. Esta disposição é mostrada na Figura 2.9.

Figura 2.9 - Instituto de Sistemas Espaciais do DLR em Bremen (esquerda) e *layout* do CEF (direita).



Fonte: Schumann et al. (2008b).

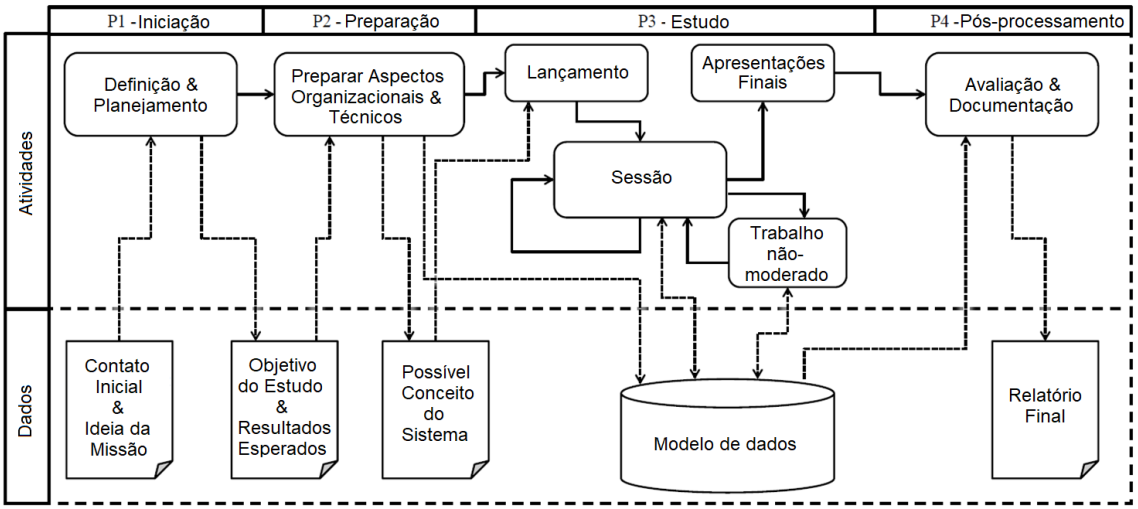
O objetivo destas discussões iniciais é obter uma visão geral do sistema espacial que será desenvolvido, cujas variáveis importantes incluem peso, custos, potência requerida para os sistemas e capacidade de armazenamento e transmissão de dados. Os engenheiros seguem um processo padrão de engenharia simultânea e obtêm um conjunto de soluções que levam em consideração *trade-offs* para a missão, em um período de algumas semanas.

Entretanto, para que o trabalho flua com mais agilidade, é fundamental a existência de um modelo de dados integrado e de uma ferramenta de apoio ao processo compatível com este modelo (SEIDER et al., 2012). Mais detalhes sobre o modelo de dados e da ferramenta de apoio utilizadas pelo CEF serão mostrados na seção 4.4.

2.5.2. Processo de Engenharia Simultânea

O processo de engenharia simultânea no DLR é mostrado no trabalho de Fischer et al. (2016a), ilustrado na Figura 2.10. Este processo é iniciado com uma atividade de definição e planejamento, que utiliza dados iniciais e uma proposta de missão para definir melhor os objetivos e resultados esperados da missão. Em seguida, são preparados os aspectos técnicos e organizacionais do sistema, gerando um possível conceito do sistema. São então realizadas várias sessões de engenharia simultânea com equipes multidisciplinares utilizando um modelo de dados padrão e um software de apoio, escrevendo os dados resultantes em um repositório de dados central. As sessões são encerradas com apresentações dos resultados intermediários, que posteriormente são documentados e compilados em um relatório final.

Figura 2.10 – Processo de engenharia simultânea do DLR.



Fonte: Adaptado de Fischer et al. (2016a). (tradução nossa).

2.6. **Simulation Process and Data Management (SPDM)**

A tecnologia de SPDM foi introduzida na Seção 1.1, onde foi mencionado que, devido ao aumento da capacidade computacional, intensificou-se o uso de M&S, o que resultou em um volume de dados difícil de administrar sem o uso de um sistema especializado. Isto motivou a criação de sistemas dedicados ao gerenciamento de dados e processos de simulação.

2.6.1. Conceitos básicos

Processos de simulação de engenharia envolvem o encadeamento de várias atividades, cada uma composta de diversos passos. As decisões de engenharia realizadas em cada atividade devem ser consistentes entre si e estar em acordo com os requisitos gerais de desenvolvimento do sistema. Estando estas decisões apoiadas em resultados de simulações, é importante que tenham sido utilizadas hipóteses e dados de entrada válidos para estas simulações. Assim, é importante que sejam registrados todos os dados destas simulações, sejam de entrada ou saída, não apenas para embasar a tomada de decisão como também para a captura da propriedade intelectual gerada. O registro apropriado também permite a rápida investigação de problemas que venham a surgir no futuro, em situações como a execução de testes físicos ou mesmo durante a operação (NORRIS, 2012).

Com o aumento progressivo do número de simulações, empresas de engenharia precisam estar continuamente atualizando sua infraestrutura de computação de alto desempenho (HPC), armazenamento de dados e de rede. Com isso, o trabalho que os engenheiros têm de registrar os dados utilizados em suas análises tende a crescer cada vez mais, prejudicando sua produtividade. Tradicionalmente, este registro é feito de forma manual, utilizando-se relatórios, planilhas, memoriais de cálculo ou outros tipos de documentos armazenados como arquivos no repositório de dados da empresa. Antigamente, este registro era realizado em relatórios impressos. A Figura 2.11 mostra o exemplo do gerenciamento de dados de apenas um dos veículos

espaciais do programa Apollo. Se uma falha fosse descoberta neste veículo, uma busca poderia ser realizada nesta documentação para identificar a origem do problema e verificar se a mesma falha poderia estar presente em outros veículos que compartilham as mesmas soluções de projeto (NORRIS, 2012).

Figura 2.11 – Exemplo de gerenciamento de dados de um veículo espacial.

Fonte: Norris (2012).

Depender de procedimentos manuais para o registro dos dados de projeto de sistemas durante seu desenvolvimento não é mais uma opção para empresas de engenharia que precisam permanecer competitivas no mercado. Para eliminar este trabalho burocrático e focar na engenharia propriamente dita, estas empresas precisam adotar sistemas de SPDM que automatizam a coleta e o registro de informações de projeto (NORRIS, 2012).

Quando o sistema SPDM não está presente, o engenheiro interage diretamente com o ambiente de simulação. Ele, por exemplo, faz *login* em uma estação de trabalho, procura o software que deve executar, procura os dados de entrada e executa o software, seja por meio de uma interface gráfica ou de uma linha de comando com argumentos especificados pelo engenheiro. Depois ele copia os resultados da simulação para uma outra pasta do sistema e escreve um relatório especificando o local onde os dados de entrada e saída estão localizados no repositório de dados da empresa, quais versões de quais programas foram executados, e quais as conclusões da análise.

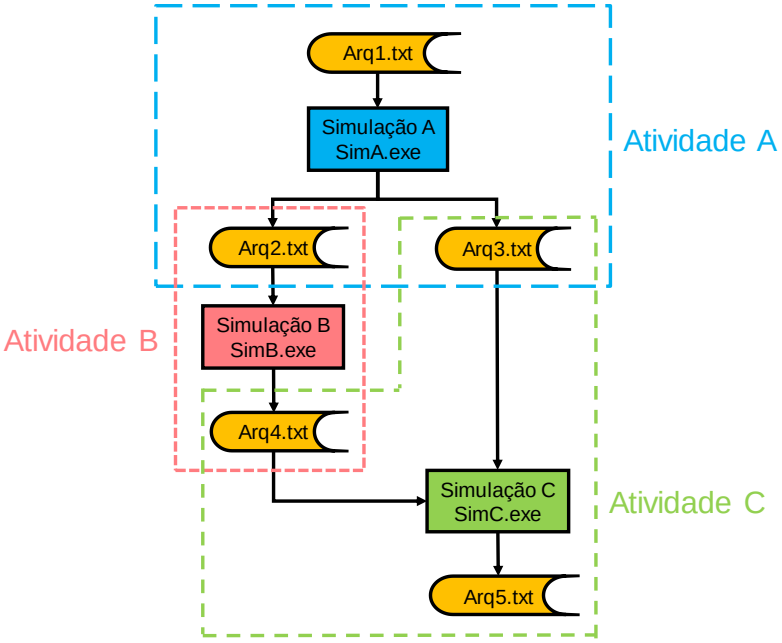
Quando sistemas de SPDM estão implantados na empresa, o procedimento de trabalho é diferente. Estes sistemas atuam como um intermediário entre o engenheiro e o ambiente de simulação. Idealmente, os processos de simulação tradicionalmente executados na empresa já foram mapeados e estão implementados na forma de fluxos de trabalho (*workflows*) dentro do sistema de SPDM. O engenheiro faz, então, seu *login* no sistema de SPDM, seleciona um destes *workflows*, fornece os dados de entrada e o executa. Automaticamente, o sistema de SPDM faz *login* nas estações de trabalho onde residem os programas contidos no *workflow*, transfere os arquivos de entrada, executa os programas na sequência apropriada, importa os dados de saída e cria os metadados de relacionamento entre dados de entrada, saída e programas executados, de forma que possam ser consultados posteriormente. Este relacionamento é chamado, dentro da tecnologia SPDM, de *simulation audit trail* (“rastros” para auditoria de simulação), *data provenance* (linhagem de dados) ou, de forma mais comum, *pedigree* de dados.

A Figura 2.12 mostra um exemplo de um processo de simulação com três atividades (A, B e C), cada uma representada por um software de simulação correspondente. Na atividade A são gerados dois arquivos que são usados como entrada para as atividades B e C. A atividade C também usa o arquivo de saída da atividade B como arquivo de entrada. É possível visualizar graficamente que o arquivo de saída do processo, “Arq5.txt”, é resultado do processamento de outros quatro arquivos por três softwares de simulação. Ao executar o processo em um sistema SPDM, o *pedigree* mostrado na figura é automaticamente capturado pelo sistema.

Metadados são dados referente a outros dados. Por exemplo, o nome do usuário que criou um arquivo, sua data de criação, nome do software utilizado e o seu *pedigree*. A grande vantagem de se utilizar sistemas de SPDM é que estes e outros metadados são automaticamente capturados durante a execução do processo de simulação e o engenheiro não precisa se dar ao

trabalho de registrá-los, focando seu tempo no trabalho de analisar os resultados da simulação.

Figura 2.12 – *Pedigree* de dados de um processo de simulação com três atividades.



Fonte: Produção do autor.

Segundo o NAFEMS (2016), “a tecnologia de *Simulation Data Management* (SDM) permite o gerenciamento de dados e metadados em todos os níveis de granularidade e abstração, incluindo parâmetros de design e análise, requisitos e resultados” (tradução nossa). A tecnologia de SPDM estende o conceito de SDM ao incorporar o gerenciamento dos processos que geram os dados e metadados.

A tecnologia de SPDM ainda é recente e está em desenvolvimento por vários fabricantes de software de simulação. Segundo o NAFEMS (2016), a visão para o desenvolvimento desta tecnologia é que:

> As soluções da tecnologia SPDM estão integradas a outros sistemas e bancos de dados que gerenciam dados de engenharia e de produto. Sistemas SPDM estão disponíveis para gerenciar e executar uma ampla gama

> de dados e processos de modelagem e simulação, em toda a abrangência de disciplinas de engenharia, suportando ambientes de análise heterogêneos. A integração de dados de projeto, teste e fabricação com modelagem, simulação e análise é suportada por sistemas SPDM (NAFEMS, 2016) (tradução nossa).

2.6.2. Principais desafios em SPDM

Segundo Larson e Wertz (1999), o desenvolvimento de sistemas complexos na área aeroespacial envolve a integração entre várias disciplinas, que possuem forte dependência entre si. Na área espacial, por exemplo, a massa total de um sistema a ser lançado ao espaço influencia a escolha do veículo lançador que, por sua vez, restringe as dimensões da estrutura a ser transportada.

Nas fases iniciais deste desenvolvimento, muitos estudos são realizados e os dados correspondentes ficam distribuídos em várias plataformas. As trocas de informações entre equipes multidisciplinares são frequentes. Na falta de um ambiente colaborativo e integrado onde os engenheiros possam centralizar os dados e ferramentas utilizados, as trocas geralmente são realizadas de maneira informal, como *e-mails* e unidades de rede compartilhadas.

Muitas vezes, a falta de um processo formal de publicação, em um ambiente comum, das versões mais recentes das informações dos sistemas pode resultar no uso incorreto de versões desatualizadas destas informações por outras equipes que dependem das mesmas. Para que haja um gerenciamento eficaz das informações, cada estudo deve ser versionado, assim como as ferramentas utilizadas e os dados gerados por elas.

As ferramentas de desenvolvimento de sistemas espaciais são geralmente especializadas para atender às necessidades de um campo específico de uma disciplina, sendo dominadas apenas por um pequeno grupo de especialistas. Frequentemente, estas ferramentas requerem plataformas de *software* ou

sistemas operacionais diferenciados, ou dependem de licenças dispendiosas, que não podem ser instaladas em muitos computadores.

Apesar disso, para que haja integração entre as disciplinas, é necessário que estas ferramentas sejam integradas, de forma que os dados de saída gerados por uma possam ser usados como dados de entrada por outra. Quando as ferramentas estão dispersas, instaladas em apenas alguns computadores do time de desenvolvimento e disponível apenas a algumas pessoas, esta integração é dificultada.

Quando o desenvolvimento de um novo sistema envolve várias equipes de uma ou mais instituições, além de fornecedores externos, uma dificuldade adicional é o controle do acesso aos dados gerados. Por exemplo, no desenvolvimento de um determinado subsistema de um satélite, apenas as equipes envolvidas devem ter privilégios de edição dos dados e atualização das ferramentas. Por outro lado, pode ser necessário que alguns dados comuns deste subsistema devam ficar visíveis a todas as equipes e dados confidenciais sejam liberados apenas a um grupo seleto de pessoas.

Para a execução de simulações, o engenheiro precisa ter à sua disposição diferentes tipos de dados, que geralmente estão armazenados em sistemas segregados. Dentre estes dados, podem-se citar dados de requisitos, dados de definição de produto (CAD), dados de propriedades de materiais e dados de resultados de ensaios físicos. Entretanto, alguns tipos de dados são específicos da área de M&S e acesso a eles é necessário apenas aos profissionais desta área. Podem-se citar como exemplos: modelos matemáticos, definições iniciais de produtos, definições de carregamento e casos de teste e definições dos processos de simulação, incluindo a sequência de atividades e passos necessários para a execução das simulações, assim como dos dados necessários de entrada e os padrões de interface entre processos (NORRIS, 2012).

É necessário rastrear quais dados de entrada deram origem a um determinado dado de saída, e quais os programas utilizados nesta conversão, o que dá origem ao *pedigree* de dados mencionado anteriormente. A determinação do *pedigree* é importante para que, caso seja necessário no futuro, o processo possa ser executado novamente, com os mesmos dados de entrada ou com dados alternativos, para a investigação de cenários de operação que não haviam sido originalmente previstos. Como os dados de simulação geralmente estão distribuídos em sistemas diferentes, o sistema de SPDM deve guardar uma ligação (*link*) com estes sistemas, para preservar a referência e ser possível reacessar o dado futuramente (NORRIS, 2012).

Uma vez que o processo de simulação tenha sido executado dentro de um sistema de SPDM, torna-se mais fácil localizar os dados de entrada e saída de uma simulação. Estes dados, muitas vezes em formato binário e contidos em arquivos de tamanho grande, só são visualizados apropriadamente nos softwares de simulação que os originaram. O problema é que, embora os dados estejam disponíveis no sistema de SPDM a todos os envolvidos, nem todos podem possuir as licenças para execução dos softwares apropriados a visualização dos dados. Neste caso, o sistema de SPDM pode ajudar ao serem criadas, no processo de simulação, atividades que convertam estes dados para um formato sem restrição de acesso e com tamanhos de arquivos menores, como formato texto ou imagem, para que possam ser visualizados por todos (NORRIS, 2012).

Outro desafio em SPDM é o armazenamento das premissas utilizadas no processo de simulação, que indicam “como” o processo foi executado, incluindo a escolha dos dados de entrada, dos casos a serem simulados e como os resultados foram interpretados. Um resumo com os principais desafios na área de SPDM pode ser visto na Tabela 2.5.

Tabela 2.5 – Desafios operacionais em SPDM.

1. Documentar as atividades de simulação de forma rigorosa e eficiente para suportar as decisões de negócio, a preservação da propriedade intelectual e o desenvolvimento tecnológico
2. Ser capaz de rastrear todos os dados de entrada usados para a obtenção de um resultado, de forma a ser capaz de reexecutar uma simulação e replicar os resultados
3. Melhorar a produtividade da engenharia ao substituir o gerenciamento manual dos dados e processos de simulação por sistemas de SPDM que automatizam esta atividade
4. Localizar informações de projeto de engenharia rapidamente para suportar o processo de tomada de decisão
5. Recuperar e armazenar dados de engenharia rapidamente
6. Visualizar informações em arquivos grandes e/ou binários

Fonte: Adaptado de Norris (2012). (tradução nossa).

2.6.3. Principais funcionalidades de sistemas SPDM

Segundo o NAFEMS (2017c), um sistema SPDM deve conter as seguintes funcionalidades mínimas, além de gerenciar informações de processo e contexto necessárias para dados de simulação:

a) **controle de acesso**: inclui funcionalidades para gerenciamento de usuários, concessão ou revogação de acesso a um determinado conjunto de dados. Isto é fundamental para a segurança da informação, garantindo que apenas as pessoas autorizadas tenham acesso a dados controlados;

b) **funções básicas de gerenciamento de dados**: inclui funcionalidades como versionamento, recuperação de versões antigas de arquivos e controle de mudança;

c) **captura automática de metadados**: permite que dados sejam identificados e classificados de forma inequívoca, facilitando a procura e o resgate destes dados futuramente;

d) **criação de relacionamento entre dados**: permite que relações de hierarquia e dependência entre dados sejam criadas, o que

facilita a navegação em estruturas de dados e o estabelecimento do *pedigree* das informações;

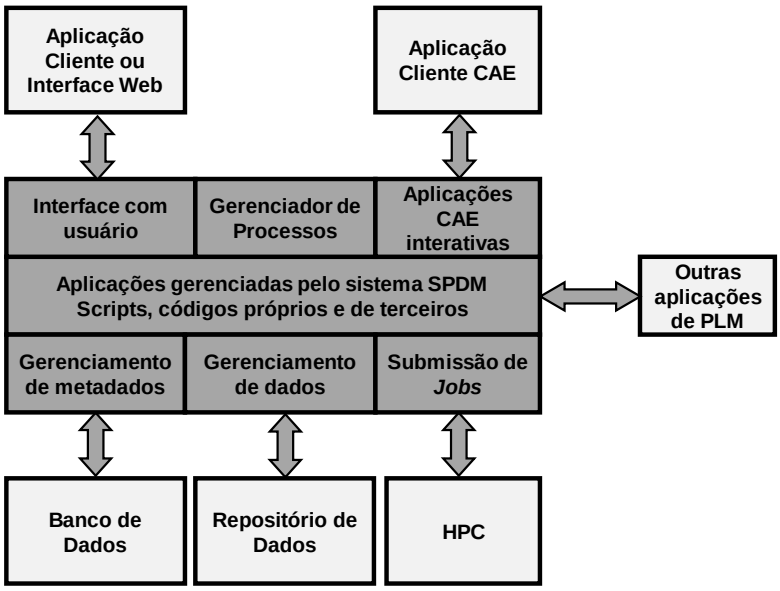
e) **criação de *workflows***: permite que um processo de simulação que envolva várias atividades seja mapeado e disponibilizado como modelo para que outras pessoas possam executá-lo.

Como funcionalidades desejáveis, pode-se citar a integração com outros sistemas de apoio, como sistemas de PDM, PLM, bancos de dados de requisitos e HPC.

2.6.4. Infraestrutura típica de sistemas SPDM

Segundo NORRIS (2012), uma plataforma típica de um sistema SPDM terá os componentes sombreados em cinza escuro na Figura 2.13. Esta plataforma geralmente está instalada em um ou mais servidores e possui conexões com os demais sistemas e aplicações representados pelos componentes em cinza claro à sua volta.

Figura 2.13 – Infraestrutura típica de um sistema SPDM.



Fonte: Adaptado de Norris (2012). (tradução nossa).

Os componentes sombreados em cinza escuro representam funcionalidades do sistema SPDM, como a de gerenciamento de dados, que mantém uma conexão com um repositório externo de dados (arquivos), usualmente chamado de *vault*, e a de gerenciamento de metadados, que mantém uma conexão com um banco de dados que contém tabelas onde são armazenados os metadados correspondentes aos arquivos armazenados no *vault*. Assim, quando um usuário do sistema SPDM publica um arquivo no sistema, ele é armazenado fisicamente no *vault*, e todas as informações sobre ele, como autor do arquivo, data de criação e dependências de outros arquivos, são armazenados em uma tabela no banco de dados. O sistema de SPDM gerencia a conexão entre dados e metadados.

Para que os arquivos armazenados no *vault* sejam protegidos de acessos indevidos, geralmente são criptografados. O sistema de SPDM cria uma estrutura interna de pastas no *vault* e renomeia os arquivos de forma que não seja possível identificá-los. A única forma de recuperar uma informação é então fazer *login* no sistema SPDM, realizar uma busca por metadados e o sistema disponibilizará o arquivo desejado para *download*, caso o usuário tenha permissão de acessá-lo.

Conforme visto na Figura 2.13, sistemas SPDM também contam com módulos para comunicação com sistemas de HPC, onde as simulações são executadas. Estes sistemas geralmente contam com funcionalidades de enfileiramento de solicitações de execução de simulações, ranqueadas por meio de critérios definidos pelo administrador do sistema. Assim, vários usuários podem submeter suas solicitações de execução de simulações ao sistema de HPC e elas serão priorizadas usando estes critérios. Alternativamente, as simulações podem ser executadas também no próprio computador do usuário.

Sistemas de SPDM contam também com funcionalidades para administrar aplicações, tanto internas, desenvolvidas pelos usuários do sistema, quanto externas, desenvolvidas por terceiros. Estas aplicações podem fazer interface

com outras aplicações de PLM, residentes fora do sistema SPDM, como aplicações de gerenciamento de requisitos.

A interface com o usuário pode ser tanto fornecida por meio de uma aplicação própria (cliente) quanto por meio de um navegador de internet, ou mesmo uma combinação das duas alternativas. Aplicações CAE interativas, como um software de edição de malha de elementos finitos, podem ser configuradas para ler dados do sistema SPDM e escrever dados modificados no próprio sistema. Isto facilita o trabalho do usuário, que não precisa fazer uma cópia manual dos arquivos do sistema SPDM para seu computador para então abri-los no software em questão, e depois não precisa realizar uma publicação manual dos dados modificados. Entretanto, estas configurações muitas vezes são dependentes do fornecedor das aplicações CAE.

Por fim, a funcionalidade de gerenciamento de processos permite que as aplicações registradas no sistema SPDM possam ser concatenadas em um *workflow* que é executado automaticamente pelo sistema, que cuida do fluxo dos dados entre as aplicações, ao mesmo tempo em que captura os metadados necessários para busca futura.

3 NORMAS APLICÁVEIS A SPDM

Neste capítulo são apresentadas as normas e recomendações da *European Cooperation for Space Standardization* (ECSS)relacionadas a M&S e gerenciamento de dados e processos de simulação. Estas normas e orientações apresentam requisitos que orientam a especificação da proposta de gerenciamento de dados e processos de simulação, apresentada no capítulo 5.

3.1. **Introdução**

Segundo a norma ECSS-P-00C (ECSS, 2013a), a padronização é uma ferramenta importante para reduzir riscos, custos e melhorar a qualidade e a comunicação entre os participantes de um programa espacial durante sua preparação e execução. No passado, as agências espaciais europeias e a indústria desenvolveram padrões próprios e os aplicaram em seus programas espaciais. A ECSS foi criada para harmonizar os requisitos de normas existentes para projetos espaciais e prover um único e coerente conjunto de normas para uso em todos os desenvolvimentos e operações de sistemas espaciais na comunidade europeia.

Segundo a norma ECSS-S-ST-00C (ECSS, 2008), o sistema de normas da ECSS é composto de documentos que abordam aspectos essenciais das três áreas vitais para o sucesso de programas e projetos espaciais: gerenciamento de projetos, engenharia e qualidade. Para estas áreas foram desenvolvidos documentos normativos (*standards*), manuais (*handbooks*) e memorandos técnicos (*technical memoranda*).

Documentos normativos contêm requisitos verificáveis que devem ser atendidos em projetos ou programas espaciais, mas não indicam o meio de cumprimento dos mesmos. Manuais são documentos não-normativos que fornecem orientação, conselhos e recomendações relacionadas a uma disciplina específica ou a técnicas, tecnologias, processos ou atividades

específicas. Memorandos técnicos são documentos não-normativos que fornecem informações úteis para a comunidade espacial sobre um assunto específico. Eles cobrem dados que não são abordados em documentos normativos e manuais, ou ainda não estão maduros o suficiente para serem publicados como normas ou manuais.

Ainda não existem documentos normativos e manuais relacionados a gerenciamento de dados e processos; existem apenas memorandos técnicos que apresentam melhores práticas e sugestões sobre o tema e podem evoluir para documentos normativos no futuro. A seguir, foram compiladas as principais recomendações da ECSS sobre gerenciamento de dados e processos de simulação.

3.2. **ECSS-E-TM-10-21A**

O memorando técnico ECSS-E-TM-10-21A, intitulado *Space Engineering - System modelling and simulation* (ECSS, 2010a) aborda os recursos de M&S necessários para suportar as atividades de análise, projeto e verificação no nível de sistema. O seu propósito é prover orientação e um conjunto de melhores práticas para os engenheiros responsáveis por projeto de sistemas sobre como utilizar M&S para suportar as atividades de *Engenharia de Sistemas*. O conteúdo integral desta seção tem como única referência a ECSS (2010a).

A tendência atual entre os desenvolvedores de sistemas espaciais é analisar com cuidado as necessidades de M&S durante todo o desenvolvimento dos sistemas para maximizar o reuso de modelos e softwares de simulação. A ideia é que um modelo precise ser desenvolvido apenas uma vez e que seja genérico e parametrizado o suficiente para que possa ser reusado em outros contextos futuramente. A intenção é que seja construído um modelo virtual ou digital do sistema completo já nas fases iniciais do projeto para suportar as atividades de definição de requisitos e definição de sistemas, evoluindo-o para suportar as fases posteriores de definição detalhada, montagem, integração,

testes e operação. O reuso de *scripts* de configuração de simulações e dados de configuração de missão é igualmente importante.

Além disso, incentiva-se o uso de um padrão comum para troca de dados entre os envolvidos no projeto, para manter a consistência dos dados e para que o esforço de engenharia seja focado em atividades relevantes e não se perca tempo em conversões desnecessárias de dados de diferentes formatos. Isto é importante especialmente em sessões de engenharia simultânea onde a interação entre múltiplas disciplinas de projeto é intensa. Os principais dados a serem harmonizados são modelos (e demais dados associados), bancos de dados de engenharia e documentações de projeto. Atualmente, existem apenas soluções parciais e *ad-hoc* para abordar este tema, embora já existam outras iniciativas da ECSS para normatizar esta questão.

Para que esta visão se concretize, é fundamental que o uso de M&S seja uma parte integral dos processos de especificação de requisitos, projeto e verificação. Esta questão foi abordada na seção 2.4.2, ilustrada pela Figura 2.7.

3.2.1. Tipos de simuladores

Nas fases iniciais de um projeto espacial, simuladores são desenvolvidos para domínios ou subsistemas específicos, como os sistemas de controle de atitude e órbita, térmico e de estruturas, ou para a análise de casos específicos de um sistema integrado como o pouso na superfície de um planeta. Assim, simuladores são usados para suportar a análise de missão e o desenvolvimento de subsistemas e sistemas. Devido ao caráter particular do desenvolvimento de simuladores para as fases iniciais de projeto, não foi definida ainda uma metodologia uniforme para suportar tais desenvolvimentos.

Existem cinco tipos de simuladores para projeto e análise tipicamente utilizados ao longo de um projeto espacial. São eles:

a) **Simulador de Conceito do Sistema (SCS):**um dos primeiros simuladores a ser utilizado, suporta necessidades específicas de disciplinas de engenharia e a avaliação rápida de conceitos de projeto de sistemas. O SCS é desenvolvido a partir de requisitos de alto nível do sistema e os resultados de suas simulações podem atualizar estes requisitos;

b) **Simulador Funcional de Engenharia (SFE):** permite a verificação de elementos críticos de uma linha de base de projeto de sistema, como os algoritmos do subsistema de controle de atitude e órbita. Recomenda-se o reuso de modelos matemáticos do SCS no SFE para preparar a base dos simuladores utilizados nas fases seguintes;

c) **Simulador de Validação Funcional (SVF):** suporta a análise do desempenho do sistema e os testes e validação dos subsistemas críticos no contexto do sistema global;

d) **Simulador de Desempenho da Missão (SDM):**permite o estabelecimento e verificação do desempenho global da missão do ponto de vista do usuário, incluindo modelos de carga paga;

e) **Simulador de Validação de Software (SVS):** permite a verificação do software embarcado em um contexto representativo do sistema espacial em seu ambiente operacional, contendo modelos representativos do hardware do sistema espacial e de seu comportamento dinâmico no espaço.

3.2.2. Componentes de uma simulação

Toda instância de simulação de um sistema espacial é composta dos seguintes componentes:

a) **modelos dedicados de simulação:** inclui todos os elementos relativos ao sistema espacial, como algoritmos, modelos lógicos e dados associados;

b) **infraestrutura do simulador:** inclui todas as funções genéricas necessárias para iniciar, controlar e executar as simulações, realizar interfaces com sistemas externos, interagir com modelos e registrar informações das simulações;

c) **configurações da simulação:** consiste de um repositório de informações disponíveis para configuração, inicialização e controle de tempo de execução de uma instância particular de uma simulação. Este componente é responsável por permitir a automação de simulações por meio de *scripts* e gerenciamento dos cenários de simulação.

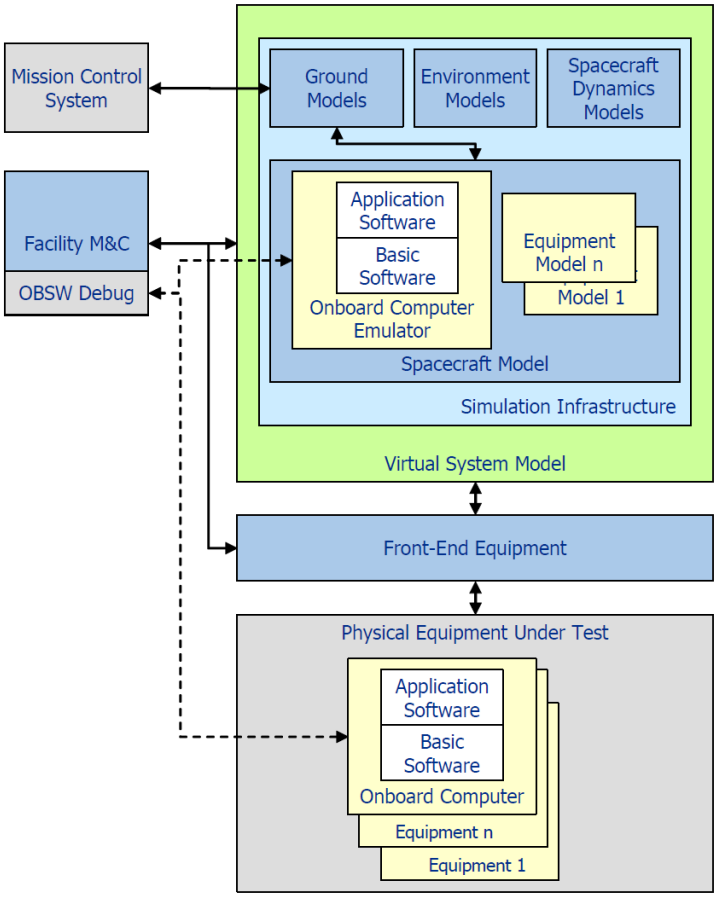
Para suportar a função de *Engenharia de Sistemas* da fase A até a fase E de um projeto espacial, um conjunto estruturado de recursos de simulações e testes é proposto pela ECSS (2010a) e mostrado na Figura 3.1. O núcleo deste conjunto de recursos é o Modelo Virtual do Sistema (*Virtual System Model*). Este modelo reflete as funções e o comportamento do sistema completo a ser construído, no nível necessário para ser capaz de suportar as tarefas de análise e verificação. O Modelo Virtual do Sistema é composto da Infraestrutura de Simulação e dos modelos do sistema espacial. Interfaces externas dos equipamentos a serem testados e recursos externos de monitoramento e controle são suportados.

3.3. **ECSS-E-TM-40-07**

O documento normativo ECSS-E-ST-40, intitulado *Space software engineering* (ECSS, 2009a), contém requisitos genéricos relacionados a *software* no contexto de sistemas espaciais. O memorando técnico ECSS-E-TM-40-07, intitulado *Space Engineering – Simulation Modelling Platform* complementa a

norma ECSS-E-ST-40, trazendo requisitos adicionais específicos para software de simulação. Este memorando inclui a interpretação dos requisitos da norma ECSS-E-ST-40 para software de simulação, além de práticas especiais aplicáveis a este tipo de software.

Figura 3.1 – Componentes da arquitetura dos recursos de simulações e testes.



Fonte: ECSS (2010a).

O memorando ECSS-E-TM-40-07 segue a estrutura da ECSS-E-ST-40 e é composto de cinco volumes:

a) **Volume 1A – *Principles and requirements***: descreve a *Simulation Modelling Platform* (SMP) e os princípios especiais aplicáveis a softwares de simulação (ECSS, 2011a);

b) **Volume 2A – *Metamodel***: descreve a *Simulation Model Definition Language* (SMDL), que fornece mecanismos independentes de plataforma computacional para construção e integração de modelos (ECSS, 2011b);

c) **Volume 3A - *Component Model***: fornece definições independentes de plataforma computacional dos componentes de uma simulação SMP, incluindo modelos, serviços e o simulador (ECSS, 2011c);

d) **Volume 4A - C++ *Mapping***: fornece um mapeamento dos modelos independentes de plataforma computacional para a plataforma ANSI/ISO C++ (ECSS, 2011d);

e) **Volume 5 – SMP *usage***: fornece uma descrição, sob o ponto de vista do usuário de simulação, dos conceitos gerais mostrados nos volumes anteriores, assim como instruções para a realização das principais tarefas envolvidas no desenvolvimento de modelos e simuladores usando SMP (ECSS, 2011e).

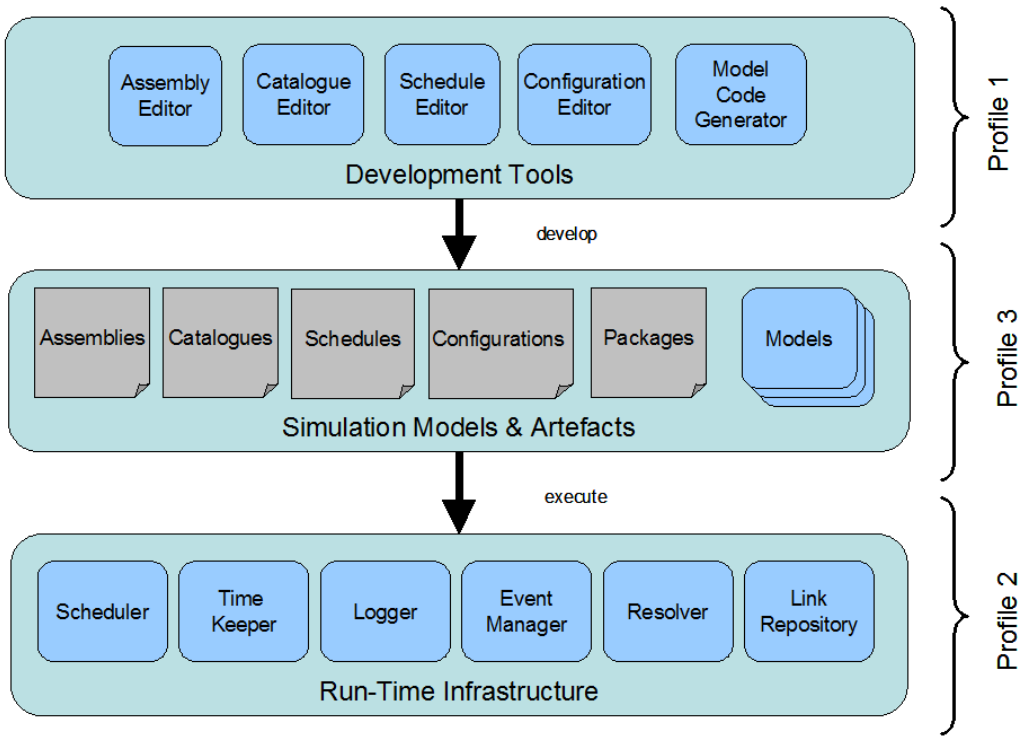
Neste memorando é definido o padrão *Simulation Model Portability*, que em sua segunda versão é conhecido pela sigla SMP2. Este padrão foi criado com o objetivo de promover o reuso de modelos de simulação tanto dentro de um mesmo projeto espacial quanto entre vários projetos diferentes. Assim, é minimizado o custo de desenvolvimento associado a modelos de simulação.

O padrão SMP2 suporta tanto o reuso de modelos em diferentes ambientes de execução (*runtime infrastructures*), pela definição de interfaces padrão, quanto a integração entre diferentes modelos de componentes de um sistema para a simulação de um sistema completo.

O memorando ainda classifica os produtos necessários para o desenvolvimento de um sistema de simulação em três grupos, chamados de *Conformance Profiles*: Ferramentas de Desenvolvimento, Modelos e Artefatos

de Simulação e Infraestrutura de Execução. Estes grupos são mostrados na Figura 3.2.

Figura 3.2 - SMP2 *Conformance Profiles*.

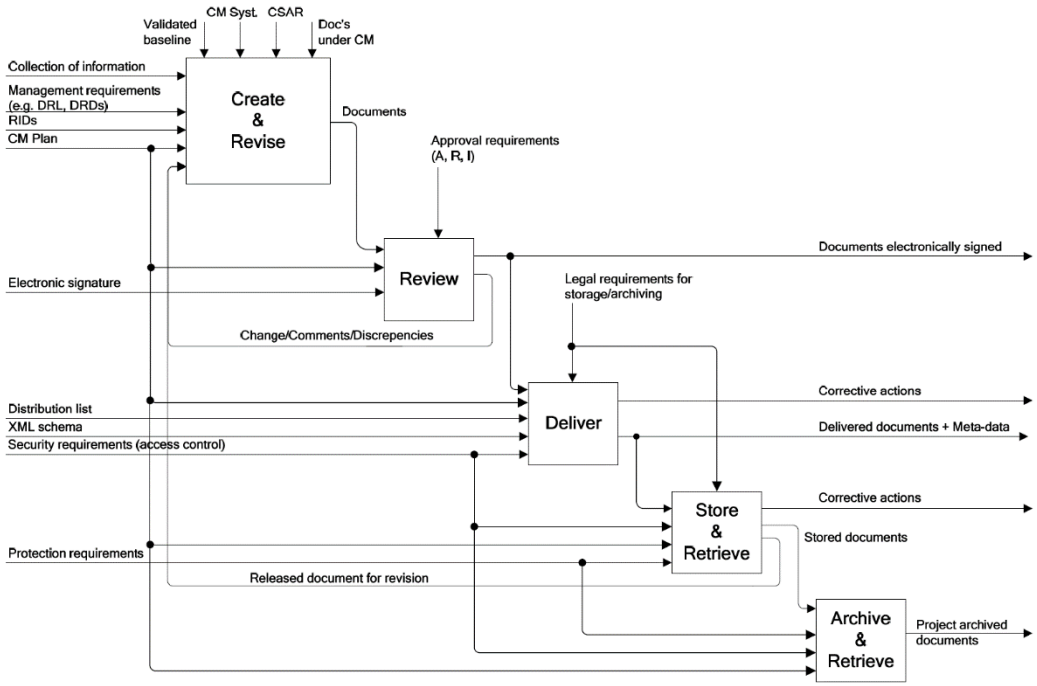


Fonte: ECSS (2011a).

3.4. **ECSS-M-ST-40**

O documento normativo ECSS-M-ST-40C, intitulado *Space project management – configuration and information management* (ECSS, 2009b), define os requisitos para gerenciamento de informações e documentações e para a configuração do produto em projetos espaciais. De acordo com a norma, o gerenciamento de informações e documentações é o processo que assegura, efetivamente e no tempo correto, a criação, coleção, revisão, entrega, armazenamento e arquivamento de informações de um projeto. Para atingir este objetivo, toda a informação registrada no projeto é gerenciada eletronicamente. A Figura 3.3 mostra o processo de implementação do gerenciamento de informações e documentações proposto pela norma.

Figura 3.3 - Implementação do gerenciamento de informações e documentações.



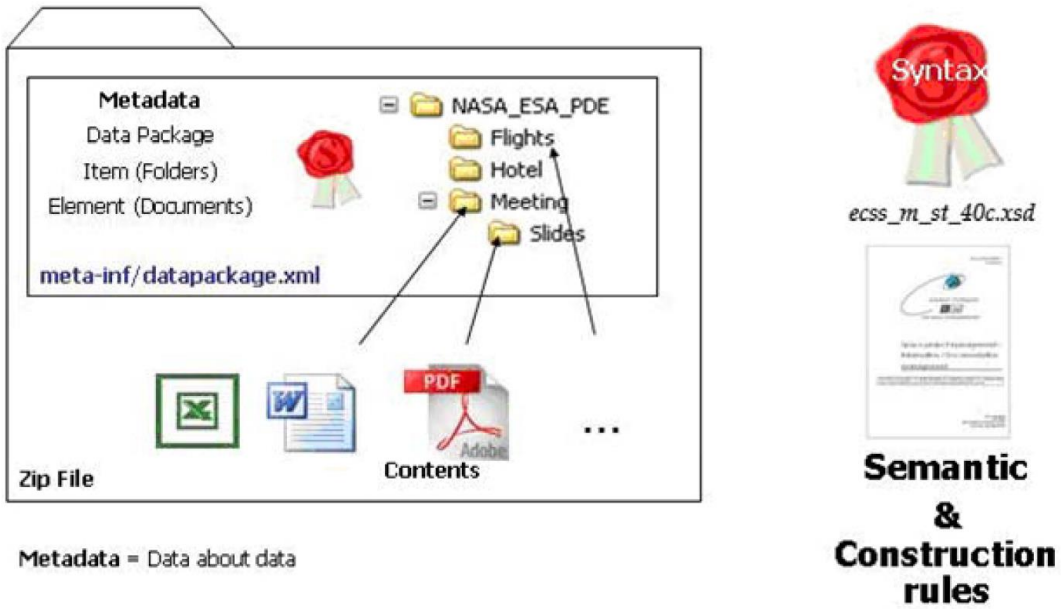
Fonte: ECSS (2009b).

Segundo o documento normativo ECSS-M-ST-40C (ECSS, 2009b), o gerenciamento de informações e documentações é aplicado durante toda a vida útil do projeto e garante que as informações fornecidas a todos os atores internos e externos ao projeto sejam corretas, acessíveis, de rápida disponibilidade, confiáveis e seguras. Além disso, assegura a coerência entre os dados armazenados e facilita a emissão de relatórios.

Um dos destaques desta norma é a especificação de um pacote de dados técnicos, ou TDP (*Technical Data Package*) usado para a distribuição de informações. O TDP, ilustrado na Figura 3.4, é um arquivo do tipo ZIP que contém dados e metadados organizados em uma estrutura padrão de pastas. Os metadados utilizados no TDP são um subconjunto do padrão ISO 10303 STEP AP232, complementados por metadados específicos, documentados no anexo L da norma ECSS-M-ST-40 (ECSS, 2009b). Para a definição destes metadados, foi elaborada uma definição de esquema XML que descreve a estrutura e os metadados do arquivo XML. Esta definição, contida em um

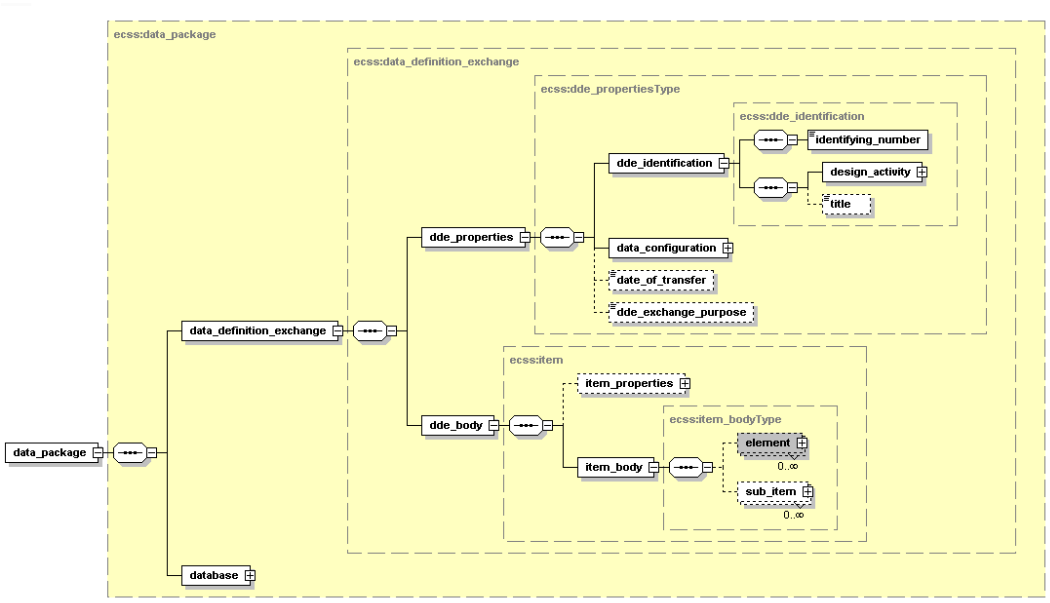
arquivo XSD (XML *Schema Definition*), está disponível para *download* na página oficial da ECSS e sua estrutura é mostrada na Figura 3.5**.**

Figura 3.4 - Technical Data Package (TDP).



Fonte: ECSS (2009b).

Figura 3.5 - Esquema XML da norma ECSS-M-ST-40C.



Fonte: ECSS (2009b).

3.5. **ECSS-E-TM-10-23A**

O memorando técnico ECSS-E-TM-10-23A, intitulado *Space Engineering – Space System Data Repository* (ECSS, 2011f), tem como objetivo estabelecer uma especificação formal da semântica dos dados de engenharia para permitir a criação de repositórios de dados de engenharia com interfaces padronizadas de troca de dados, o que é chamado de “repositório de dados do sistema espacial”. A ECSS pretende transformar este memorando técnico em um documento normativo em um futuro próximo, mas ainda não é possível estabelecer normas que tenham a maturidade e validação industrial requeridas para aplicação em projetos espaciais atuais.

Atualmente existem muitas famílias diferentes de sistemas espaciais, cada uma com diferentes requisitos relacionados aos dados a serem compartilhados, além de muitas tecnologias diferentes de sistemas de informação implantadas e muitos usuários diferenciados destes sistemas. O memorando deixa claro que, devido a este contexto, é impossível e inviável o estabelecimento de um repositório de dados central e único em um projeto espacial. O Repositório de Dados de Sistemas Espaciais proposto é um conjunto de diferentes bancos de dados logicamente integrados em uma arquitetura interoperável de forma que os dados possam ser compartilhados de forma efetiva e confiável. Espera-se, com isso, um aumento do reuso de dados de desenvolvimentos anteriores e melhoria da qualidade, consistência e rastreabilidade dos dados. O conteúdo integral desta seção tem como única referência a ECSS (2011f).

3.5.1. Motivação

O compartilhamento confiável de dados durante o ciclo de vida de um sistema espacial é essencial para melhorar tanto a efetividade quanto a eficiência dos processos de engenharia. Infelizmente, na prática, é um grande desafio conseguir os dados corretos, no momento correto e do interessado correto. Encontrar as versões corretas de dados e o histórico de mudanças em versões anteriores pode ser muito ineficiente e frustrante. Disponibilizar dados em um

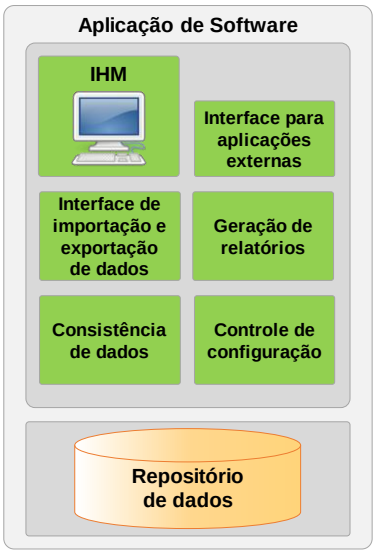
formato compatível com o compartilhamento com outros participantes do projeto pode ser difícil, senão impossível.

Para o desenvolvimento de sistemas complexos, cada vez mais estão sendo utilizados processos de engenharia simultânea que integram times responsáveis por diferentes disciplinas de projeto. Estes processos estão cada vez mais suportados e dependentes de sistemas computacionais que permitam a troca confiável de dados e os processos de revisão de projeto, verificação e validação.

3.5.2. Aplicações de software no suporte ao ciclo de vida de sistemas espaciais

Todos os processos no ciclo de vida de sistemas espaciais usam e produzem dados. Estes processos são suportados por diversas aplicações de software, que possuem repositórios de dados com diversas funções, como armazenamento de dados, interfaces de importação e exportação de dados, interfaces homem-máquina, geração de relatórios, verificação de consistência de dados e controle de configuração e versão dos dados. Estes componentes estão representados na Figura 3.6.

Figura 3.6 – Aplicação de software típica e seus componentes de dados.

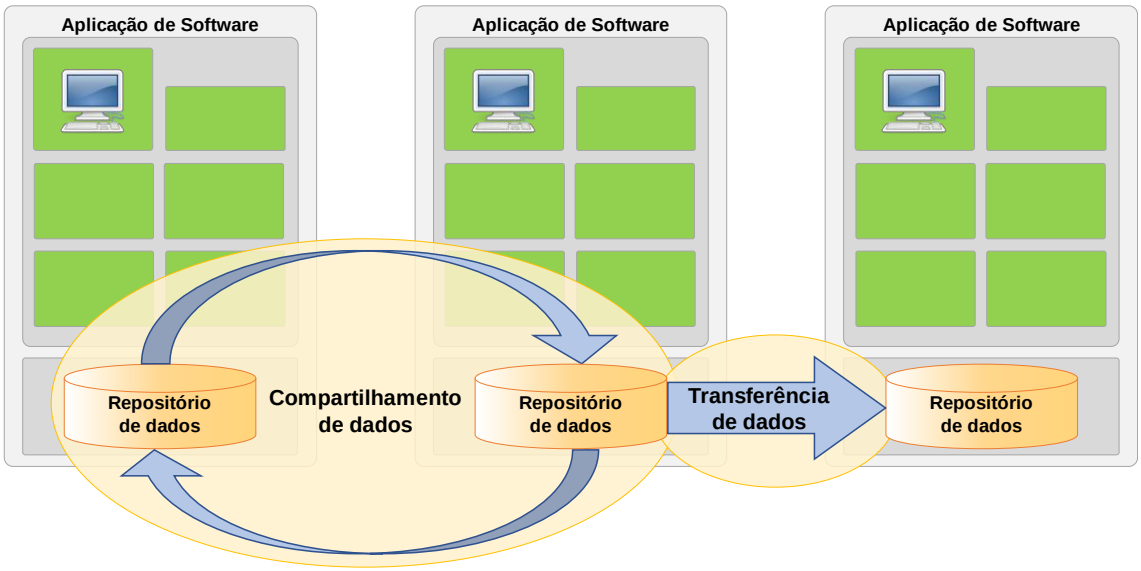


Fonte: Adaptado de ECSS (2011f). (tradução nossa).

Ao longo do ciclo de vida dos sistemas espaciais, dados são compartilhados entre diferentes usuários e para diversos propósitos, como, por exemplo, a troca de dados interdisciplinar entre a equipe de análise térmica e a equipe de análise estrutural em uma sessão de engenharia simultânea. Como mostrado na Figura 3.7, a troca de dados entre aplicações de software pode se dar de duas formas: um compartilhamento de dados dinâmico entre duas ou mais aplicações ou uma transferência de dados única, feita mediante a exportação de dados de um repositório e sua importação em outro.

Dado que existem diversas aplicações de software já desenvolvidas usando diferentes ferramentas de banco de dados, haveria um grande impacto no tempo e custo de um projeto espacial se novas aplicações que compartilhassem um mesmo padrão de ferramenta de banco de dados tivessem que ser desenvolvidas. Assim, a recomendação da ECSS é que a padronização dos dados a serem compartilhados se dê no nível semântico, por meio de um modelo de dados conceitual, independentemente de qualquer tecnologia de implantação de software, sendo inapropriado definir, por exemplo, formatos padrão de troca de dados, APIs ou protocolos como XML/XSD, ODBC ou serviços Web.

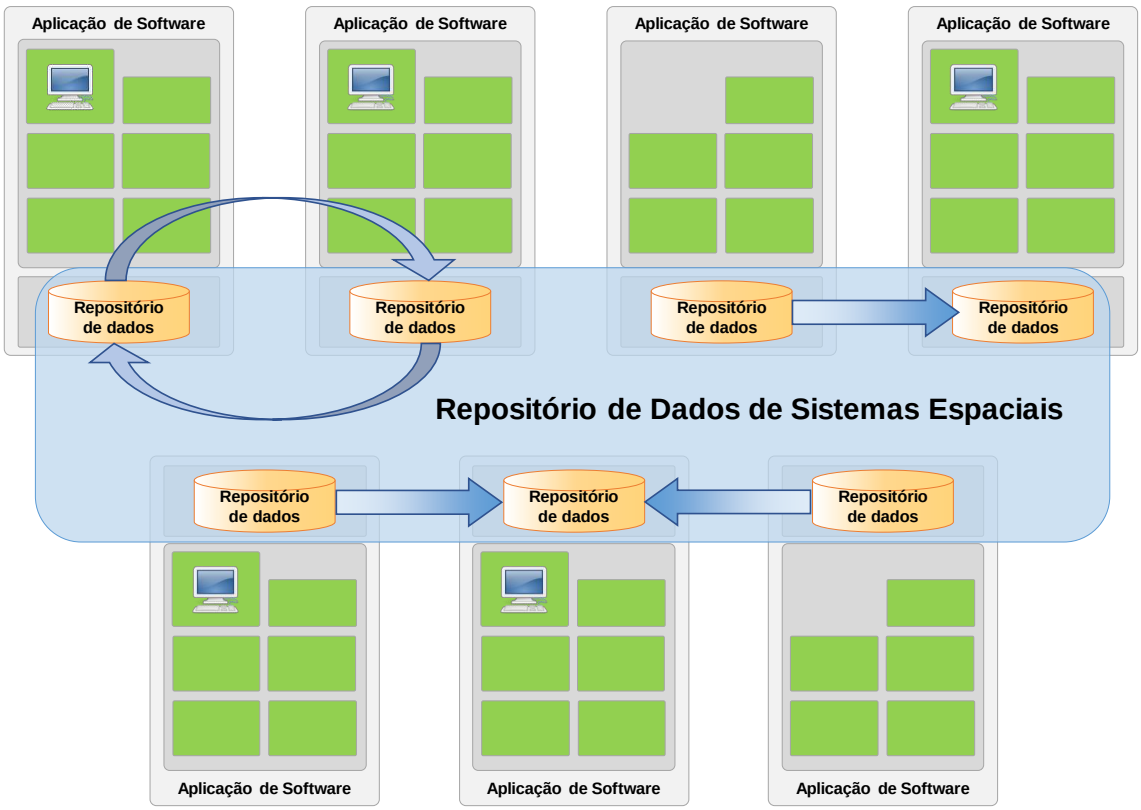
Figura 3.7 – Cenários de troca de dados entre aplicações de software.



Fonte: Adaptado de ECSS (2011f). (tradução nossa).

Se uma interoperabilidade semântica é alcançada entre diferentes aplicações de software, então o conceito de um Repositório de Dados de Sistemas Espaciais, mostrado na Figura 3.8, pode ser atingido. Tal repositório não é um banco de dados único e centralizado, e sim um conjunto de bancos de dados, dispersos tanto fisicamente quanto temporalmente, que são logicamente integrados de tal forma que dados podem ser trocados efetivamente e confiavelmente entre eles. Em alguns cenários, não é necessária a transferência completa do banco de dados de uma aplicação para outra, e sim apenas um subconjunto de dados. Entretanto, deve ser possível estabelecer referências externas (*links*) entre os objetos pertencentes a repositórios de dados de diferentes aplicações de software dentro do mesmo Repositório de Dados de Sistemas Espaciais.

Figura 3.8 – Repositório de dados de sistemas espaciais.



Fonte: Adaptado de ECSS (2011f). (tradução nossa).

É importante que se realize uma gestão da configuração dos dados que residem nos repositórios de dados das diversas aplicações de software do Repositório de Dados de Sistemas Espaciais, para fins do estabelecimento de linhas de base, controle de configuração e controle de versão dos dados. É recomendado que uma mesma abordagem de gerenciamento de configuração seja aplicada a todos as aplicações do Repositório de Dados de Sistemas Espaciais.

Para se compreender como a interoperabilidade semântica de dados pode ser atingida, é necessário analisar o processo de desenvolvimento das aplicações de software. Uma das etapas deste processo é a criação de **modelos de dados** por meio da aplicação de regras formais de modelamento. Estes modelos de dados seguem uma hierarquia composta de três tipos de modelos (conhecida como “esquema de dados”): modelo de dados conceitual, modelo de dados lógico e modelo de dados físico. O **modelo de dados conceitual** especifica a semântica dos dados relativos a um domínio, incluindo dados significativos para usuários finais, suas características e associações entre eles. O **modelo de dados lógico** é desenvolvido a partir do modelo de dados conceitual e é uma expressão do mesmo por meio de uma metodologia particular de modelamento de dados, como bancos de dados relacionais ou orientação a objetos. O **modelo de dados físico** é derivado do modelo de dados lógico e é o modelo implementado em uma linguagem específica (ex.: SQL) que será incorporada na aplicação de software final.

Para se atingir a interoperabilidade semântica entre as aplicações de software do Repositório de Dados de Sistemas Espaciais, é necessário que os modelos de dados conceituais utilizados em seus desenvolvimentos sejam subconjuntos de um **modelo de dados conceitual global** cujo escopo engloba todo o ciclo de vida do sistema espacial. Este modelo de dados conceitual global contém todos os dados significativos de todas as disciplinas de sistemas. Alternativamente, no caso de aplicações pré-existentes que não utilizaram o modelo de dados conceitual global em seu desenvolvimento, deve-se analisar

a possibilidade de estendê-las com a criação de adaptadores de interface compatíveis com o modelo de dados conceitual global.

O estabelecimento de um Repositório de Dados de Sistemas Espaciais para um dado projeto espacial envolve os seguintes passos:

a) atribuir o papel de *autoridade de projeto* para o Repositório de Dados de Sistemas Espaciais;

b) definir o escopo do Repositório de Dados de Sistemas Espaciais para o projeto, definindo o que será envolvido, por exemplo: quais disciplinas de engenharia, quais fases do ciclo de vida, quais partes da cadeia interessado-fornecedor, etc;

c) produzir o modelo de dados conceitual global do projeto por meio da adaptação do modelo de dados conceitual global desenvolvido pela ECSS;

d) definir a arquitetura do Repositório de Dados de Sistemas Espaciais, definindo, por exemplo: quais aplicações e repositórios de dados, quais interfaces de troca de dados entre repositórios, qual a abordagem para controle de configuração de dados, etc;

e) derivar o modelo de dados conceitual “local” de cada aplicação a partir do modelo de dados conceitual global do projeto;

f) implementar e manter a infraestrutura do Repositório de Dados de Sistemas Espaciais.

Os passos c), d) e e) deste processo são mostrados na Figura 3.9.

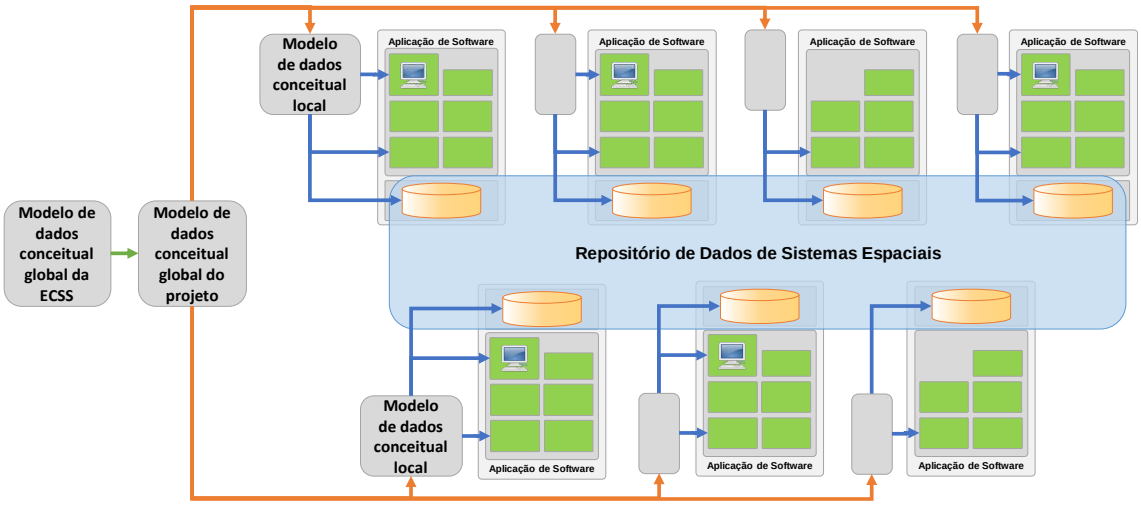
3.5.3. Modelo de dados conceitual global de sistemas espaciais

Modelos conceituais podem ser vistos como representações do mundo real em termos de tipos de objetos, suas características e relacionamentos entre eles.

Nestes modelos, o conhecimento do mundo real é expressado em termos de fatos elementares, restrições e regras de derivação.

O modelo de dados conceitual global de um sistema espacial é composto de vários elementos de sistema organizados hierarquicamente. O elemento de sistema que ocupa o topo desta hierarquia é o próprio sistema espacial e os demais elementos subjacentes representam a decomposição do sistema espacial em suas partes principais. A Figura 3.10 mostra uma típica árvore de produto de um sistema espacial com a hierarquia de seus elementos de sistema.

Figura 3.9 – Adoção do modelo de dados conceitual global da ECSS para um projeto.

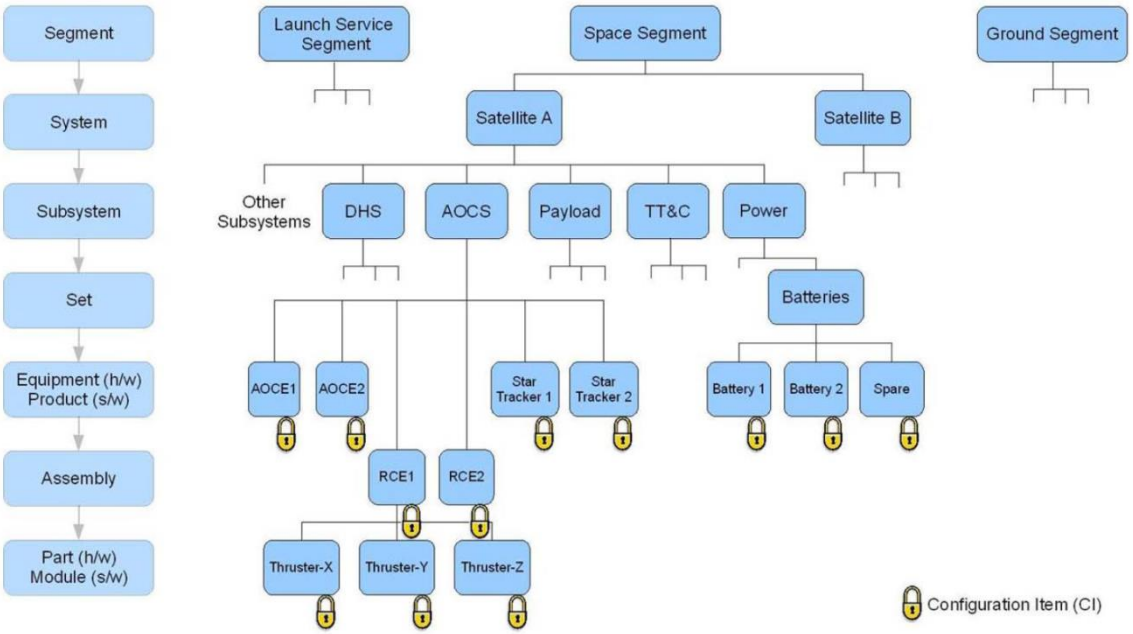


Fonte: Adaptado de ECSS (2011f). (tradução nossa).

Um elemento de sistema, além da sua decomposição física em seus subelementos constitutivos, pode conter outros tipos de objetos como atividades, dados de relatórios, funções, requisitos, etc. Estes tipos de objetos podem ser constituídos, por sua vez, de entidades, valores e associações entre eles. A Figura 3.11 ilustra este conceito, mostrando que o elemento de sistema é constituído, entre outros tipos de objetos, do tipo Atividade, que por sua vez é composto de Argumentos e valores associados, como nome, descrição, tipo, valor padrão, etc.

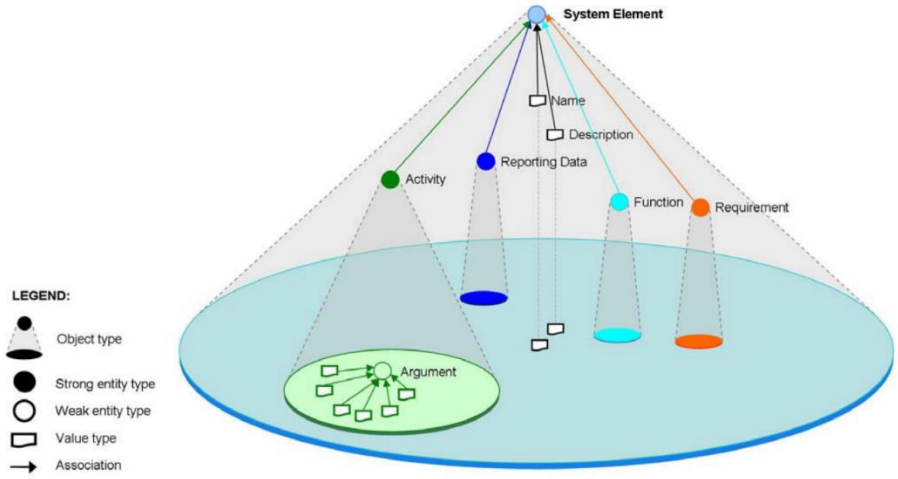
É importante que, em um dado repositório de dados, cada objeto tenha uma identificação única, conhecida como UUID (*Universally Unique IDentifier*), para garantir que os dados sejam corretamente localizados posteriormente usando ferramentas de busca. No contexto do Repositório de Dados de Sistemas Espaciais, constituído de vários repositórios de dados, é importante também que os objetos sejam identificados pela junção de um “nome relativo” (identificação única no contexto do repositório de dados local) com um “caminho de referência” (contendo a identificação única do repositório e o “caminho” desde o elemento de sistema de topo até o elemento que se quer identificar).

Figura 3.10 – Árvore de produto típica de um sistema espacial.



Fonte: ECSS (2011f).

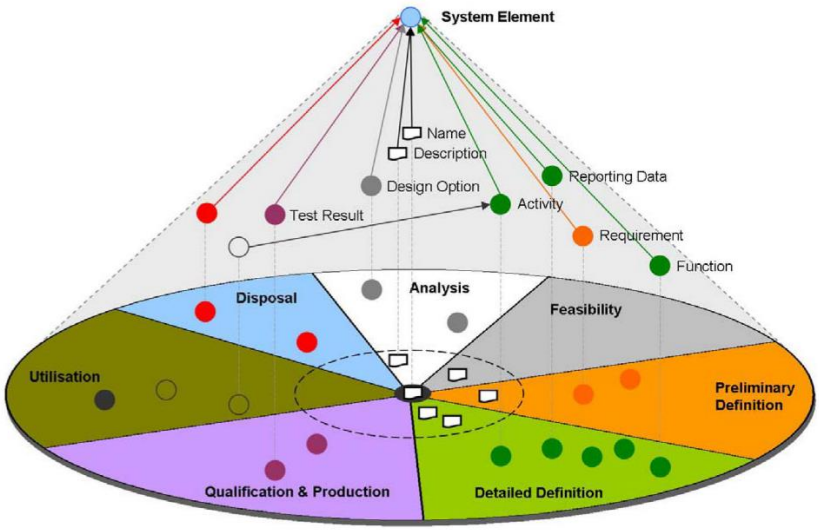
Figura 3.11 – Exemplo de tipos de objetos e entidades de um elemento de sistema.



Fonte: ECSS (2011f).

O modelo de dados conceitual global precisa também estar preparado para receber dados de todas as fases do projeto espacial que foram especificadas no escopo de um Repositório de Dados de Sistemas Espaciais. A Figura 3.12 mostra um exemplo de população de dados de um elemento de sistema ao longo das fases de Análise, Viabilidade, Definição Preliminar, Definição Detalhada, Qualificação & Produção, Utilização e Descarte.

Figura 3.12 – Dados de um elemento de sistema ao longo de seu ciclo de vida.



Fonte: ECSS (2011f).

Esta organização dos dados é importante para auxiliar as equipes de projeto durante as revisões de projeto que ocorrem geralmente na transição de uma fase para outra. Dados podem ser úteis tanto para uma revisão de projeto, embasando a evolução da maturidade do projeto para seguir para a próxima fase, quanto para ser usado como referência em fases subsequentes.

3.6. **ECSS-E-TM-10-25A**

O memorando técnico ECSS-E-TM-10-25A, intitulado *Space Engineering - Engineering design model data exchange (CDF)* (ECSS, 2010b), contém recomendações para a troca de dados de modelos, análises e seus resultados entre organizações envolvidas em um projeto espacial durante suas fases 0 e A. Isto é alcançado por meio da definição de um modelo de informação, intitulado *System Engineering Information Model* (SEIM),em que o sistema é decomposto até o nível de equipamentos, incluindo parâmetros e disciplinas associadas. Os objetivos deste memorando são: a criação de *Concurrent Design Facilities* (CDF) utilizando um modelo de informação comum e permitir a transferência de dados, colaboração em tempo real e atividades conjuntas entre organizações envolvidas em um projeto espacial. O conteúdo integral desta seção tem como única referência a ECSS (2010b).

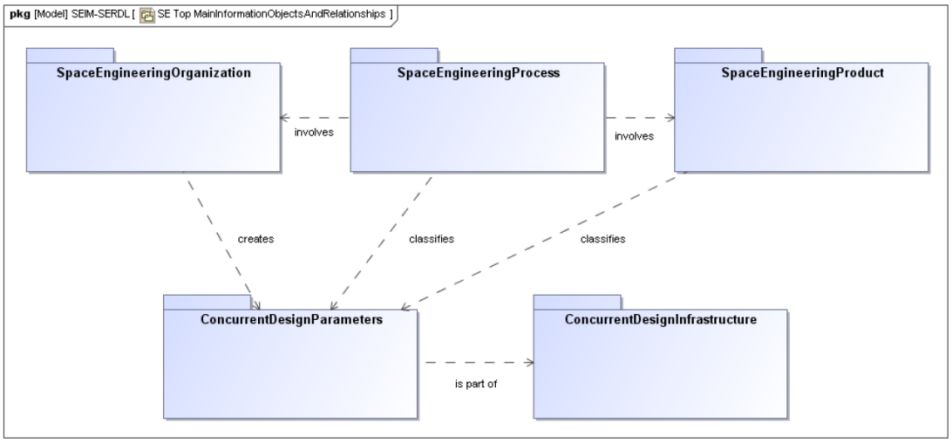
3.6.1. System Engineering Information Model (SEIM)

O SEIM descreve os principais processos e estrutura de dados necessários em estudos de engenharia simultânea. Isto inclui, por exemplo, nomes dos elementos e subelementos do sistema, propriedades de materiais, parâmetros iniciais de projeto, parâmetros de resultados de análises e simulações e nomes de disciplinas.

A Figura 3.13 mostra a vista de topo do SEIM com os cinco pacotes que agrupam os conceitos do modelo e as principais relações entre eles (todas as figuras desta seção seguem a notação UML2). Um Processo de Engenharia Espacial (*SpaceEngineeringProcess*) envolve uma Organização de Engenharia

Espacial (*SpaceEngineeringOrganization*) para conduzir o processo e um Produto de Engenharia Espacial (*SpaceEngineeringProduct*) para ser desenvolvido. As várias disciplinas desta Organização criam um conjunto de Parâmetros de Projeto Concorrente (*ConcurrentDesignParameters*), incluindo valores e metadados, classificados em termos dos passos do processo e/ou produto que eles descrevem. Estes parâmetros são armazenados e gerenciados em uma Infraestrutura de Projeto Simultâneo (*ConcurrentDesignInfrastructure*).

Figura 3.13 – Vista de topo do SEIM.



Fonte: ECSS (2010b).

Cada um dos pacotes representados na Figura 3.13 contém vários tipos de objeto necessários para capturar a informação gerada em sessões de engenharia simultânea. A Figura 3.14 mostra estes tipos de objeto em cada pacote do SEIM. Estes tipos de objetos estão agrupados nas seguintes classes, denotadas por estereótipos UML:

a) organisationClass;

b) processClass;

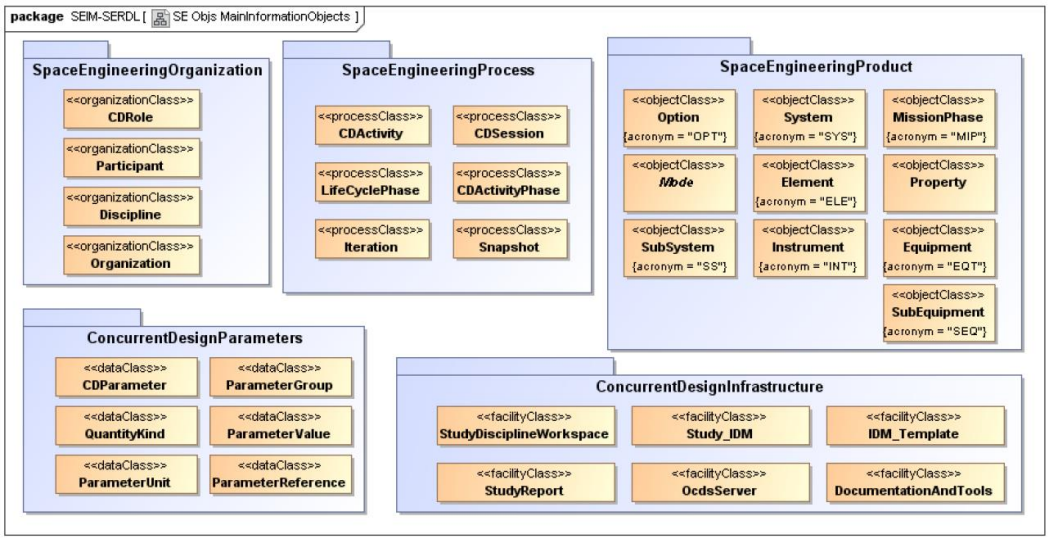
c) objectClass;

d) dataClass;

e) facilityClass.

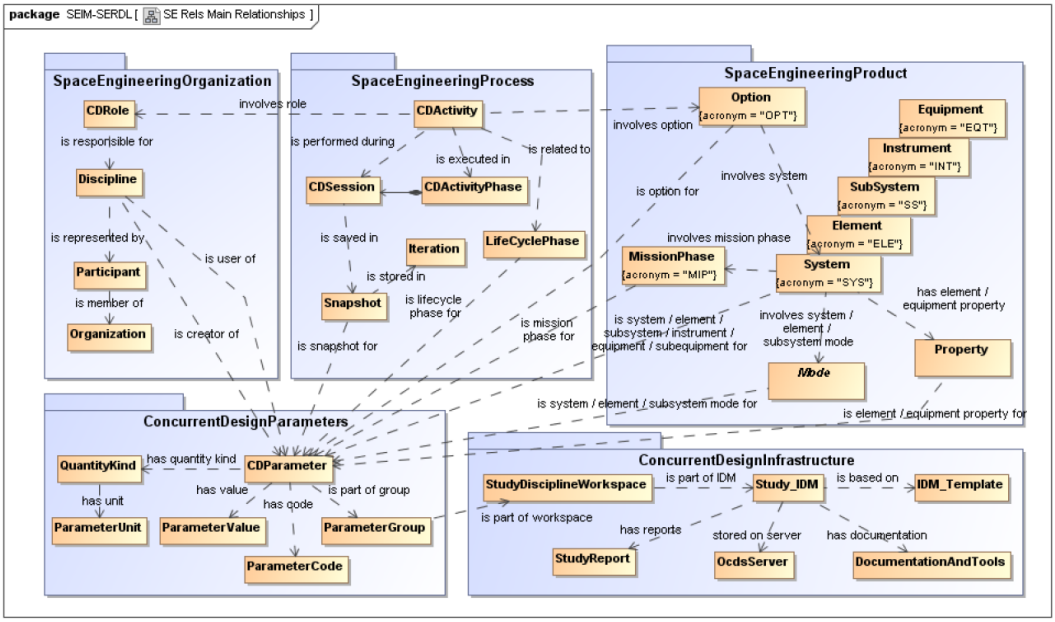
A Figura 3.15 apresenta uma informação de mais alto nível do SEIM ao mostrar os principais relacionamentos existentes entre os objetos dos pacotes mostrados na Figura 3.14.

Figura 3.14 - Tipos de objetos definidos no SEIM.



Fonte: ECSS (2010b).

Figura 3.15 - Tipos de objetos definidos no SEIM e seus relacionamentos.



Fonte: ECSS (2010b).

Para permitir a divisão do trabalho de desenvolvimento de um sistema espacial entre diversas equipes e para ajudar a lidar com a complexidade do sistema, o mesmo precisa ser decomposto hierarquicamente em subelementos. A Figura 3.16 mostra a forma proposta pelo memorando ECSS-E-TM-10-25A de decomposição do sistema, aplicável às fases 0 e A.

Nesta decomposição, as seguintes regras são válidas:

a) um Sistema (*System*) representa o sistema de nível mais elevado e pode ser decomposto em sistemas-filhos por meio do relacionamento hasSystem;

b) uma Opção (*Option*) representa uma certa opção de projeto considerada em um estudo. Um sistema pode ter mais de uma Opção;

c) um Sistema pode conter um número de Elementos (*Elements*);

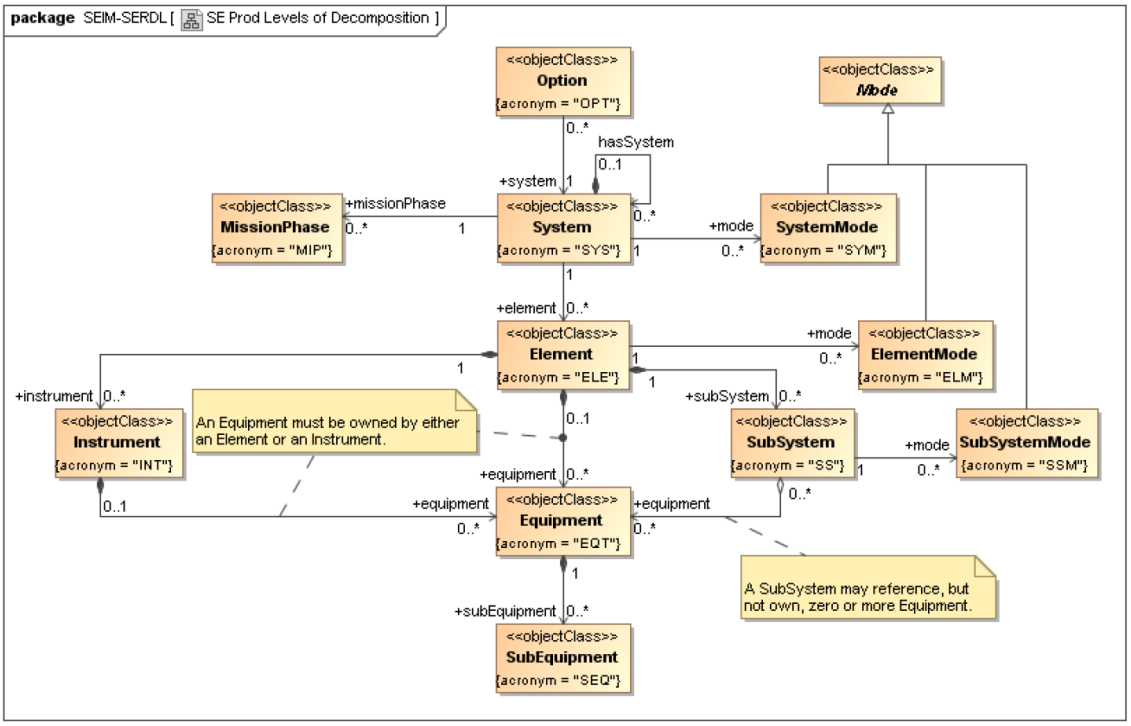
d) elementos podem conter um número de Equipamentos (*Equipments*). Alternativamente, um Elemento pode ser decomposto em Instrumentos (*Instruments*) que também podem ser decompostos em Equipamentos. Entretanto, um Equipamento pode pertencer apenas a um Instrumento ou Elemento, não ambos;

e) elementos podem conter também um número de SubSistemas (*SubSystems*), que por sua vez podem referenciar um número de Equipamentos. Note que a relação entre SubSistemas e Equipamentos é de agregação e não de composição, o que significa que um SubSistema é um agrupamento lógico de Equipamentos e não uma decomposição física;

f) equipamentos podem conter um número de SubEquipamentos (*SubEquipment*);

g) modos de operação podem estar associados ao sistema nos níveis de Sistema (*SystemMode*), Elemento (*ElementMode*) ou SubSistema (*SubSystemMode*).

Figura 3.16 - Decomposição do sistema e modos associados no SEIM.



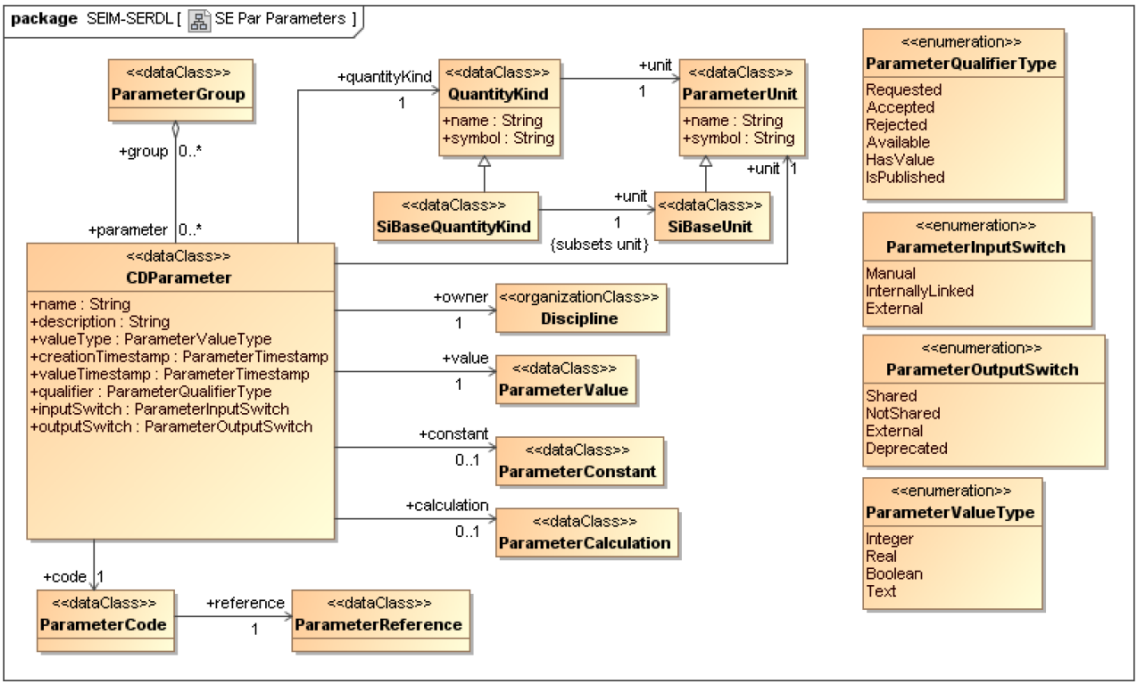
Fonte: ECSS (2010b).

O SEIM contém um conjunto de Parâmetros de Projeto Concorrente (*ConcurrentDesignParameters*), usados para capturar o projeto conceitual de sistemas de forma concisa e flexível. Estes Parâmetros de Projeto e seus valores são a base para a captura do estado de um projeto de engenharia simultânea e compartilhá-lo entre todas as disciplinas participantes. A Figura 3.17 mostra o diagrama UML com a representação destes parâmetros, que seguem as seguintes definições principais:

a) um ParâmetroPS (*CDParameter*) é o objeto central que representa um parâmetro e possui atributos e relacionamentos com os demais tipos de objeto. Este parâmetro possui atributos de nome (*name*) e descrição (*description*);

b) um GrupoDeParâmetros (*ParameterGroup*) é utilizado para agrupar parâmetros, por exemplo um grupo de parâmetros de um mesmo subsistema;

Figura 3.17 - Representação dos parâmetros de projeto simultâneo no SEIM.



Fonte: ECSS (2010b).

c) um ParâmetroPS contém um TipoDeQuantidade (*QuantityKind*) que especifica o tipo da quantidade que o parâmetro representa, como comprimento, massa, tempo, etc, e também o tipo adimensional (*dimensionless*);

d) um ParâmetroPS contém uma Unidade (*Unit*) que descreve a unidade de medida em que o parâmetro é descrito. Parâmetros adimensionais têm a unidade “1”;

e) um ParâmetroPS contém um TipoDeValorDeParâmetro (*ParameterValueType*) que descreve o tipo de dado armazenado no banco de dados, como Real, Inteiro, Booleano e Texto;

f) um ParâmetroPS contém um CódigoDeParâmetro (*ParameterCode*) que é um nome curto para o parâmetro, que segue uma convenção de nomenclatura mostrada logo em seguida nesta seção;

g) um ParâmetroPS contém uma ConstanteDeParâmetro (*ParameterConstant*) ou um CálculoDeParâmetro (*ParameterCalculation*), mas não ambos. Uma ConstanteDeParâmetro define uma constante associada ao parâmetro e um CálculoDeParâmetro define uma expressão matemática que define o valor do parâmetro em função dos valores de outros parâmetros;

h) um ParâmetroPS contém umProprietário (*Owner*) que é a Disciplina responsável pela administração do parâmetro.

A convenção de nomenclatura de umCódigoDeParâmetro sugerida pela ECSS (2010b) consiste da concatenação de códigos dos componentes do sistema de acordo com sua relação hierárquica de decomposição. A lista a seguir mostra esta relação, incluindo, em cada linha, o nome em português, nome em inglês e o código do componente:

a) Opção / *Option*/ OPT

b) Sistema / *System* / SYS

c) Fase da Missão / *MissionPhase* / MIP

d) Modo do Sistema / *SystemMode* / SYM

e) Elemento / *Element* / ELE

f) Modo do Elemento / *ElementMode* / ELM

g) Propriedade do Elemento / *ElementProperty* / ELP

h) Subsistema / *Subsystem* / ss

i) Modo do Subsistema / *SubsystemMode* / SSM

j) Instrumento / *Instrument* / INT

k) Equipamento / *Equipment* / EQT

l) Propriedade do Equipamento / *EquipmentProperty* / EQP

m) Subequipamento / *SubEquipment* / SEQ

A regra consiste na concatenação dos códigos do componente com números (representados por # no exemplo a seguir) e sinal de *underscore* (_), mostrando sua relação hierárquica, seguidos do nome da ReferênciaDoParâmetro (*ParameterReference*). Por exemplo, “OPT#_SYS#_MIP#_SYM#_ELE#_PRname” representa um modelo de código válido até o nível de Elemento. A massa de um hipotético Elemento 1 em uma certa hierarquia poderia ser representado como: “OPT3_SYS2_MIP1_SYM1_ELE1_mass”.

4 AMBIENTES DE GERENCIAMENTO DE DADOS E PROCESSOS DE SIMULAÇÃO

Neste capítulo é apresentado o conceito de “ambiente” de gerenciamento de dados e processos de simulação. É realizada também uma introdução ao software *framework* RCE, selecionado como base do ambiente proposto, e também ao VirSat, que é uma extensão deste software *framework*. São mostrados os requisitos que deram origem ao seu desenvolvimento, assim como suas principais características, funcionalidades e formas de configuração. Por fim, são mostrados os requisitos do ambiente de dados e processos de simulação proposto para ser utilizado no INPE.

4.1. **DEFINIÇÕES**

Conforme mencionado na seção 2.4.1, um “ambiente de simulação” (*simulation environment*), de acordo com a ECSS (2010a), é uma infraestrutura de software que é usada para executar os modelos de simulação. Tais ambientes geralmente têm um agendador (*scheduler*) de simulações, suporta o controle dos modelos (por meio de *scripts* e/ou interface gráfica com o usuário), a visualização de suas variáveis de estado e fornece o tempo de simulação. Também pode fornecer outros serviços, como salvar, restaurar e registrar eventos.

Um ambiente de gerenciamento de dados e processos de simulação, além de proporcionar as funcionalidades de um “ambiente de simulação”, também conta com funcionalidades para gerenciar os dados e processos de simulação gerados neste ambiente. Estas funcionalidades foram discutidas em detalhe na seção 2.6.3.

4.2. **SOFTWARE *FRAMEWORK* RCE**

4.2.1. Histórico

O RCE é um software *framework* de código aberto cujo desenvolvimento iniciou-se em 2005 no Centro Espacial Alemão (DLR) e, desde então, está em constante evolução. O escopo inicial do projeto era suportar o trabalho colaborativo e distribuído na indústria naval (SEIDER et al., 2012). Muitos requisitos de desenvolvimento de produtos desta indústria são comuns à indústria espacial. Ambas desenvolvem sistemas complexos usando o conhecimento de diversas disciplinas que possuem forte dependência entre si. Assim, o RCE chamou a atenção de outras equipes no DLR que o aplicaram, com sucesso, em casos de uso aeronáuticos e espaciais, tornando-se a plataforma de referência de colaboração no DLR também nestas áreas.

4.2.2. Requisitos

O RCE foi desenvolvido desde o princípio com um grande cuidado para que pudesse ser reusado de maneira eficaz por vários outros projetos dentro do DLR. Segundo Sommerville (2011) um software *framework* é um conjunto integrado de artefatos de software, como classes, objetos e componentes, que colaboram entre si para prover uma arquitetura reusável, por meio de especialização e adição de objetos, para uma família de aplicações relacionadas. De maneira diferente de bibliotecas de software especializadas em alguma função, software *frameworks* são responsáveis pelo fluxo de controle da aplicação, o que é chamado de “inversão de controle”. Quando bibliotecas são reusadas, existe uma aplicação principal desenvolvida pelo usuário que chama estas bibliotecas no momento em que são necessárias. No caso de um software *framework*, ele é a aplicação principal, que chama aplicações desenvolvidas pelo usuário, ou extensões do software *framework*, seguindo padrões de interface bem definidos. Assim, software *frameworks* são mais fáceis de reusar, pois é necessário um conhecimento menor da aplicação

principal, bastando obedecer aos padrões de interface ao desenvolver as extensões.

Mesmo que, por definição, software *frameworks* sejam criados para serem reusados, eles podem ser desenvolvidos de maneira mais genérica ou mais especializada, neste caso procurando antever todas as necessidades do usuário e procurando resolvê-las. Entretanto, quanto mais especializado for um software *framework*, maior será a dificuldade em ser reusado em uma gama ampla de possíveis aplicações. Uma solução de compromisso entre generalidade e especialização precisa ser encontrada. No caso do RCE, os seguintes requisitos de reuso foram estabelecidos (SEIDER et al., 2012):

a) **modularidade**: o software *framework* deve ter uma arquitetura baseada em componentes, de forma que os usuários possam decidir quais componentes reusar ou não;

b) **extensibilidade**: o software *framework* deve ser extensível para que os desenvolvedores possam tanto estender suas funcionalidades para atender a seus requisitos individuais ou deixar que outras ferramentas de domínio específico interajam com ele;

c) **portabilidade**: o software *framework* deve poder ser instalado em diversas plataformas e sistemas operacionais, de forma a se adaptar às aplicações de domínio específico que interagem com ele, e não o contrário;

d) **abertura**: o software *framework* deve ser livre de licenças de *software*, podendo ser livremente distribuído.

Estes requisitos são relacionados apenas ao reuso do software *framework*. Os requisitos a seguir foram usados como referência para o desenvolvimento das funcionalidades de alto nível que o RCE oferece (SEIDER et al., 2012):

a) **distribuição**: o software *framework* deve prover a capacidade de distribuir dados e ferramentas em um ambiente comum, acessível por todos os usuários. Desta forma, um usuário deve ser capaz de executar, a partir de seu computador, um *software* que esteja disponível apenas em outro computador pertencente ao mesmo ambiente;

b) **gerenciamento de dados**: o software *framework* deve prover a capacidade de gerenciamento de dados científicos. Dados gerados em um estudo colaborativo podem estar espalhados em várias plataformas de hardware, mas sob o ponto de vista do usuário, devem ser exibidos no sistema em uma interface gráfica que agrupe todos os dados do estudo em uma mesma visualização. Usuários devem ser capazes de armazenar e recuperar dados de qualquer tipo e tamanho, além de realizar buscas de dados que respeitem os privilégios de acesso aos mesmos;

c) **gerenciamento de privilégios de acesso**: o software *framework* deve prover as capacidades de autorização e autenticação de usuários. Em um ambiente compartilhado por pesquisadores, fornecedores e desenvolvedores, privilégios de acesso precisam ser bem definidos para proteger dados e ferramentas do uso incorreto. Apenas usuários qualificados devem ser capazes de armazenar e editar dados de um determinado subsistema, por exemplo;

d) **motor de fluxo**: o software *framework* deve prover funcionalidade de acoplar várias ferramentas em uma determinada sequência e executá-las na ordem estabelecida. Os arquivos de saída gerados por ferramentas em uma etapa anterior do fluxo devem poder ser usados como entradas de ferramentas em uma etapa posterior. Em um fluxo colaborativo de ferramentas e dados, qualquer usuário deve ser

capaz de contribuir com uma etapa do fluxo, provendo dados e ferramentas que podem ser acessadas pelos demais usuários;

e) **interface gráfica**: o software *framework* deve prover elementos gráficos genéricos para cada componente, como, por exemplo, tela de *login* de acesso, visualização dos dados de um fluxo e representação gráfica do mesmo em formato de fluxograma. A interface deve ser extensível para permitir que desenvolvedores a modifiquem, de forma a adaptá-la a futuras implementações. Sob o ponto de vista do usuário, o sistema deve ser fácil de aprender, eficiente na operação e agradável de usar;

f) **independência de plataforma**: o software *framework* deve poder ser executado em diferentes sistemas operacionais. Como visto no item d), o software *framework* integra várias ferramentas diferentes em um mesmo motor de fluxo. Estas ferramentas podem rodar em diferentes sistemas operacionais e o software *framework* deve poder ser instalado nas máquinas onde estas ferramentas são executadas.

4.2.3. Arquitetura do sistema

Definidos os requisitos de reuso, o DLR decidiu utilizar o *Eclipse Rich Client Platform* (RCP) (MCAFFER, 2005) como base do software *framework* RCE. Esta plataforma de código aberto é muito usada no mundo acadêmico, para desenvolvimento de software em pesquisas científicas, e também nas indústrias aeroespacial e automotiva. Uma das principais razões para a popularidade do Eclipse é a utilização do Java, uma linguagem moderna e disponível livremente para os desenvolvedores (SEIDER et al., 2012).

Segundo a OSGi Alliance (2016), a tecnologia OSGi (*Open Service Gateway Initiative*) é um conjunto de especificações que definem um sistema modular e uma plataforma de serviços para a linguagem de programação Java, que implementa um modelo de componente completo e dinâmico, inexistente em

ambientes Java independentes. Estas especificações permitem um modelo de desenvolvimento em que os aplicativos são dinamicamente compostos de diferentes pacotes reutilizáveis.

O Eclipse usa as especificações OSGi por meio do Equinox, uma implementação da especificação OSGi mantida pelos mesmos desenvolvedores do Eclipse. Isto permite que o requisito de modularidade para reuso seja atendido, pois o código dos componentes do RCE pode ser encapsulado em pacotes, que podem ser carregados ou descarregados dinamicamente pelo software *framework*. Os pacotes podem também fornecer serviços que são consumidos por outros pacotes. Assim, uma aplicação aeroespacial que seja desenvolvida a partir do RCE pode escolher quais componentes usar ou não (SEIDER et al., 2012). A Figura 4.1 mostra uma aplicação aeroespacial desenvolvida a partir do RCE, que pode usar em paralelo todo o software *framework* Eclipse RCP, além de componentes individuais do RCE.

Figura 4.1 - Interação do RCE com Eclipse RCP/OSGi e Aplicação Aeroespacial.

Fonte: Adaptado de Seider et al. (2012). (tradução nossa).

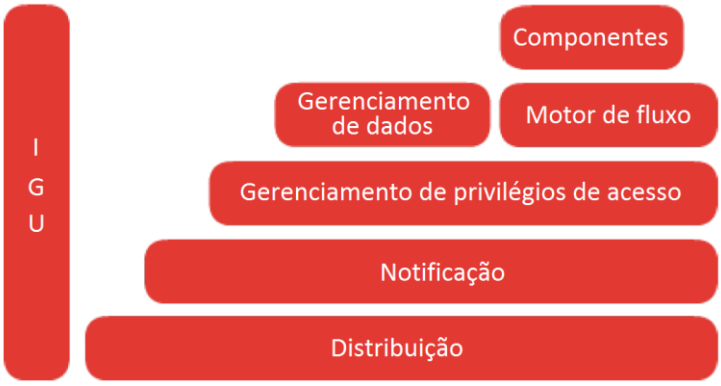
4.2.4. Componentes

O RCE foi desenvolvido com o objetivo de trazer soluções às dificuldades em colaboratividade e gerenciamento de dados e processos de simulação mostrados na seção2.6.2. O desenvolvimento foi baseado nos requisitos de reuso da plataforma e requisitos de funcionalidades de alto nível mostrados na seção 4.2.2.

As funcionalidades de alto nível no RCE foram desenvolvidas como pacotes que podem ser individualmente reusados por uma aplicação baseada neste software *framework*. Estes pacotes, ou componentes do RCE, assim como suas dependências, podem ser vistos na Figura 4.2.

Nesta figura, componentes de software em um nível superior fazem uso dos componentes em nível inferior. Por exemplo, o componente de Gerenciamento de dados consome serviços do Gerenciamento de privilégios de acesso, Notificação e Distribuição. Por sua vez, o componente de Notificação consome apenas serviços do componente de Distribuição. O componente IGU (Interface Gráfica com Usuário) acessa todos os demais componentes e seus serviços (SEIDER et al., 2012).

Figura 4.2 - Componentes do RCE e suas dependências.



Fonte: Adaptado de Seider et al. (2012). (tradução nossa).

O funcionamento de cada componente do RCE é detalhado a seguir:

a) **distribuição**: este componente é a base do software *framework* RCE e permite a formação de uma rede virtual entre vários computadores que têm uma instância do RCE em execução, chamados de *nós* da rede. Um serviço disponibilizado em um destes nós pode ser acessado por qualquer outro nó. Ao invés de uma arquitetura cliente-servidor, o RCE é baseado em uma abordagem *peer-to-peer* (SEIDER et al., 2013), onde cada par de nós pode se comunicar de

forma transparente entre si. Os protocolos de comunicação suportados são o RMI (*Remote Method Invocation*) e o SOAP;

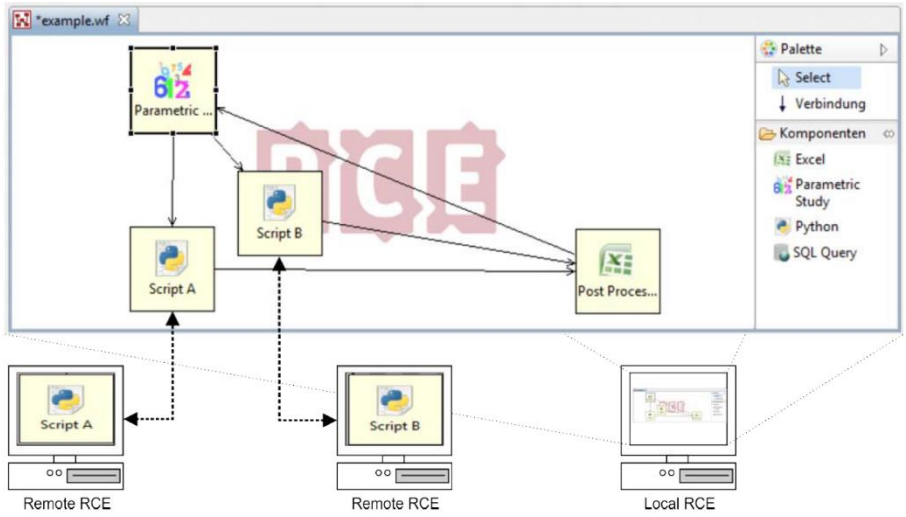
b) **notificação**: este componente é responsável pela geração de mensagens de *status* dos serviços do RCE rodando em cada instância. Como este componente consome o serviço de distribuição, as notificações são distribuídas entre as instâncias da mesma rede virtual. Assim, uma instância do RCE pode notificar a outra, enviando uma mensagem como “rodada do *workflow* finalizada” ou “instância remota inoperante” (SEIDER et al., 2012);

c) **gerenciamento de privilégios de acesso**: este componente é responsável pelos serviços de autenticação e autorização. O primeiro verifica se o usuário é realmente quem ele afirma ser. O segundo verifica os privilégios que o usuário tem no projeto, como de apenas leitura em um subsistema e de escrita em outro subsistema, por exemplo. Como este componente consome os serviços de notificação, é possível, por exemplo, que uma instância notifique as outras de que um novo usuário logou no sistema (SEIDER et al., 2012);

d) **gerenciamento de dados**: este componente é responsável pelas funcionalidades de busca e armazenamento de dados. Este gerenciamento é descentralizado, ou seja, cada instância do RCE tem um gerenciamento de dados local. Entretanto, como este componente consome serviços de gerenciamento de privilégios de acesso, notificação e distribuição, um usuário rodando RCE em uma instância local tem acesso aos dados publicados em todas as outras instâncias, respeitando seus privilégios de acesso. Embora os dados estejam distribuídos em várias instâncias, o usuário os percebe centralizados, pois na interface gráfica são mostrados agrupados no mesmo contexto (SEIDER et al., 2012);

e) **motor de fluxo**: este componente permite que o RCE realize a integração entre várias ferramentas por meio da criação de um fluxo colaborativo de dados e ferramentas (*workflow*). Neste fluxo, os dados de saída de uma ferramenta podem ser definidos como dados de entrada de outra e a execução das mesmas é automatizada. Ferramentas e dados são disponibilizados localmente em cada instância do RCE, mas como o componente de motor de fluxo consome os demais serviços mencionados anteriormente, ferramentas e dados locais tornam-se disponíveis para as demais instâncias, respeitando os privilégios de acesso de cada usuário. Isto pode ser visto na Figura 4.3, que mostra duas instâncias remotas fornecendo as ferramentas "*Script* A" e "*Script* B" que são usadas em um fluxo criado em uma instância local, que lê parâmetros locais, roda as duas ferramentas e armazena os resultados em uma planilha (SEIDER et al., 2012);

Figura 4.3 - Exemplo de fluxo colaborativo de ferramentas e dados.



Fonte: Seider et al. (2012).

f) **interface gráfica com usuário (IGU)**: este componente é composto de elementos básicos de interface, disponibilizados em pacotes individuais, frequentemente utilizadas por aplicações que usam o

RCE. Estes elementos variam desde uma janela de *login* até uma visualização de dados consumidos e produzidos em um fluxo colaborativo de ferramentas e dados;

g) **componentes:** no topo da Figura 4.2 é mostrado um bloco de "Componentes" que são as aplicações de software desenvolvidas pelos usuários do RCE para serem utilizadas em *workflows* colaborativos. Estando no topo, estes componentes fazem uso de todas as funcionalidades mencionadas acima dos demais componentes estruturais do RCE.

4.2.5. Configuração da rede RCE

O RCE não possui um programa de instalação. O software é fornecido em um arquivo no formato ZIP, contendo um executável "*rce.exe*" e demais arquivos de apoio, que devem ser descompactados em uma área local do computador do usuário. Por ser um software de código aberto, é possível também obter o código-fonte e realizar a compilação do executável. Quando o RCE é executado pela primeira vez, é criada a pasta oculta "*.rce"* na pasta *home* do usuário, que no Windows é a pasta "*C:**users*\<*login do usuário*>\". Subpastas nesta pasta serão utilizadas mais tarde pelo componente de gerenciamento de dados do RCE para armazenar todos os dados e metadados utilizados nas simulações que o usuário executar. Nota-se, assim, que os dados produzidos pelo usuário e os dados da instalação do RCE estão localizados em pastas distintas.

Dentro da pasta *".rce*" são criadas duas outras pastas: "*common"* e "*default"*. Na pasta "*default"* é criado um arquivo padrão de perfil de usuário (*user profile*) chamado "*configuration.json*", que pode ser editado posteriormente. AFigura 4.4mostra um exemplo do conteúdo deste arquivo, escrito no formato JSON (*JavaScript Object Notation*). Cada perfil de usuário está associado a uma instância (*instance*) do RCE. Múltiplas instâncias do RCE conectadas entre si dão origem à uma rede RCE.

A forma de configuração das instâncias RCE é por meio da edição dos arquivos "*configuration.json*" correspondentes. Uma lista de referência com todos os parâmetros de configuração editáveis pode ser encontrada no *RCE User Guide* (RCE, 2017).

Figura 4.4 – Arquivo de configuração padrão do RCE.

```
{
    "general" : {
        "instanceName" : "Default instance started by
        \"${systemUser}\" on ${hostName}"
    },
    "network" : {
        "connections" : {
            "exampleConnection1" : {
                "host" : "127.0.0.1",
                "port" : 20001,
                "connectOnStartup": false,
                "autoRetryInitialDelay" : 5,
                "autoRetryMaximumDelay" : 300,
                "autoRetryDelayMultiplier" : 1.5
            }
        }
    }
}
```

Fonte: Produção do autor.

As instâncias do RCE podem ser iniciadas no modo *cliente* ou no modo *serviço*. No modo *cliente*, o usuário abre a interface gráfica do RCE e interage manualmente com ela. No modo *serviço*, toda vez que o computador do usuário é ligado, uma instância RCE é automaticamente iniciada (sem lançar a interface gráfica). Em ambos os modos, quando a instância é iniciada, ela estabelece conexão com as demais instâncias e os componentes associados a ela são disponibilizados na rede RCE para os demais usuários.

De acordo com Seider et al. (2013), existem três tipos de instâncias possíveis em uma rede RCE, detalhadas a seguir:

a) **máquinas clientes:** instâncias padrão, utilizadas pelos usuários da rede RCE para acessar aplicações e *workflows* disponibilizados por outros usuários e executar simulações. Máquinas clientes devem

estar conectadas a um Servidor *Relay* para terem acesso à rede RCE;

b) **servidores *relay*:** responsáveis por conectar as máquinas clientes entre si, servindo como “ponte” entre elas. Uma rede RCE deve ter pelo menos um servidor *relay*;

c) **nós de computação:** são instâncias geralmente iniciadas no modo serviço, disponibilizando recursos de software e execução remota de simulações para as máquinas clientes.

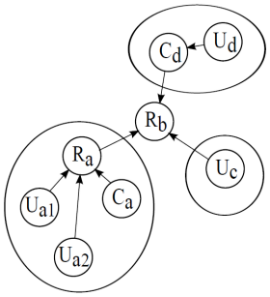
Uma arquitetura típica de rede de computadores é a “cliente-servidor”, onde existe a figura de um servidor central que controla todas as interações da rede. Embora esta arquitetura seja simples de implementar, as redes resultantes são relativamente estáticas, o que não é recomendável em ambientes onde há intensa colaboração. Para entregar maior flexibilidade, as redes RCE foram concebidas com uma arquitetura ponto-a-ponto (*peer-to-peer*), onde qualquer instância do RCE pode aceitar conexões de outras instâncias. Todas as instâncias que estão ligadas direta ou indiretamente são combinadas em uma rede virtual que é independente da(s) rede(s) física(s) onde estas instâncias estão conectadas. Isto permite que tanto redes virtuais transitórias ou de longa duração sejam facilmente configuradas.

Segundo RCE (2017), o tráfego padrão na rede RCE não é criptografado, sendo recomendada sua configuração apenas dentro de uma rede segura, como a rede interna de uma instituição de pesquisa como o INPE. Existe a possibilidade de criar conexões seguras do tipo SSH (*Secure Shell*) no RCE, mas nem todas as funcionalidades estão disponíveis para este tipo de conexão.

Um exemplo de configuração de rede RCE é mostrada na Figura 4.5, onde “R” representa servidores *relay*, “U” representa máquinas clientes de usuários da rede e “C” representa nós de computação. Nesta configuração, o grupo de

usuários “a” é representado pelas instâncias U_{a1}, U_{a2} e C_a, o grupo de usuários “d” é representado pelas instâncias C_d e U_d e o grupo de usuários “c” é representado apenas pela instância U_c. Cada um destes grupos reside em uma rede física diferente, possivelmente em instituições de pesquisa diferentes, cujas redes não se comunicam. No grupo “a”, as instâncias U_{a1}, U_{a2} e C_a têm, em seu arquivo de configuração, o endereço IP do servidor *relay* R_a, que por sua vez informa a estas instâncias os endereços das demais instâncias da rede virtual, permitindo a conexão direta entre elas. Assim, por exemplo, U_{a1} pode se conectar diretamente a C_a, que é um nó de computação, consumindo recursos disponibilizados nesta instância. No grupo “d”, o nó de computação C_d faz também o papel de servidor *relay* para U_d, permitindo que estas duas instâncias troquem informações entre si. Isto mostra a flexibilidade do RCE, permitindo que uma mesma máquina atue como nó de computação e também como servidor *relay*. O grupo “c” é formado por apenas uma instância, U_c. Para que estes três grupos sejam parte da mesma rede RCE, permitindo que todas as instâncias troquem informações entre si, o servidor *relay* R_a, o nó de computação C_d e a máquina cliente de usuário U_c têm, em seu arquivo de configuração, o endereço IP do servidor *relay* R_b, que reside em uma área pública (por exemplo, em uma DMZ), acessível a todas estas instâncias. Desta forma, por exemplo, quando uma nova instância U_{a3} é conectada ao servidor *relay* R_a, esta informação é propagada para todas as demais instâncias, permitindo a conexão direta entre elas. Assim, esta nova instância poderia consumir recursos do nó de computação C_d, por exemplo (Seider et al., 2013).

Figura 4.5 – Exemplo de configuração de rede RCE.

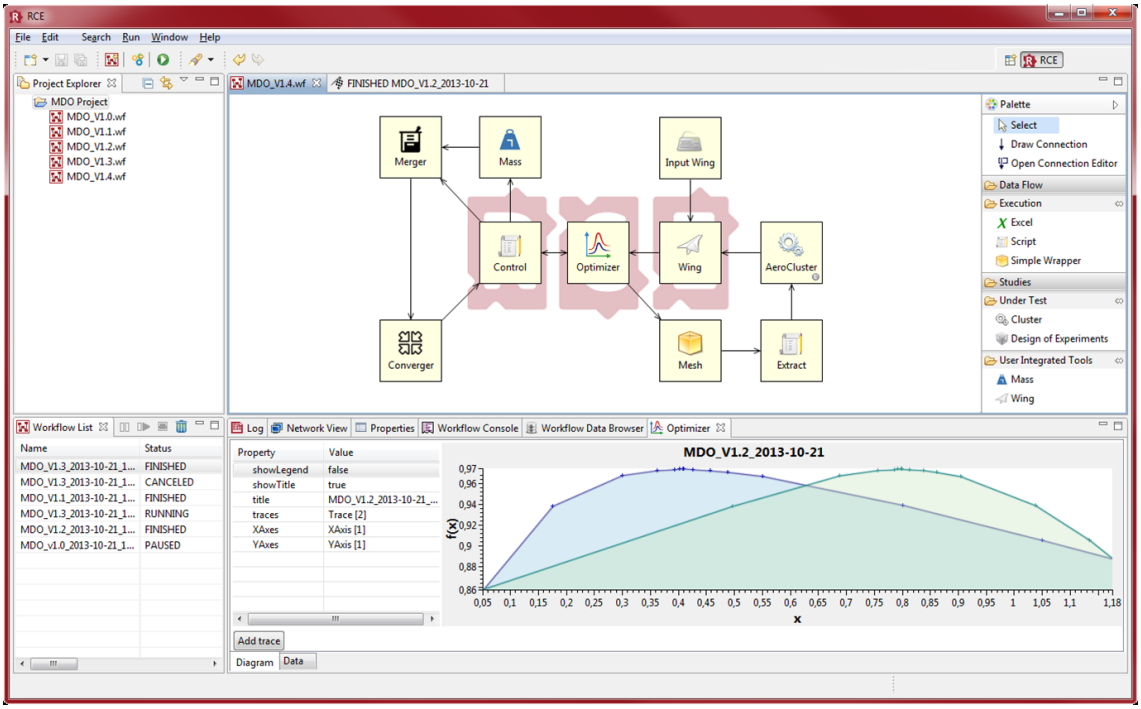


Fonte: Seider et al. (2013).

4.2.6. Interface gráfica

A interface gráfica do RCE é composta de diversos editores e visualizações que podem ser (re)arranjados pelo usuário, tornando-se visíveis ou não, de acordo com sua conveniência. A Figura 4.6apresenta um exemplo da configuração da interface, cujos elementos principais são detalhados a seguir:

Figura 4.6 – Interface gráfica do RCE com exemplos de visualizações e editores.



Fonte: RCE (2017).

a) ***Project Explorer***: mostrado no canto superior esquerdo, é utilizado para gerenciar projetos, que são pastas contendo a estrutura de dados que será utilizada para a execução de *workflows*;

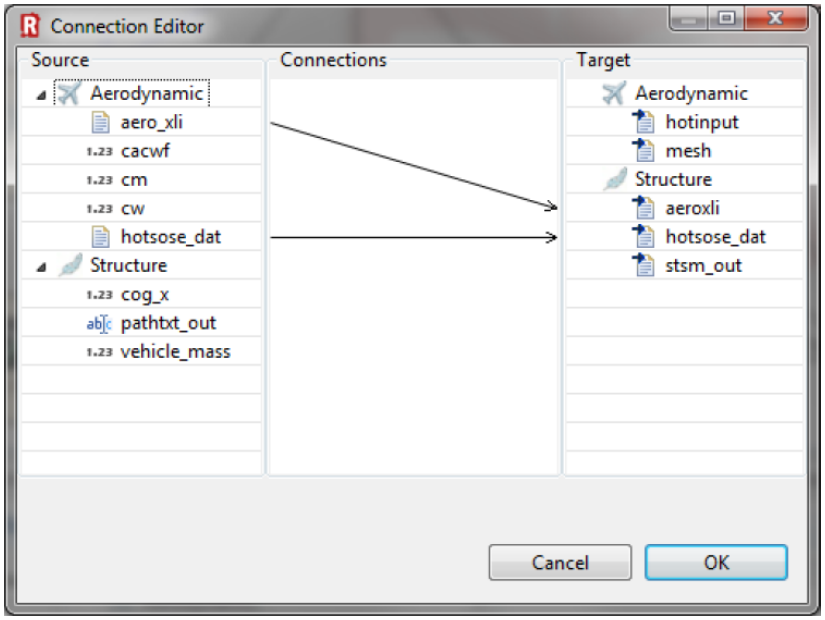
b) ***Workflow List***: mostrado no canto inferior esquerdo, é utilizado para listar todos os *workflows* ativos e gerenciá-los, permitindo ações como parar, pausar, reiniciar e descartar execuções de *workflows*;

c) ***Workflow Editor***: mostrado na parte superior central, é a visualização principal do RCE, utilizada para construir e configurar a execução de *workflows*;

d) ***Palette***: mostrada no canto superior direito, é utilizada para listar todos os componentes disponíveis na rede RCE para serem utilizados em *workflows*. Estes componentes podem estar tanto disponíveis no próprio computador do usuário, quanto em outros componentes da rede RCE, e incluem tanto componentes estruturais do RCE quanto ferramentas desenvolvidas pelos usuários. Possui também um atalho para acessar o *Connection Editor*, mostrado a seguir;

e) ***Connection Editor***: mostrado na Figura 4.7, é um editor utilizado para mapear o fluxo de dados entre os componentes do *workflow*. Cada componente tem mapeado seus dados de entrada e saída. Neste editor, associa-se, por exemplo, um dado de saída de um componente com um dado de entrada de outro componente que será executado em seguida no *workflow*;

Figura 4.7 – Janela do *Connection Editor* do RCE.

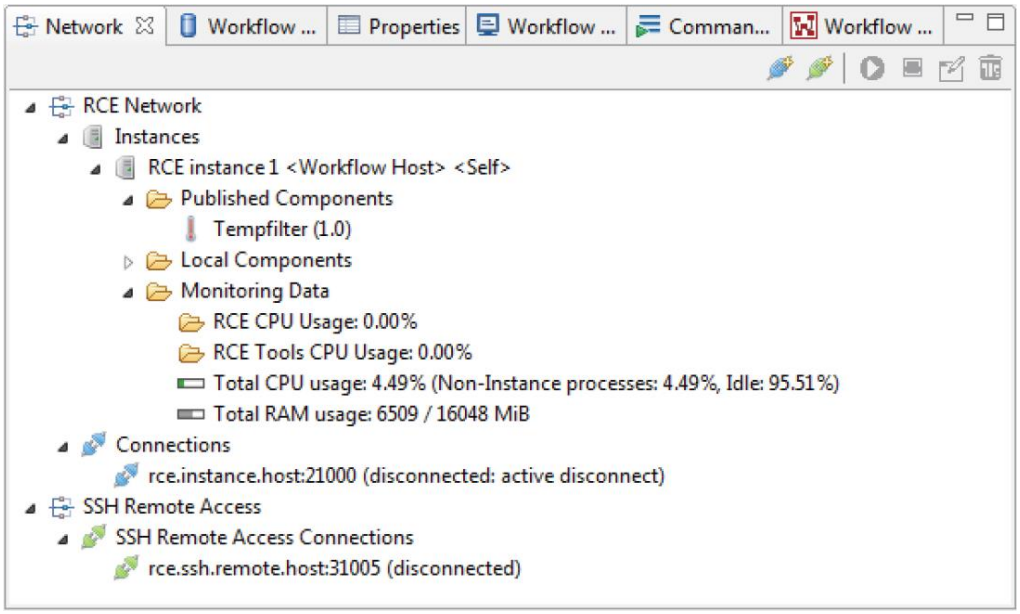


Fonte: RCE (2017).

f) ***Log***: mostrado na parte inferior central, esta visualização mostra todas as saídas de registro de execução de *workflows* do RCE, como, por exemplo, mensagens de erro durante as execuções;

g) ***Network View***: mostrado em detalhes na Figura 4.8, esta visualização mostra todas as instâncias ativas na rede RCE, assim como os componentes disponibilizados por elas para reuso. É possível também administrar todas as possíveis conexões ativas da instância em uso com as demais e verificar utilização de CPU e memória RAM de todas as instâncias;

Figura 4.8 – Janela do *Network View* do RCE.



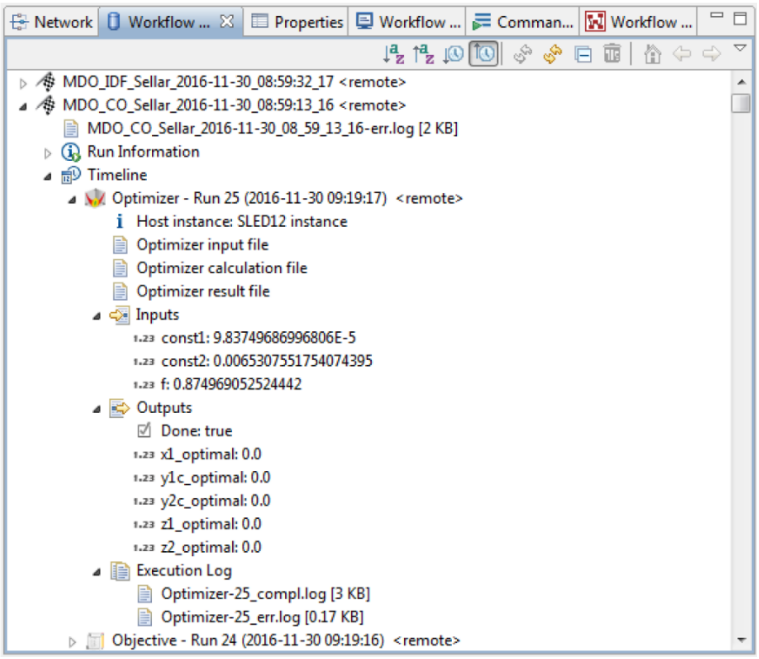
Fonte: RCE (2017).

h) ***Workflow Data Browser***: mostrado em detalhes na Figura 4.9, mostra todos os dados utilizados nas execuções de *workflows*, assim como metadados capturados pelo RCE, como identificação do usuário e instância onde o *workflow* foi executado, tempo de execução, etc. Esta visualização é importante porque, por meio dela, é possível acessar dados de entrada e saída de todos os *workflows*

executados na rede RCE, independentemente se foram executados na instância local ou em instâncias remotas;

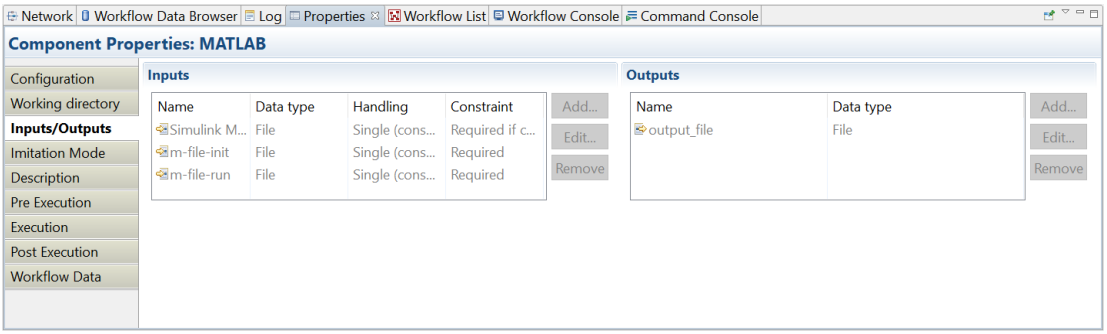
i) ***Properties***: mostrado em detalhes na Figura 4.10, é uma visualização adaptativa, que muda para cada componente de *workflow* selecionado no *Workflow Editor*. Mostra, para cada componente, suas configurações, como dados de entrada e saída e *scripts* de pré e pós-execução;

Figura 4.9 – Janela do *Workflow Data Browser* do RCE.



Fonte: RCE (2017).

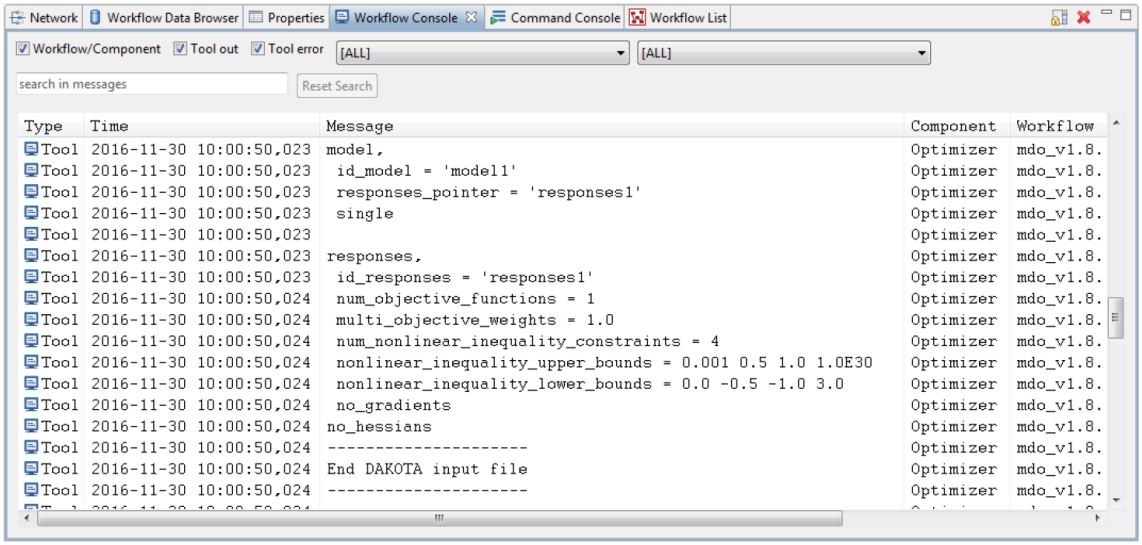
Figura 4.10 – Janela Properties do RCE.



Fonte: Produção do autor.

j) ***Workflow Console***: mostrado em detalhes na Figura 4.11, é uma visualização que apresenta todas as saídas de texto geradas pelos componentes dos *workflows*. Por exemplo, se durante a execução de um componente, este precisa exibir uma mensagem na tela, ela é mostrada no *Workflow Console*.

Figura 4.11 – Janela Workflow Console do RCE.



Fonte: RCE (2017).

4.2.7. Integração e distribuição de ferramentas

Segundo Seider et al. (2013), um dos pré-requisitos para colaboração em *workflows* multidisciplinares é a utilização de um software *framework* que permita a um usuário disponibilizar uma aplicação de software (ferramenta) desenvolvida por ele em um ambiente acessível por outros usuários, para ser reusada. Esta é uma das principais funcionalidades no RCE, tanto que o seu nome, que significa “ambiente de componentes remotos” (*Remote Component Environment*), provém desta funcionalidade. Como já visto, “componente”, neste caso, significa uma ferramenta compartilhada neste ambiente.

Para que uma ferramenta possa ser integrada no RCE, existem alguns requisitos que precisam ser atendidos (RCE, 2017):

a) a ferramenta deve ser executada por meio de uma única linha de comando, sem outras interfaces gráficas com o usuário;

b) a ferramenta deve ser executada sem a intervenção do usuário, de modo não interativo;

c) todos os dados de entrada devem estar disponíveis antes da execução da ferramenta;

d) os dados de entrada devem ser apenas parâmetros e arquivos;

e) todos os dados de entrada devem estar localizados em uma pasta específica;

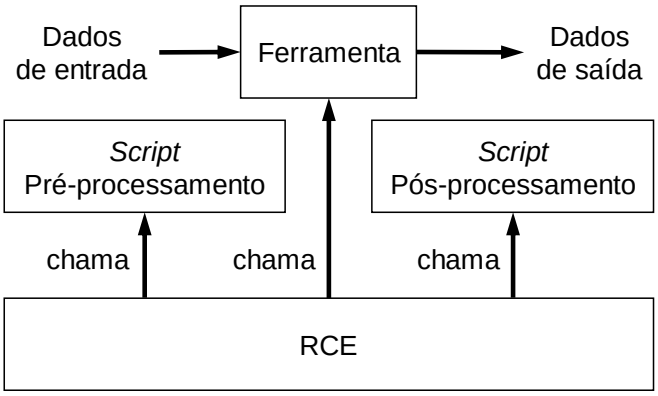
f) todos os dados de saída devem ser escritos em uma pasta específica.

Segundo Seider et al. (2013), no conceito de integração do RCE, ferramentas são tratadas como “caixas-pretas”, ou seja, apenas suas entradas e saídas são expostas ao usuário, que não tem acesso às suas configurações internas. Este conceito é mostrado na Figura 4.12. O RCE executa as ferramentas como se elas estivessem sendo executadas por meio de uma linha de comando, incluindo os parâmetros necessários. É possível executar *scripts* opcionais de pré e pós-processamento, antes e depois da execução da ferramenta, respectivamente, para realizar um tratamento dos dados de entrada e saída. Isto é especialmente importante no caso da construção de *workflows* incluindo estas ferramentas, pois dependendo das particularidades das ferramentas que venham antes ou depois, pode ser necessário adaptar ou converter o formato dos dados antes de se executar a ferramenta. Enquanto o RCE executa a ferramenta, todas as mensagens que seriam escritas no console (no caso de se executar manualmente via linha de comando) são direcionadas para o *Workflow Console*, visualização mostrada na Figura 4.11.

Para se integrar uma nova ferramenta no RCE, existe um guia passo-a-passo na própria interface gráfica que orienta o usuário a fornecer todos os dados

necessários para esta configuração. Uma vez integrada, a ferramenta fica disponível para todos os integrantes da rede RCE, aparecendo na visualização "*Pallete*" mostrada na Figura 4.6, no subitem "*User Integrated Tools*". Quando outro usuário reusa esta ferramenta, embora pareça que a execução seja local sob o ponto de vista deste usuário, na verdade os dados de entrada são automaticamente copiados do computador do usuário para a máquina onde está publicada a ferramenta, a execução é realizada nesta máquina e depois os arquivos de saída são automaticamente enviados para a máquina do usuário. A execução da ferramenta na máquina onde foi publicada facilita as atualizações de versão e garante que os usuários estejam utilizando a versão correta da ferramenta. Em outros ambientes de trabalho sem RCE, usuários podem copiar o executável de ferramenta para suas máquinas locais, o que dá margem ao uso de versões desatualizadas das mesmas. Uma vez que um desenvolvedor "congele" a configuração de uma ferramenta, pois ela se encontra estável, sendo executada sem erros, ele pode movê-la de sua máquina cliente para um nó de computação, que a princípio está sempre ligado e disponível para os demais usuários (SEIDER et al., 2013).

Figura 4.12 – Conceito de integração de ferramentas no RCE.



Fonte: Adaptado de Seider et al. (2013). (tradução nossa).

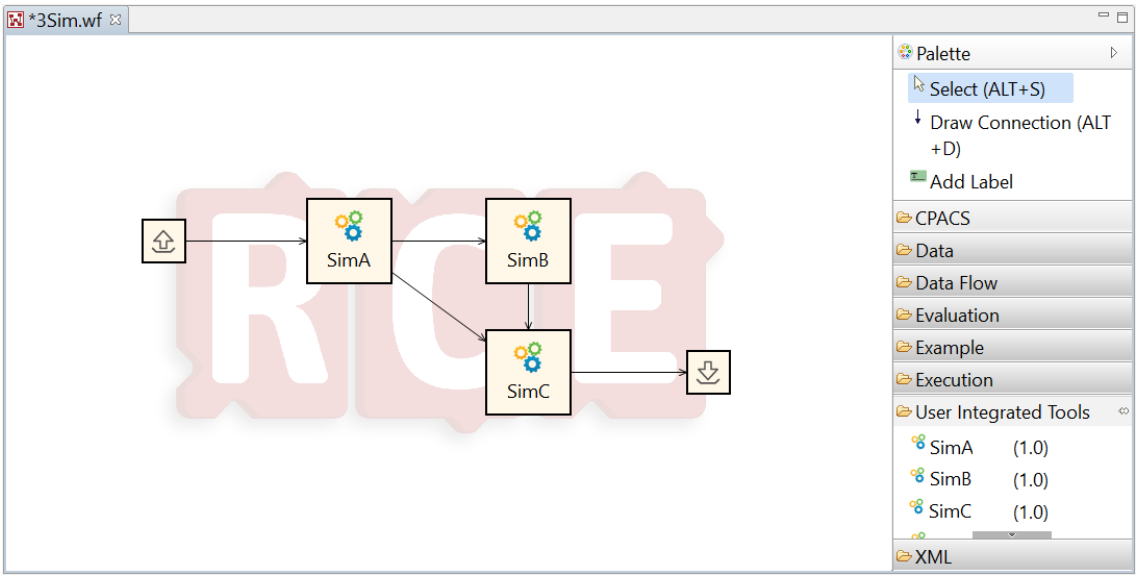
4.2.8. Workflows

Segundo o RCE *User Guide* (RCE, 2017), o RCE foi desenvolvido para a execução automatizada e distribuída de *workflows*, que são compostos por

componentes acoplados entre si. *Workflows* são criados dentro de projetos no RCE, que contêm o arquivo de definição de *workflow*, assim como todos os dados associados.

A edição dos *workflows* é feita no *Workflow Editor* do RCE. Neste editor, os componentes são selecionados a partir da *Palette* que contém tanto os componentes padrão do sistema quanto os integrados pelo usuário (*User Integrated Tools*). A Figura 4.13 mostra um exemplo de edição do *workflow* intitulado “3Sim”, usando um processo de simulação mostrado anteriormente na Figura 2.12. Os três componentes integrados pelo usuário (SimA, SimB e SimC) são arrastados da *Pallete* para a área de trabalho do editor e conectados entre si. Pode-se notar que as setas mostram o relacionamento entre os componentes, mas não determinam como os dados são trocados entre eles. Esta troca é configurada no *Connection Editor* do RCE, e a configuração resultante pode ser vista na Figura 4.14.

Figura 4.13 – Exemplo de edição de *workflow* no RCE.

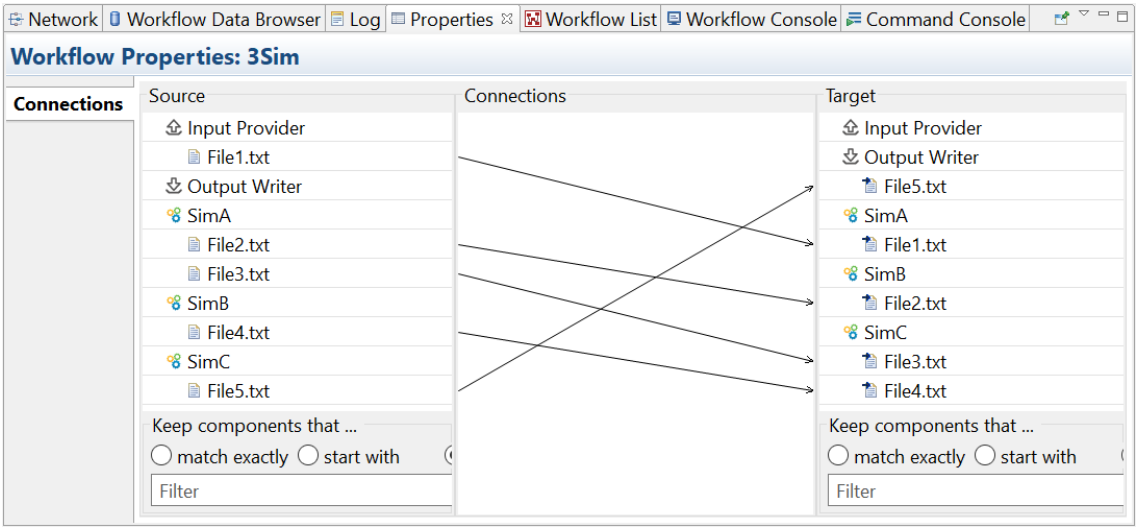


Fonte: Produção do autor.

Além dos três componentes integrados pelo usuário, foram utilizados também dois componentes nativos do RCE, um *Input Provider* (componente que provê dados de entrada) e um *Output Writer* (componente que escreve dados de

saída em um local determinado previamente). Pode-se notar que o fluxo de dados modelado no RCE e mostrado na Figura 4.14 é o mesmo mostrado anteriormente na Figura 2.12.

Figura 4.14 – Conexões de dados em um *workflow* do RCE.



Fonte: Produção do autor.

A ordem de execução de componentes em um *workflow* do RCE é orientada a dados. Isto significa que um componente só irá ser executado quando todos os arquivos identificados como entrada (*Inputs*) estiverem disponíveis. Por exemplo, no caso do *workflow* 3Sim, ao se iniciar a execução, apenas o arquivo "*File1.txt*" está disponível (fornecido pelo componente *Input Provider*), então o componente SimA é executado, gerando os arquivos "*File2.txt*" e "*File3.txt*" como saídas. O componente SimB tem como entrada apenas o arquivo "*File2.txt*", enquanto que o componente SimC tem como entradas os arquivos "*File3.txt*" e "*File4.txt*". Como o arquivo "*File4.txt*" ainda não está disponível (será gerado pelo componente SimB), o próximo componente a ser executado é o SimB, seguido então pelo SimC.

O RCE conta também com um recurso de verificação manual dos resultados de um componente durante a execução de um *workflow*. Se um componente é configurado com este recurso, a execução do *workflow* é suspensa após os resultados do componente serem gerados, para que o usuário confira os

resultados. Se tudo correr como esperado, o usuário autoriza a continuação da execução dos próximos componentes do *workflow*.

O RCE permite que usuários desenvolvam ferramentas e as compartilhem na rede RCE como componentes. O mesmo pode ser realizado com *workflows*. Por exemplo, o *workflow* 3Sim mostrado nesta seção poderia ser configurado para ser acessado remotamente (*Remote Workflow Access*) e outros usuários poderiam reusá-lo, fornecendo o arquivo de entrada requerido e obtendo o arquivo de saída gerado pelo *workflow* que, neste caso, é visto como uma "caixa-preta" pelo usuário.

4.2.9. Gerenciamento de dados

Os dados em uma dada instância do RCE residem em duas áreas principais: na pasta padrão *storage* localizada em "*C:**users*\<*login do usuário*>*.rce**default**storage*", definida pelo RCE, e na pasta *workspace*, definida pelo usuário. Na pasta *workspace* são armazenados todos os dados temporários utilizados na execução dos *workflows* e os próprios arquivos de definição dos *workflows* (com extensão ".wf"). Na pasta *storage*, o RCE armazena um banco de dados Apache Derby (banco de dados relacional de código aberto implementado em Java).

Ao se executar um *workflow* no RCE, os dados de entrada e saída estão localizados na pasta *workspace* e podem ser vistos na interface gráfica do RCE por meio da visualização *Project Explorer*, mostrada na Figura 4.6. Toda a estrutura de pastas e arquivos utilizada pelos *workflows* é mostrada utilizando-se este recurso. Simultaneamente à execução do *workflow*, o RCE grava uma série de metadados e todos os arquivos relacionados no banco de dados Derby, de forma criptografada. Estes metadados e arquivos podem ser consultados na visualização *Workflow Data Browser* do RCE, mostrada na Figura 4.9.

Após uma primeira execução de um *workflow*, pode-se alterar os arquivos de entrada para a realização de testes e se executar novamente o *workflow,* obtendo-se novos resultados. Ao fazer isto, os dados utilizados na primeira execução são sobrescritos na pasta *workspace*, perdendo-se a sua referência. Entretanto, o RCE grava todos os dados e metadados da segunda execução como uma nova entrada do banco de dados. Assim, no *Workflow Data Browser,* os dados de ambas as execuções são preservados, podendo-se rastrear as diferenças entre os dados de entrada.

O *Workflow Data Browser* mostra não apenas os dados e metadados de *workflows* executados na instância local do usuário, mas também de todas as demais instâncias conectadas à rede RCE. Assim, é possível recuperar dados e metadados utilizados por outros usuários da mesma rede, que colaboram, por exemplo, em uma *Concurrent Design Facility*.

4.3. **CHAMALEON**

O *Chamaleon* é uma aplicação aeronáutica desenvolvida pelo DLR que tem como objetivo permitir aos seus usuários a realização de simulações colaborativas durante o projeto preliminar de aeronaves. Isto é alcançado por meio do reuso do *software framework* RCE, que teve alguns componentes estendidos para atender a necessidades específicas da área aeronáutica (SEIDER et al., 2012).

O projeto preliminar de aeronaves envolve o uso de várias ferramentas que são integradas no *Chamaleon* por meio do reuso do componente de motor de fluxo do RCE. Como cada ferramenta tem seus formatos próprios de entrada e saída de dados, é natural que sejam necessários conversores de formato de dados entre estas ferramentas. Em um cenário com N ferramentas a serem integradas entre si, existem N^2 possibilidades de trocas de dados, o que poderia resultar na necessidade da criação de N^2 conversores de dados (NAGEL et al., 2012).

Para aumentar a eficiência do trabalho colaborativo, o DLR desenvolveu o CPACS (*Common Parametric Aircraft Configuration Scheme*), um modelo padrão de dados para ser utilizado como linguagem comum na comunicação entre ferramentas usadas no projeto de aeronaves. Isto transforma o problema de se ter N^2 conversores em um problema de ordem N, ou seja, cada ferramenta deve ser capaz de ler e escrever dados apenas em um formato, o padrão CPACS (NAGEL et al., 2012).

O CPACS é implementado como um *Schema Definition* (XSD) da linguagem XML (*eXtensible Markup Language*). O arquivo XSD que contém a definição do CPACS descreve a estrutura dos arquivos XML usados pelas ferramentas para intercâmbio de dados. O CPACS contém tanto dados de produto, como dados geométricos e hierarquia de componentes, quanto dados de processo, como parâmetros para geração de malha de elementos finitos a partir da geometria (NAGEL et al., 2012).

Ferramentas como o TIXI (API para linguagens de programação que facilita o tratamento de elementos dos arquivos XML), TIGL (processamento de geometrias) e VAMPzero (criação de aeronave conceitual completa a partir de requisitos de alto nível) foram desenvolvidas para facilitar o uso do padrão CPACS (NAGEL et al., 2012). O Chamaleon estende o RCE com componentes que permitem o uso destas e de outras ferramentas de apoio (SEIDER et al., 2012).

4.4. **VIRTUAL SATELLITE**

4.4.1. Histórico

Em 2008, o DLR iniciou um programa espacial focado em desenvolvimento de satélites com o objetivo de ganhar acesso independente ao espaço para execução de experimentos científicos. Este programa contou com a criação do CEF (*Concurrent Engineering Facility*), um espaço de trabalho onde engenheiros de diversas especializações, equipados com poderosas estações

de trabalho, podem melhor interagir e conduzir sessões de engenharia simultânea. Estas sessões ocorrem nas primeiras semanas de um ciclo de desenvolvimento e permitem aos engenheiros avaliar variáveis básicas de projeto de um satélite, assim como o equilíbrio entre elas. Isto é facilitado pelo fato de engenheiros especializados em diversas disciplinas compartilharem um espaço de trabalho colaborativo e conseguirem se comunicar de maneira rápida e efetiva (SEIDER et al., 2012).

Para a criação do CEF, o DLR usou como modelo o CDF (*Concurrent Design Facility*) utilizado pela Agência Espacial Europeia (ESA), com o mesmo propósito. Para conduzir estudos de viabilidade de missões espaciais seguindo a abordagem da engenharia simultânea, a ESA criou o IDM (*Integrated Design Model*) (SCHUMANN et al., 2010).

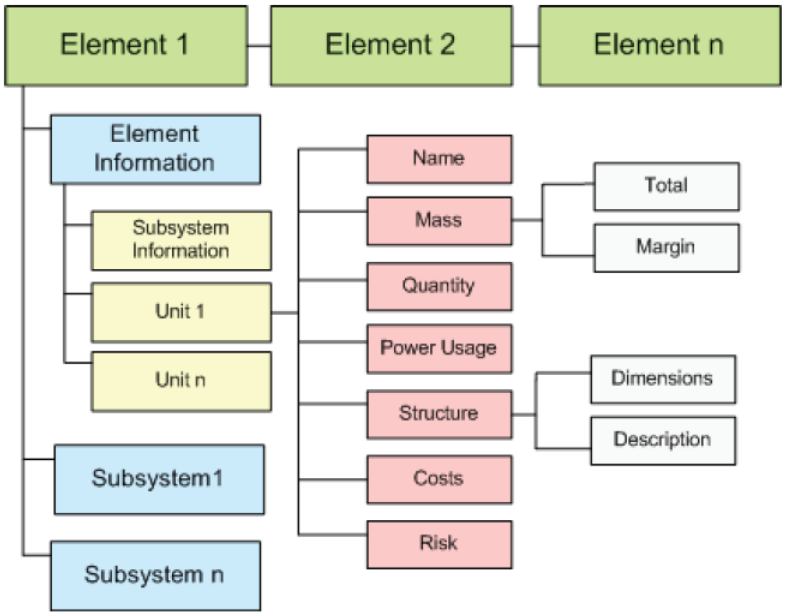
O IDM é um modelo para representação de sistemas espaciais baseado na aplicação Microsoft® Excel que segue uma estrutura hierárquica onde cada estudo é representado por *Elementos*, como um veículo de transporte e um módulo de pouso. Cada elemento é associado a um componente *Informação do Elemento* e composto de diferentes *Subsistemas,* como o de telecomunicações. Cada subsistema é subdividido em *Unidades* (como as de transmissão e recebimento do subsistema de telecomunicações) e cada *unidade* é descrita por diferentes *Parâmetros* (como consumo de potência ou peso, por exemplo) (SCHUMANN et al., 2010). A Figura 4.15 mostra um exemplo desta hierarquia.

Segundo Schumann et al. (2010), o IDM é composto, na prática, de várias pastas de trabalho do Excel, cada uma representando um subsistema, uma disciplina (como cálculo de custos gerais) ou uma combinação de ambos. Cada pasta de trabalho é um arquivo Excel que contém as seguintes planilhas padronizadas:

a) ***output***: contém a lista de parâmetros que são calculados na pasta de trabalho Excel e distribuídos para outras pastas de trabalho. Cada parâmetro contém um nome, valor e unidade;

b) ***input***: contém a lista de todos os parâmetros de entrada necessários para o cálculo dos parâmetros mostrados na planilha *Output*;

Figura 4.15 - Vista hierárquica do *Integrated Design Model* (IDM) da ESA.



Fonte: Schumann et al. (2010).

c) ***calculation***: contém fórmulas que leem os valores dos parâmetros de entrada da planilha *Input* e calculam os valores dos parâmetros de saída da planilha *Output*;

d) ***presentation***: por meio dos recursos visuais do Excel, como gráficos e tabelas dinâmicas, os parâmetros calculados são apresentados de maneira eficiente para os demais engenheiros.

Uma pasta de trabalho chamada *Database Exchange* tem a função de conectar todas as demais pastas de trabalho do estudo por meio de macros Excel. Esta

pasta de trabalho coordena a troca de informações entre as planilhas de *Input* e *Output* das demais pastas de trabalho (SCHUMANN et al., 2010).

Um dos requisitos funcionais mais importantes para um ambiente computacional de apoio para a engenharia simultânea é que o ambiente permita o trabalho distribuído. Usando o IDM, cada engenheiro trabalha no arquivo Excel que representa o subsistema sob sua responsabilidade. O líder do estudo sendo realizado é responsável por executar as macros de integração na pasta de trabalho *Database Exchange* e sincronizar todas as demais planilhas (SCHUMANN et al., 2010).

Após usar o IDM por algum tempo e analisar as suas vantagens e desvantagens, o DLR optou por substituí-lo e desenvolveu a aplicação *Virtual Satellite* (VirSat), baseada no software *framework* RCE, para suportar o processo de engenharia simultânea no CEF. Segundo Deshmukh (2015), o VirSat é uma ferramenta de *Model-Based Systems Engineering* (MBSE) com um modelo de dados consistente, capaz de criar uma hierarquia de componentes como sistema, subsistema, componente, equipamento, etc. Cada componente permite que sejam adicionados parâmetros para descrever as propriedades do equipamento, como massa, tamanho e posição.

Segundo o DLR (2017b), atualmente existe uma versão pública e gratuita do VirSat, a 3.9.0, oferecida como uma ferramenta de MBSE para as fases 0 e A de projetos de desenvolvimento de sistemas espaciais. O VirSat é a ferramenta padrão para apoiar estudos de engenharia simultânea no CEF em Bremen, Alemanha, e é utilizada também em outras instituições e universidades ao redor do mundo, assim como empresas europeias da área espacial.

Para apoiar o desenvolvimento de sistemas espaciais não apenas das fases 0 e A, mas de todas as fases do ciclo de vida de um sistema espacial, o DLR está atualmente desenvolvendo a versão 4 do VirSat, que ainda não está disponível ao público externo. O DLR é receptivo a parcerias em projetos de pesquisa que corroborem este desenvolvimento (DLR, 2017b).

4.4.2. Arquitetura do sistema

Segundo Schaus et al. (2010), o VirSat foi desenvolvido como um software *framework* modular para cobrir as etapas de projeto conceitual e estudos de requisitos de missão (fases 0 e A) que cada vez mais são realizadas em ambientes de engenharia simultânea. Outro objetivo principal é prover uma plataforma de projeto para fases subsequentes, em que a simulação exerce um papel cada vez mais importante. A ideia é usar o valor agregado da simulação tão cedo quanto possível no processo de desenvolvimento, oferecendo um ambiente onde seja possível, de forma rápida, gerar simulações de um satélite.

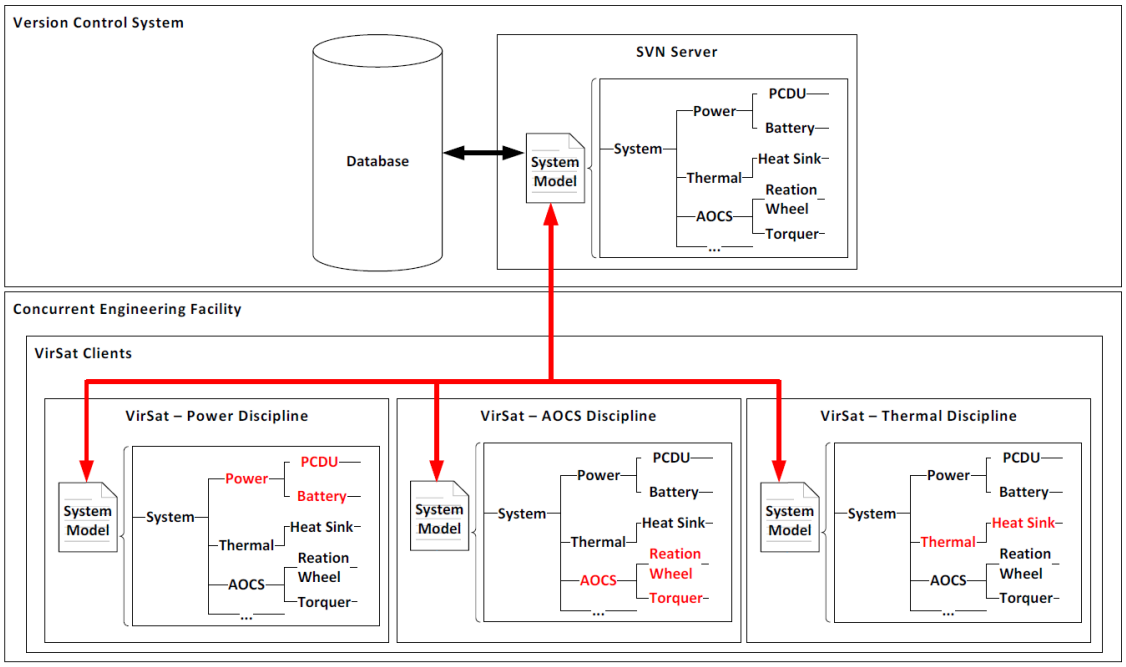
Para atender ao requisito de trabalho distribuído, cada engenheiro trabalha em uma instância local do VirSat e pode adicionar, editar, remover ou reorganizar os componentes nos níveis de sistema e subsistema da missão espacial sendo estudada. Estas modificações são realizadas na cópia local do sistema e são sincronizadas com um repositório central que utiliza o sistema de controle de versões SVN (*Subversion*) (DESHMUKH, 2015). O repositório salva o estudo em formato XMI (*XML Metadata Interchange*) e cada disciplina contribui com apenas uma parte do arquivo XMI final (SCHAUS et al., 2010). A Figura 4.16 mostra a arquitetura do VirSat e a troca de informações de modelos de sistemas dentro da CEF.

O sistema de controle de versões SVN cria uma nova revisão dos dados sempre que um novo carregamento de dados é realizado de uma instância local para o servidor central. Cada instância local é também configurável para que apenas os engenheiros responsáveis por um determinado subsistema possam modificá-lo. Isto representa um avanço em relação ao IDM, pois este modelo permitia que engenheiros pudessem erroneamente abrir uma pasta de trabalho de um subsistema que não estava sob sua responsabilidade e modificá-la (SCHUMANN et al., 2010).

O elemento central do VirSat é o seu modelo de dados que segue uma abordagem de orientação a objetos e representa o projeto do satélite com

flexibilidade e extensibilidade. O modelo de dados segue também uma abordagem hierárquica, permitindo que sejam criadas visualizações em formato de árvore. É possível também criar visualizações por disciplina, por exemplo, visualizar todos os objetos relacionados ao gerenciamento de suprimento de energia (SCHAUS et al., 2010).

Figura 4.16 - Arquitetura do *Virtual Satellite.*



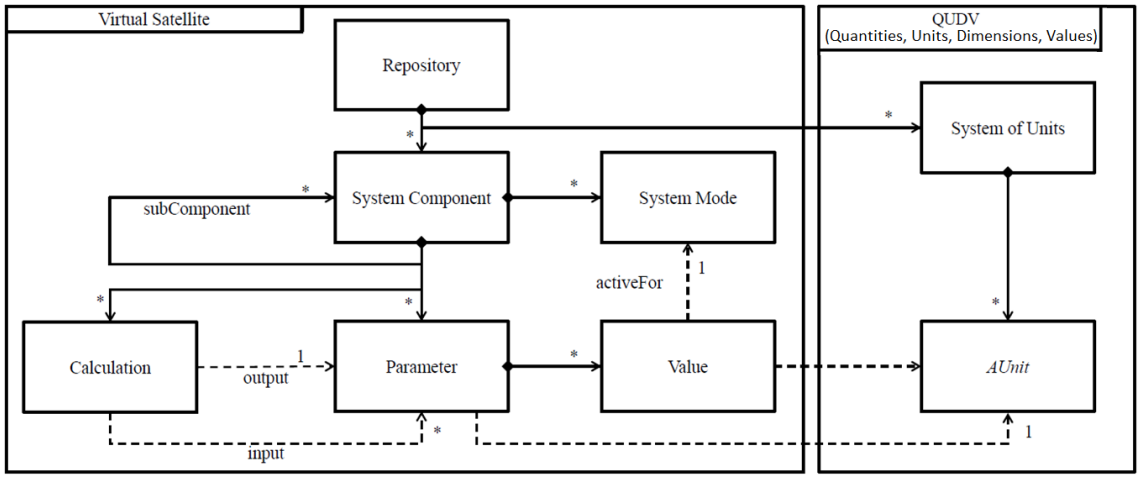
Fonte: Adaptado de Deshmukh (2015).

Segundo Schumann et al. (2008a), o modelo de dados do VirSat é compatível com o *System Engineering Information Model* (SEIM) definido no memorando técnico ECSS-E-TM-10-25A, apresentado na seção 3.6.1. Assim, os dados no VirSat estão estruturados hierarquicamente na forma de uma estrutura de componentes, subcomponentes, parâmetros e cálculos. A Figura 4.17 mostra o modelo de dados conceitual do VirSat, que pode ser interpretado como uma visão simplificada do SEIM.

A Figura 4.18 apresenta um resumo da arquitetura do modelo de dados do VirSat. De acordo com Fischer et al. (2016a), os dados do VirSat residem em um banco de dados de um sistema de controle de versão, que possui um

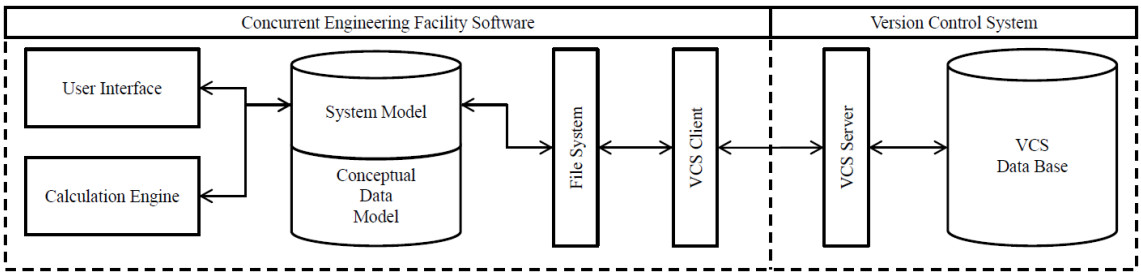
servidor central acessado pelos usuários do VirSat. Localmente, em cada instância do VirSat, é feita uma cópia local do modelo do sistema armazenado no servidor, contendo uma estrutura de dados que segue as regras do modelo de dados conceitual. Por meio da interface gráfica do VirSat, os usuários alteram os valores dos parâmetros dos componentes do sistema e um módulo de cálculos do VirSat realiza as operações matemáticas definidas entre os parâmetros, atualiza seus valores e verifica se há inconsistências. Neste caso, uma lista de possíveis problemas com valores de parâmetros é mostrada na interface gráfica do VirSat. Por exemplo, pode-se definir para um parâmetro uma faixa de valores que ele pode assumir. Se o valor deste parâmetro é calculado a partir dos valores de outros parâmetros, e o valor resultante estiver fora desta faixa, uma mensagem de erro é exibida para todos os usuários. Para que o novo valor do parâmetro esteja disponível para os demais usuários, é realizada uma publicação do mesmo no servidor do sistema de controle de versão, que adiciona o novo valor, mantendo o valor antigo como uma versão anterior.

Figura 4.17 – Modelo de dados conceitual do VirSat.



Fonte: Adaptado de Fischer et al. (2016a).

Figura 4.18 – Arquitetura do modelo de dados do VirSat.



Fonte: Fischer et al. (2016a).

Embora o VirSat tenha sido desenvolvido como uma extensão do RCE, existem algumas diferenças estruturais fundamentais, listadas a seguir:

a) **modelo de dados**: o VirSat possui um modelo dados baseado no SEIM, onde os dados e metadados são organizados em uma estrutura hierárquica bem definida de componentes, subcomponentes, parâmetros e cálculos. No RCE, existe apenas uma estrutura de projetos que contém *workflows*, e todos os dados e metadados estão ligados à execução destes *workflows*;

b) **armazenamento de dados**: o VirSat possui um repositório SVN central utilizado por todos os membros da rede VirSat. Quando um usuário modifica um parâmetro, por exemplo, ele deve publicar esta modificação no repositório e a modificação estará disponível para os demais membros da rede VirSat quando eles sincronizarem suas bases de dados locais com o repositório. No RCE, cada instância possui o seu banco de dados local, utilizado para registrar todos os dados e metadados dos *workflows* executados na instância. Entretanto, os membros da rede RCE possuem acesso aos dados de todas as instâncias;

c) **execução de simulações**: como o VirSat é baseado no SEIM, a única forma de cálculo definida é a execução de operações matemáticas envolvendo parâmetros do sistema, por meio de expressões matemáticas simples ou integração com planilhas Excel.

Isto foi mostrado na Figura 3.17, onde um *CDParameter* está associado a um *ParameterCalculation*. No RCE, simulações são executadas por meio de ferramentas compartilhadas na rede RCE e integradas em *workflows* colaborativos;

d) **controle de acesso**: o VirSat possui um sistema de gerenciamento de papéis (*Role Management*) onde podem ser definidas disciplinas e associá-las a *logins* de usuário que serão responsáveis pela edição de todos os parâmetros associados a elas. Quando um novo componente é criado na estrutura de dados do sistema, pode-se associá-lo a uma destas disciplinas e apenas o usuário cadastrado como responsável por esta disciplina pode editar os dados do componente. Demais usuários possuem somente acesso de leitura. No RCE, pode-se restringir o acesso a determinados componentes e *workflows* por meio da configuração de uma conexão segura com identificação do usuário por *login* de rede e senha;

e) **arquitetura de rede:** o VirSat adota uma arquitetura de rede clássica cliente-servidor, com um repositório de dados central compartilhado por todos os usuários, enquanto que o RCE adota uma arquitetura ponto-a-ponto, conforme visto na seção 4.2.5.

4.4.3. Interface gráfica

A interface gráfica do VirSat é composta de diversas visualizações que podem ser (re)arranjadas pelo usuário, tornando-se visíveis ou não, de acordo com sua conveniência. O VirSat possui atalhos que configuram um conjunto de vistas padronizadas. O atalho mais usado é o “PhaseA” mostrado na Figura 4.19, cujas visualizações são descritas a seguir:

a) ***Study Navigator***: situado no quadrante superior esquerdo, mostra os objetos definidos no modelo conceitual de dados do VirSat e sua

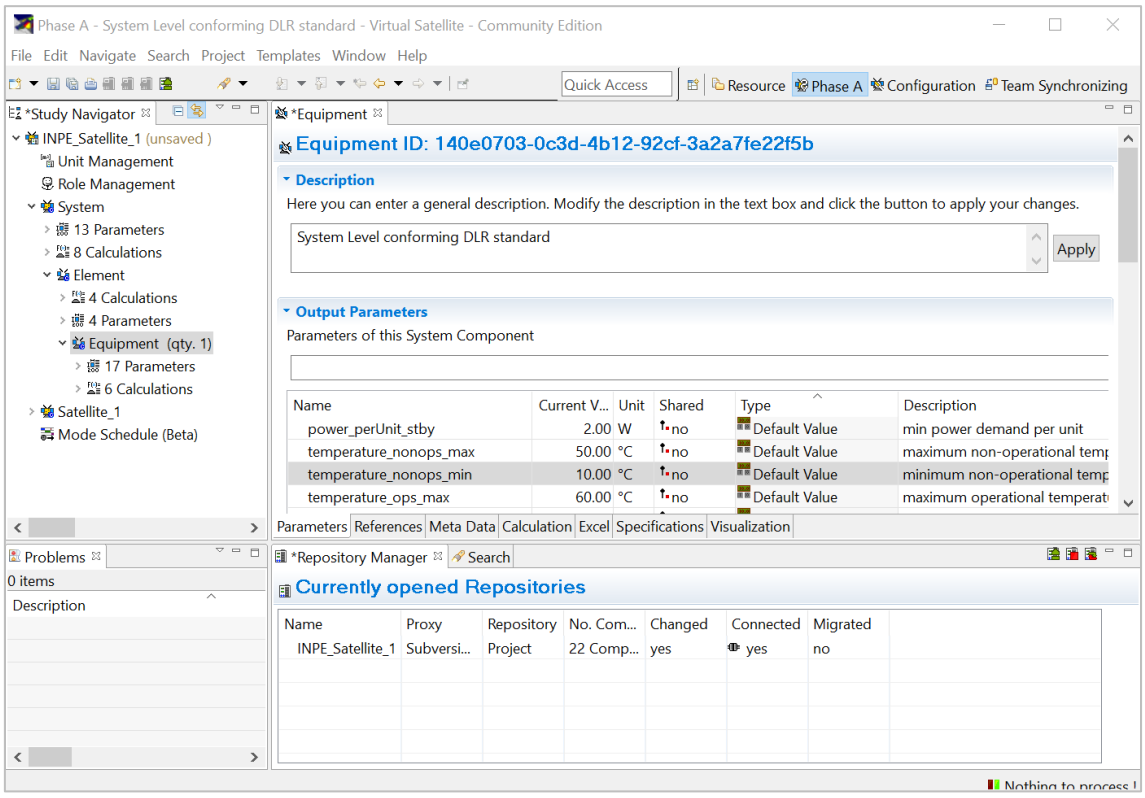
relação hierárquica. É possível ver os *Repositories, System Components, Subcomponents, Parameters* e *Calculations*;

b) ***Component View***: situado no quadrante superior direito, mostra a visão detalhada dos *System Components* e *Subcomponents* de um estudo. Esta visualização possui várias abas na parte inferior que mostram mais detalhes dos componentes;

c) ***Problems***: situado no quadrante inferior esquerdo, mostra a lista de inconsistências encontrada pelo módulo de cálculos de valores de parâmetros do VirSat, que deve ser resolvida pelo usuário;

d) ***Repository Manager***: situado no quadrante inferior direito, mostra a lista de repositórios abertos no VirSat, possibilitando abrir novos repositórios, fechá-los ou reiniciar uma conexão.

Figura 4.19 – Interface gráfica do VirSat com o conjunto de visualizações *PhaseA.*

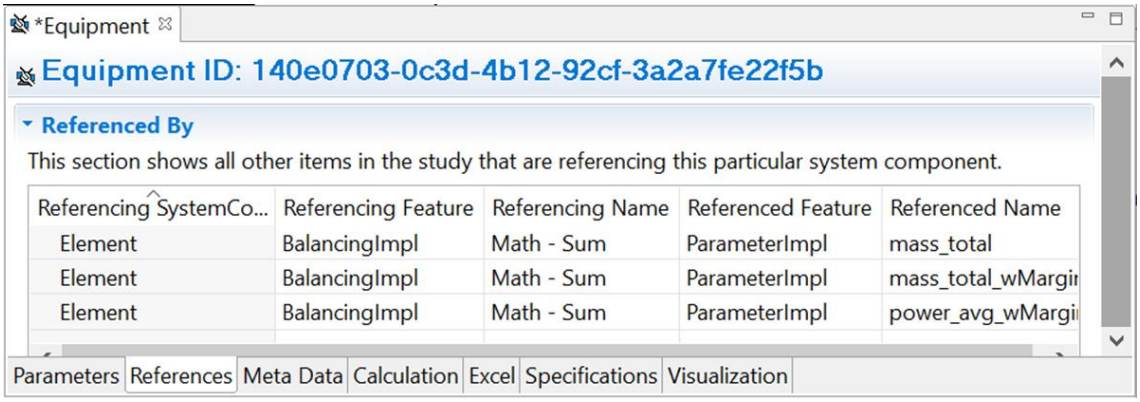


Fonte: Produção do autor.

A aba *Parameters*, mostrada na visualização *Component View* descrita acima, contém a descrição do componente, assim como seus parâmetros de saída e a definição de cálculos entre parâmetros, modos e disciplina responsável pelo componente.

A aba *References*, mostrada na Figura 4.20, mostra uma lista com todos os componentes do sistema cujos parâmetros fazem algum tipo de referência aos parâmetros do componente em questão. Assim, é possível rastrear os possíveis impactos que a mudança do valor de um parâmetro de um componente pode ter nos parâmetros dos demais componentes afetados.

Figura 4.20 – Aba *References* na visualização de componentes do VirSat.



Fonte: Produção do autor.

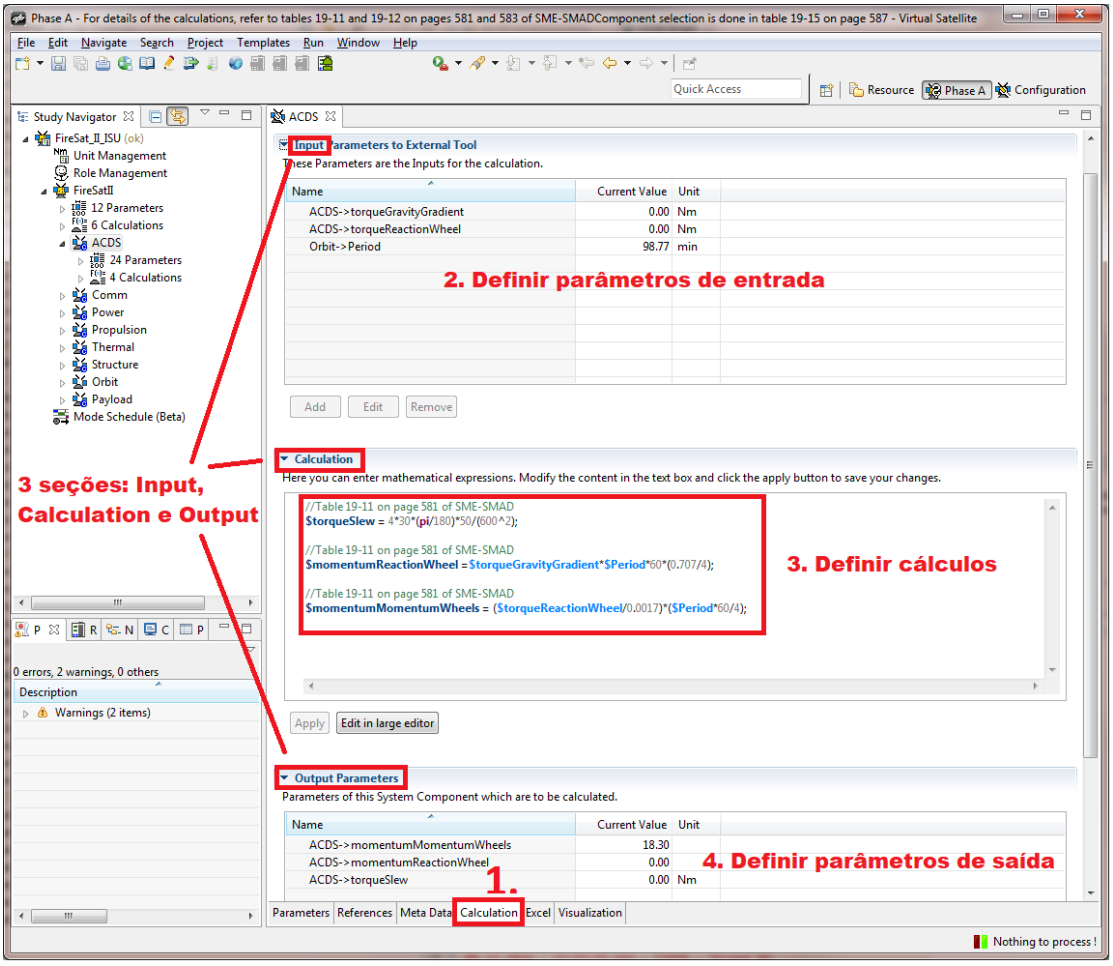
A aba *Calculation* mostrada na Figura 4.21 mostra uma das três formas de cálculo de parâmetros no VirSat. Nesta aba, podem-se definir parâmetros de entrada, escrever equações utilizando os nomes destes parâmetros e associar os resultados destas equações a valores de parâmetros de saída.

Nas primeiras versões do VirSat, a aba *Calculations* permitia apenas cálculos pré-definidos, o que era uma limitação em comparação à liberdade para a construção de fórmulas complexas com que os engenheiros estavam acostumados no IDM, baseado no Excel. Nas versões mais recentes, o VirSat incorporou uma integração com o Excel, de forma que, para cada componente do sistema, possa ser anexado um arquivo Excel. Parâmetros de entrada definidos no modelo de dados do VirSat podem ser lidos pelo Excel, que realiza

os cálculos necessários e realimenta parâmetros de saída com os resultados dos cálculos.

Assim, os engenheiros podem continuar a trabalhar com um paradigma semelhante ao IDM, com o qual já estavam acostumados, sem perder o benefício de um modelo de dados distribuído e com controle de acesso (SEIDER et al., 2012). Esta integração com o Excel é realizada em uma aba com o mesmo nome, mostrada na Figura 4.22.

Figura 4.21 – Aba *Calculation* na visualização de componentes do VirSat.

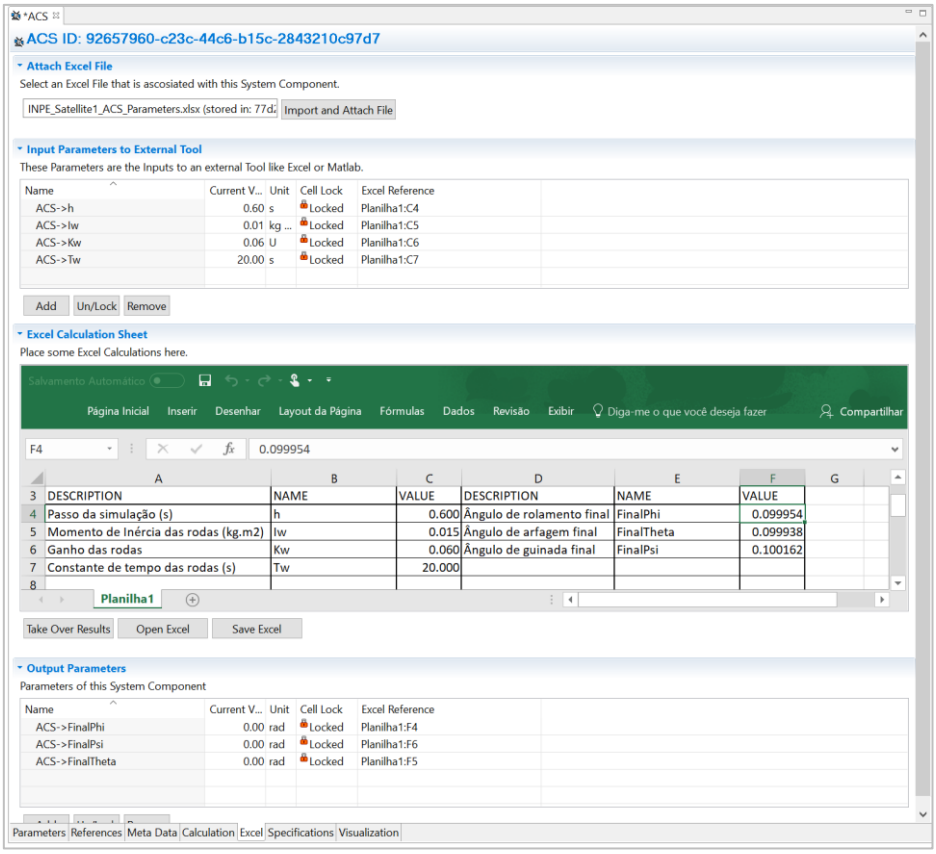


Fonte: Adaptado de DLR (2017b).

Além destes métodos de definir valores de parâmetros por meio de expressões matemáticas e planilhas Excel, o VirSat possui um método padrão que é a utilização do objeto *Calculation*, previsto no modelo de dados mostrado na

Figura 4.17. Conforme mostrado na visualização do *Study Navigator*, os objetos do tipo *Calculation* são mostrados hierarquicamente abaixo dos componentes do sistema, assim como os objetos do tipo *Parameter*. Entretanto, a interface para a definição de expressões matemáticas por meio do objeto *Calculation* é bem simplificada e permite apenas a definição de expressões elementares, como mostrado na Figura 4.23.

Figura 4.22 – Aba Excel na visualização de componentes do VirSat.



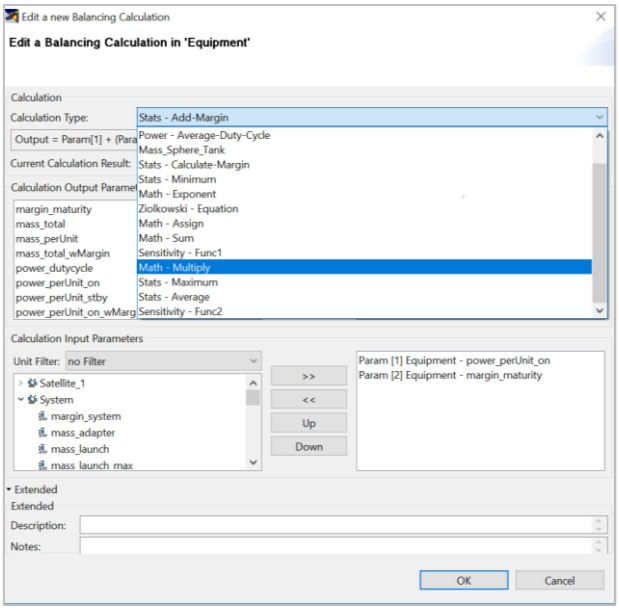
Fonte: Produção do autor.

Segundo SEIDER et al. (2012), o DLR pretendia reusar o componente de motor de fluxo do RCE no VirSat, o que possibilitaria a integração de componentes como *scripts* Python e macros Excel para conectar várias disciplinas em um *workflow* colaborativo. Entretanto, a versão 3.9.0 atualmente disponível (DLR, 2017b) não conta com este recurso.

4.4.4. Configuração do sistema

O VirSat, assim como o RCE, não possui programa de instalação. O software é fornecido em um arquivo ZIP contendo o executável do software e arquivos de apoio, que devem ser descompactados em uma pasta local do computador do usuário. Diferentemente do RCE, o código-fonte do VirSat não está disponível publicamente. Entretanto, é possível baixar gratuitamente a versão 3.9.0 da internet (DLR, 2017b) e utilizá-la.

Figura 4.23 – Janela de edição de Cálculos entre parâmetros do VirSat.



Fonte: Produção do autor.

Ao se definir um determinado estudo que será realizado utilizando o VirSat, deve ser criado para este estudo um arquivo de propriedades na mesma pasta que contém o executável do VirSat, e nele são registrados o nome do estudo (“INPE_Satellite_1”, no exemplo a seguir) e o nome e o endereço do repositório SVN (criado previamente) que será utilizado, assim como as credenciais do usuário para acesso ao repositório. Um exemplo de arquivo de configuração do VirSat é mostrado na Figura 4.24.

Figura 4.24 – Exemplo de arquivo de configuração do VirSat.

```
virsat.persistence.repository.proxy =
de.dlr.virsat.datastore.proxy.user.SVNResourceRepositoryProxy

# the name of the study in the SVN repository. Must not be empty
virsat.persistence.repository.name = INPE_Satellite_1

# Path to the SVN repository, can be left blank for a local study without
collaboration with other users
virsat.persistence.repository.remote= https://DESKTOP-126D28U/svn/VirtualSatellite

# put your credentials here or leave blank, and VirSat will ask for them
virsat.persistence.repository.user=duran
virsat.persistence.repository.pass=duran
```

Fonte: Produção do autor.

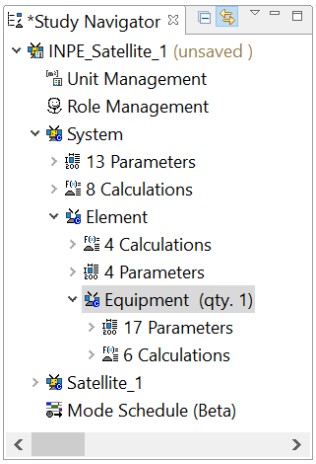
O repositório SVN deve ser criado em uma máquina que atuará como servidor do sistema de controle de versão, como mostrado do lado direito da Figura 4.18. O administrador do sistema deve criar contas de acesso ao repositório para todos os usuários do VirSat. Na máquina local do usuário do VirSat é criada a pasta "*virsatws*" dentro do diretório onde está localizado o executável do VirSat, contento uma cópia de todos os dados do estudo publicados no servidor. A atualização dos dados é garantida por meio de sincronizações periódicas com o servidor, acordadas entre os participantes do estudo.

O VirSat é então iniciado por meio de um *script* que define alguns argumentos para o executável. Um destes argumentos é o "-disciplineAdmin", que inicia o VirSat em modo administrativo, o que é necessário para realizar as configurações iniciais. O arquivo de propriedades é então carregado no VirSat, que estabelece uma conexão com o servidor do repositório SVN.

A seguir, como mostrado na Figura 4.25, a visualização *Study Navigator* apresenta o estudo definido no arquivo de propriedades (INPE_Satellite_1) no topo, com elementos de *Unit Management* (usado para gerenciar as unidades de medida utilizadas no estudo) e *Role Management* (usado para definir as disciplinas usadas no estudo), logo abaixo. Usando a opção "*Create New Study Template*" do menu *Templates*, é criada uma estrutura padrão de componentes, contendo o componente de sistema principal, chamado *System* e os subcomponentes *Element* e *Equipment*. Tanto o componente principal

quanto os subcomponentes possuem os objetos dos tipos *Parameter* e *Calculation*, que contêm metadados a respeito dos componentes.

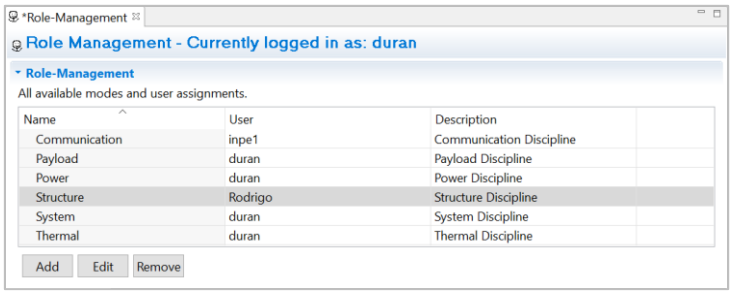
Figura 4.25 – Visualização do *Study Navigator* no VirSat.



Fonte: Produção do autor.

Esta estrutura padrão pode ser editada com a adição de mais componentes, subcomponentes, parâmetros e cálculos, ou pode ser criada uma nova estrutura manualmente. Em seguida, devem ser configuradas as permissões de edição dos dados de cada componente. Na visualização do Role Management, mostrada na Figura 4.26, são definidos os nomes das disciplinas participantes do estudo (*Name*) e os *logins* de usuários (*User*) que serão os administradores dos dados de cada disciplina, assim como uma descrição (*Description*) de cada uma. Os *logins* de usuário são os mesmos *logins* de rede utilizados ao realizar o *login* no sistema operacional do computador.

Figura 4.26 – Visualização do *Role Management* no VirSat.

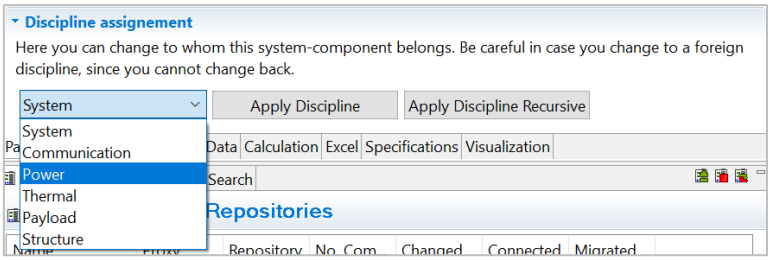


Fonte: Produção do autor.

Após a criação das disciplinas, deve-se associá-las aos componentes existentes na estrutura de dados. Enquanto nenhuma disciplina é associada a eles, qualquer usuário pode editá-los. Para realizar a associação de uma disciplina a um componente, deve-se clicar na aba *Parameters* localizada no *Component View*, acessar a parte inferior da interface (mostrada na Figura 4.27), escolher a disciplina definida na etapa anterior no primeiro *pull-down menu* e pressionar o botão *Apply Discipline* para realizar a associação. Caso o componente possua subcomponentes, o botão "*Apply Discipline Recursive*" aplica a mesma associação a todos os subcomponentes.

Com a realização das configurações mostradas nesta seção, o VirSat está pronto para a execução de uma sessão de engenharia simultânea no contexto da fase A de um projeto de desenvolvimento de um sistema espacial.

Figura 4.27 – Menu de associação de disciplinas a componentes no VirSat.



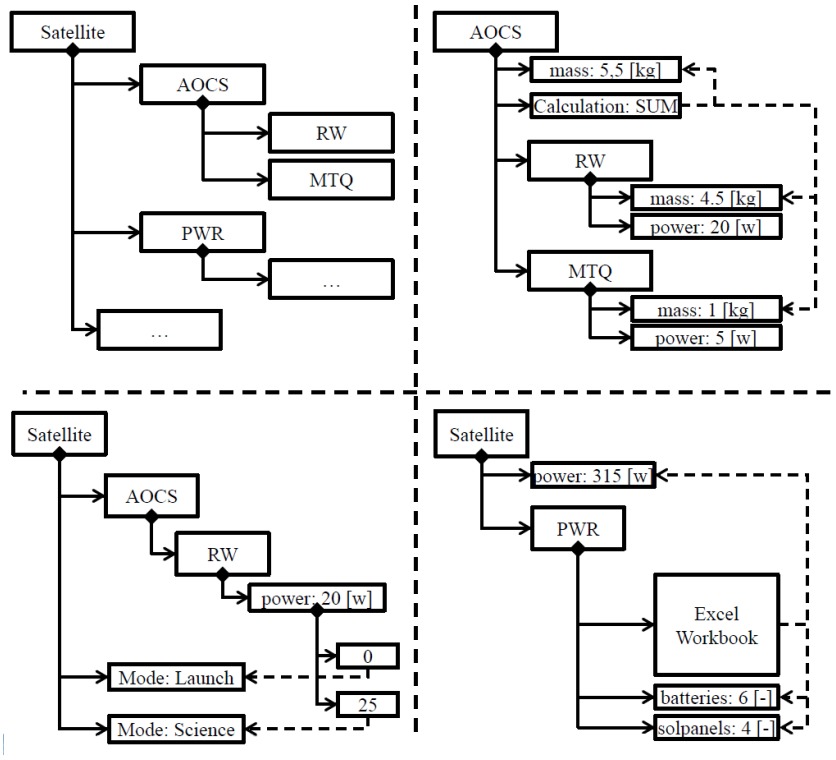
Fonte: Produção do autor.

4.4.5. Casos de uso

A Figura 4.28 mostra exemplos de informações coletadas durante estudos realizados em sessões de engenharia simultânea com o VirSat para alguns casos de uso. O sistema principal (*Satellite*) é decomposto em subsistemas (como o AOCS e PWR mostrados na figura) que por sua vez também são decompostos em subcomponentes. A massa do subsistema AOCS é calculada por meio de uma *Calculation*, que atribui ao parâmetro *mass* do subsistema o valor da soma das massas de seus subcomponentes. Podem ser também definidos modos de operação do sistema e o consumo de potência do satélite para cada um destes modos, além do consumo global. Cálculos mais

complexos podem ser introduzidos no modelo por meio da associação com uma planilha Excel (FISCHER et al., 2016a).

Figura 4.28 – Exemplos de casos de uso de modelamento de sistemas.



Fonte: Fischer et al. (2016a).

4.5. **REQUISITOS DO AMBIENTE DE GERENCIAMENTO DE DADOS E PROCESSOS DE SIMULAÇÃO A SER USADO NO INPE**

Após a revisão da literatura sobre as funcionalidades e potenciais aplicações do RCE e do VirSat, foram propostos alguns requisitos que o ambiente de gerenciamento de dados e processos de simulação a ser proposto neste trabalho, baseado no software *framework* RCE, deveria cumprir. Estes requisitos são:

a) O ambiente deve permitir a distribuição de aplicações de software geradas por um usuário para os demais usuários;

b) O ambiente deve permitir o reuso de aplicações de software geradas por um usuário pelos demais usuários;

c) O ambiente deve permitir a construção de *workflows* de simulação que contenham o encadeamento de aplicações de software desenvolvidas por seus usuários;

d) O ambiente deve permitir o armazenamento de dados gerados em *workflows* de simulação em repositórios de dados acessíveis a todos os seus usuários;

e) O ambiente deve permitir que um usuário consulte dados de *workflows* de simulação executadas por qualquer usuário;

f) O ambiente deve permitir o modelamento da hierarquia de subsistemas e componentes de um sistema;

g) O ambiente deve permitir o modelamento de parâmetros de sistemas, subsistemas e componentes;

h) O ambiente deve permitir que dados de parâmetros de sistemas, subsistemas e componentes sejam usados como entradas de *workflows* de simulação;

i) O ambiente deve permitir que dados gerados por *workflows* de simulação sejam utilizados para atualizar valores de parâmetros de sistemas, subsistemas e componentes;

j) O ambiente deve permitir que apenas usuários autorizados sejam capazes de executar modificações em dados de sistemas armazenados no ambiente;

k) O ambiente deve permitir o registro da evolução dos valores de parâmetros de sistemas, subsistemas e componentes;

l) O ambiente deve permitir o registro da evolução dos dados de *workflows* de simulação utilizados para atualizar os valores de parâmetros de sistemas, subsistemas e componentes;

m) O ambiente deve permitir o arquivamento de *workflows* de simulação usados para a atualização de valores de parâmetros de sistemas, subsistemas e componentes;

n) O ambiente deve permitir que *workflows* de simulação arquivados sejam reexecutados para reavaliar valores de parâmetros de sistemas, subsistemas e componentes.

5 PROPOSTA DO AMBIENTE DE GERENCIAMENTO DE DADOS E PROCESSOS DE SIMULAÇÃO PARA O INPE

Neste capítulo é mostrada a arquitetura do ambiente de gerenciamento de dados e processos de simulação proposto para ser utilizado no INPE durante as fases 0 e A de desenvolvimento de sistemas espaciais. É mostrada ainda a motivação que orientou a especificação da proposta, a modificação no modelo conceitual de dados necessária para substanciá-la e as configurações das aplicações requeridas para tornar esta proposta uma realidade.

5.1. **Introdução**

Ao estudar o processo de *Engenharia de Sistemas* aplicado ao desenvolvimento de sistema espaciais na seção 2.2, verifica-se que a utilização de M&S nas fases iniciais do desenvolvimento de produtos traz uma série de vantagens, dentre elas o aumento de maturidade do produto e a redução de custos de desenvolvimento.

Na parte esquerda do modelo “V” de *Engenharia de Sistemas*, no intervalo entre duas linhas de base, são realizadas interações acima e abaixo, fora do núcleo do V. Nas iterações abaixo, a equipe de desenvolvimento investiga opções de solução de sistema, testando novos conceitos em níveis mais baixos de detalhe e balanceando riscos, visando minimizá-los, para que o desempenho esperado pelos interessados seja atendido. Estas investigações são realizadas, na maioria das vezes, por meio de M&S, sendo criados muitos modelos de sistemas e executadas muitas simulações. Entretanto, nas fases iniciais (0 e A), os estudos geralmente estão limitados a cálculos simplificados utilizando-se valores de parâmetros dos sistemas. Uma das propostas deste trabalho é que os estudos realizados nas fases 0 e A incluam atividades de M&S, aumentando-se assim a precisão dos resultados obtidos. Para viabilizar estas atividades, propõe-se a utilização de um ambiente de gerenciamento de dados e processos de simulação que é descrito em detalhes neste capítulo.

Na seção 4.4.2 foram mostradas as diferenças estruturais entre VirSat e RCE, embora aquele tenha sido derivado deste. Dados os requisitos que o ambiente de gerenciamento de dados e processos de simulação deve cumprir, mostrados na seção 4.5, nota-se que existem ganhos emergentes ao se integrar o RCE e o VirSat para a concepção da proposta do ambiente deste trabalho. Em linhas gerais, este ambiente é composto da integração do VirSat com o RCE, por meio da utilização de um modelo de dados conceitual modificado do VirSat, também proposto neste trabalho, como base para o ambiente. Assim, combinam-se os pontos fortes das duas aplicações: a capacidade de gerenciamento de dados e processos de simulação do RCE com um modelo hierárquico de dados e um repositório de dados central do VirSat. A identificação das sinergias entre RCE e VirSat, a configuração da integração entre eles, assim como os procedimentos criados para executá-la, são as principais contribuições deste trabalho, sendo mostradas ao longo deste capítulo.

Para o gerenciamento de dados e processos de simulação além das fases 0 e A, incluindo todas as fases do ciclo de vida dos sistemas espaciais, é necessário um modelo de dados mais abrangente e descritivo e o desenvolvimento de uma nova aplicação de software compatível com este modelo. Segundo Fischer et al. (2016b), o DLR está atualmente trabalhando no desenvolvimento da versão 4 do VirSat que se propõe a ser uma plataforma de *Model Based Systems Engineering* para todo o ciclo de vida de um sistema espacial. De acordo com o DLR (2017b), há interesse do DLR em firmar parcerias com outros institutos de pesquisa para fortalecer as pesquisas nesta área.

5.2. **Modelo de dados conceitual**

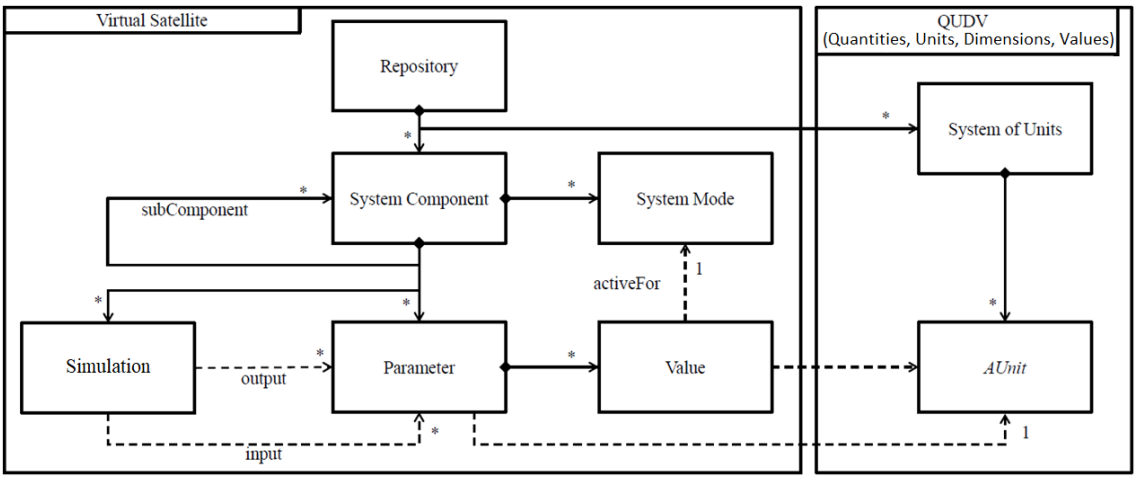
Conforme mostrado na seção 3.5.2, de acordo com o memorando técnico ECSS-E-TM-10-23A existem diversas aplicações de software utilizadas no desenvolvimento de sistemas espaciais, cada uma com suas particularidades

no que diz respeito ao repositório de dados utilizado. Seria inviável exigir que todas as aplicações compartilhassem de um mesmo repositório e formato de dados, pois todas as aplicações de software deveriam ser reescritas para isso, o que aumentaria muito os custos do projeto. A recomendação da ECSS é que a padronização dos dados a serem compartilhados se dê no nível semântico, independentemente de qualquer tecnologia de implantação de software. Isto é alcançado por meio da definição de um modelo de dados conceitual que seja utilizado nativamente por todas as aplicações de software, ou senão que as mesmas sejam estendidas com a criação de adaptadores de interface compatíveis.

O modelo de dados conceitual especifica a semântica dos dados relativo a um domínio, incluindo dados significativos para usuários finais, suas características e associações entre eles. A adaptação do SEIM para o software VirSat, mostrado Figura 4.17, é um exemplo de modelo de dados conceitual aplicado às fases 0 e A de projetos espaciais. No contexto deste trabalho, propõe-se a utilização de uma versão modificada deste modelo de dados conceitual, com a substituição do elemento *Calculation* pelo elemento *Simulation*, como descrito a seguir.

O elemento *Calculation* do modelo de dados conceitual do VirSat é utilizado para o modelamento de expressões matemáticas utilizadas no cálculo de valor de um parâmetro, utilizando como entrada valores de outros parâmetros. Como se propõe a utilização não apenas de cálculos simples, mas também de simulações para o cálculo de valores de parâmetros do sistema, é proposta uma modificação deste modelo de dados conceitual, ampliando o escopo do elemento *Calculation*, renomeando-o para *Simulation*, como mostrado na Figura 5.1, e permitindo que a simulação tanto receba quanto escreva valores de múltiplos parâmetros. Como o VirSat não é capaz de executar simulações, isto é realizado pelo RCE. Estas duas aplicações integradas compõem a base do ambiente, como mostrado na próxima seção.

Figura 5.1 – Modelo de dados conceitual da proposta.



Fonte: Adaptado de Fischer et al. (2016a).

5.3. **Repositório de dados de sistemas espaciais**

Na seção 3.5.2, o memorando técnico ECSS-E-TM-10-23A recomenda que um repositório de dados de sistemas espaciais seja formado pelo agrupamento de vários repositórios de dados, que trocam dados entre si, seguindo um mesmo modelo de dados conceitual. Este conceito foi ilustrado na Figura 3.8. O repositório de dados de sistemas espaciais proposto neste trabalho, mostrado na Figura 5.2, consiste da integração do banco de dados do RCE com o servidor SVN utilizado pelo VirSat, com interoperabilidade semântica viabilizada pela adoção do modelo de dados conceitual mostrado na Figura 5.1.

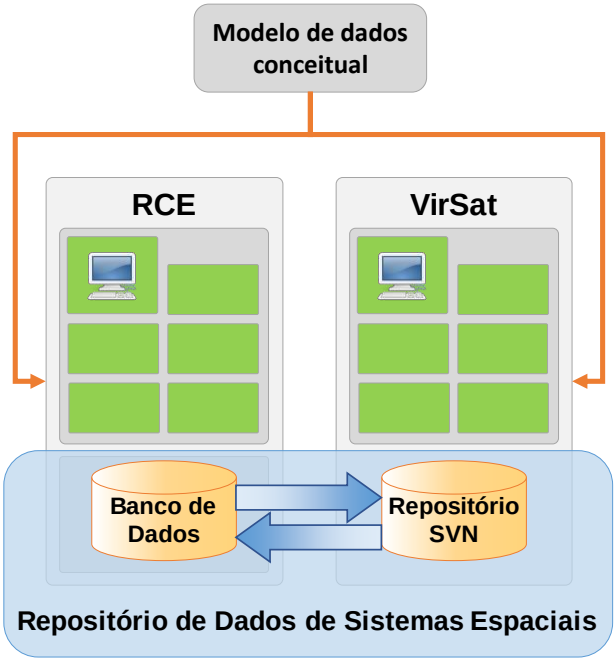
Em uma simulação de sistemas, o VirSat fornece os dados de valores de parâmetros de entrada para o RCE, que os adapta para o formato necessário para o software de simulação utilizado. Ao término da simulação, o RCE retorna para o VirSat os dados dos valores dos parâmetros de saída do sistema.

Conforme discutido na seção 2.3, embora vários estudos sejam realizados durante o processo de *Engenharia de Sistemas*, somente linhas de base aprovadas e incorporadas ao núcleo do V são colocadas em controle de configuração ao final de cada revisão de projeto. Estudos, análises, modelos e

demais artefatos não diretamente associados à linha de base aprovada não são formalmente controlados. Assim, neste repositório proposto, dados oficiais que irão substanciar decisões de projeto durante uma troca de fase são armazenados no repositório SVN do VirSat. Dados temporários, intermediários e de simulações utilizados em estudos que não resultaram em decisões de projeto ficam armazenados no banco de dados do RCE.

Propõe-se que os dados oficiais armazenados no repositório SVN do VirSat sejam não apenas os valores dos parâmetros dos sistemas, mas também, para cada componente do sistema, um arquivo ZIP contendo o projeto do RCE, incluindo o *workflow* e os dados que deram origem aos parâmetros.

Figura 5.2 – Repositório de dados de sistemas espaciais da proposta.



Fonte: Produção do autor.

5.4. **Arquitetura do ambiente**

O ambiente proposto neste trabalho tem como objetivo o gerenciamento de dados e processos de simulação durante sessões de engenharia simultânea nas fases 0 e A do projeto de sistemas espaciais. Conforme visto na seção 2.5.1, a ESA criou o conceito do CDF para facilitar a interação dos engenheiros

em uma equipe multidisciplinar nestas sessões de engenharia simultânea. A Figura 2.9 mostra o *layout* dos computadores do CEF do DLR. Segundo Chagas (2016), o INPE também desenvolveu uma infraestrutura similar, denominada “Centro de Projeto Integrado de Missões Espaciais” (CPRIME). O ambiente proposto neste trabalho tem o potencial de ser avaliado para utilização neste centro.

Para exemplificar a arquitetura do ambiente proposto, foi concebido um exemplo com quatro computadores, cujas funções são apresentadas na Tabela 5.1. Cada computador possui uma instalação do RCE e do VirSat, com configurações distintas (exceto o computador 3 que possui apenas a instalação do RCE), que lhe conferem funções específicas.

Tabela 5.1 – Função de cada computador no exemplo de arquitetura.

Computador	Configuração RCE	Configuração VirSat
1	Servidor *Relay*	Servidor SVN
2	Máquina Cliente	Usuário
3	Nó de Computação	-
4	Máquina Cliente	Usuário

Fonte: Produção do autor.

O computador 1 tem a função de servidor *relay* para a rede RCE. Todos os demais computadores desta rede possuem, em seu arquivo de configuração, o endereço de rede deste servidor. Este computador é também o servidor do sistema de controle de versão SVN utilizado pelo VirSat. Este computador não é utilizado pelos usuários para acessar o ambiente.

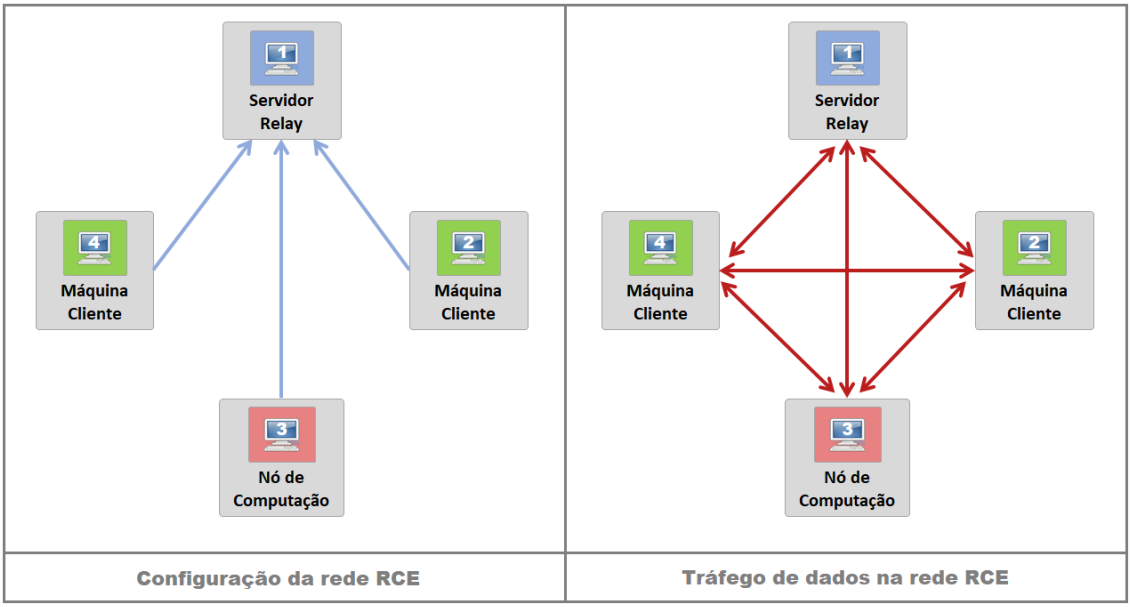
Os computadores 2 e 4 possuem configurações semelhantes. Cada um possui uma instalação padrão de RCE e de VirSat e seu objetivo é ser a interface do usuário com o ambiente. Engenheiros participando de sessões de engenharia simultânea fariam *login* apenas nestes dois computadores para interagirem com o sistema.

O computador 3 é um nó de computação do RCE, o que significa que contém as versões mais estáveis das ferramentas de software distribuídas na rede

RCE e permanece sempre ligado para que as aplicações estejam sempre disponíveis. Neste computador, o RCE é iniciado como serviço, o que significa que, mesmo havendo uma queda de energia, quando o computador for religado, o RCE é iniciado automaticamente. Os computadores 2 e 4, sendo usados como interface dos usuários com o ambiente, podem estar ocasionalmente desligados. Caso uma ferramenta de software seja disponibilizada nestes computadores e eles estejam desligados, elas não estarão disponíveis para os demais usuários. Assim, é recomendável que ferramentas de software desenvolvidas pelos usuários nos computadores 2 e 4 sejam posteriormente disponibilizadas apenas no computador 3, para garantir a disponibilidade das mesmas aos demais usuários.

A Figura 5.3 ilustra a configuração da rede RCE (lado esquerdo), mostrando que os computadores 2, 3 e 4 conectam-se ao computador 1 para ingressarem na rede RCE. Uma vez dentro desta rede, cada computador pode acessar diretamente qualquer outro computador pertencente à mesma rede, o que é mostrado no lado direito da Figura 5.3.

Figura 5.3 – Conexões de rede entre os computadores na rede RCE no ambiente.



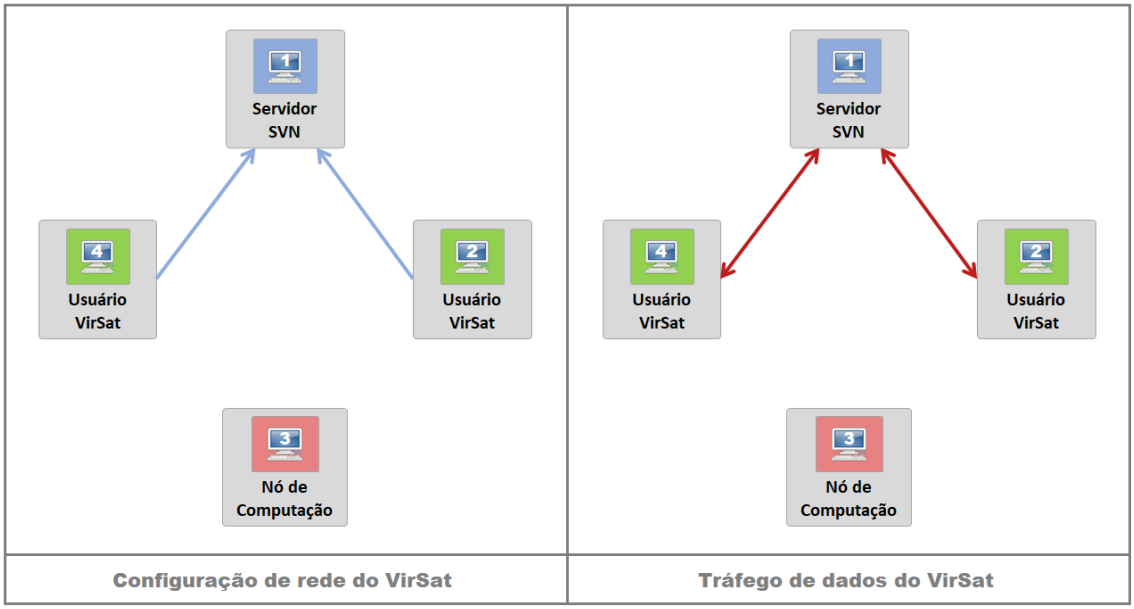
Fonte: Produção do autor.

A Figura 5.4 ilustra a configuração de rede do VirSat (lado esquerdo), mostrando que as instalações do VirSat nos computadores 2 e 4 estão configuradas para acessar o servidor SVN do computador 1. Uma vez conectados ao servidor, estes computadores podem tanto descarregar dados do servidor, criando uma cópia local destes dados, quanto carregá-lo com novos dados, o que é mostrado no lado direito da Figura 5.4. Este processo permite, por exemplo, que um novo parâmetro carregado no servidor pelo usuário do computador 2 seja descarregado no computador 4 por outro usuário.

Na prática, o ambiente proposto é uma sobreposição da Figura 5.3 e Figura 5.4. Assim, por exemplo, um usuário pode utilizar o computador 2, usando o VirSat, para configurar uma simulação de sistema cujos valores de parâmetros estão armazenados no servidor SVN (computador 1). Primeiramente, o usuário sincroniza o repositório local do VirSat com o servidor, para obter a versão mais recente dos valores dos parâmetros. A seguir, o usuário cria um *workflow* no RCE que vai utilizar ferramentas disponíveis nos computadores 2, 3 e 4. Ao executar a simulação, o RCE internamente transfere dados de entrada e saída entre os computadores 2, 3 e 4 até que o resultado final seja obtido e esteja disponível no computador 2.

Neste processo, cada ferramenta é executada no respectivo computador onde foi disponibilizada, não há transferência de ferramentas entre computadores, apenas transferência de dados de entrada e saída. Os resultados da simulação são usados para atualizar valores de parâmetros de saída do sistema, que são atualizados no repositório local do VirSat no computador 2. Por fim, o usuário sincroniza o repositório do servidor SVN (computador 1) com o computador 2, permitindo assim que os demais usuários usufruam dos valores atualizados dos parâmetros do sistema.

Figura 5.4–Conexões de rede entre os computadores utilizando o VirSat.



Fonte: Produção do autor.

Conforme proposto na seção 5.3, dados oficiais, tanto os arquivos utilizados em simulações quanto os valores de parâmetros, são armazenados no repositório SVN do VirSat. Dados transitórios e de estudos que não resultaram em alterações no produto permanecem no banco de dados RCE durante o projeto e podem ser excluídos ao seu término.

O RCE possui a funcionalidade *Export* que permite a exportação de um arquivo ZIP contendo uma estrutura de pastas com todos os arquivos utilizados em um *workflow*. No futuro, caso os arquivos do *workflow* não estejam mais disponíveis no RCE, é possível importar o arquivo ZIP no RCE e reproduzir a mesma estrutura de dados, o que permite uma nova execução do *workflow* com os mesmos dados ou dados modificados. Para publicar os arquivos utilizados em simulações no repositório de dados do VirSat, foi desenvolvido um componente no RCE que copia este arquivo ZIP para o repositório local do VirSat. Ao sincronizar o repositório local com o do servidor SVN, o arquivo entra em controle de versão e torna-se disponível para os demais usuários.

No exemplo acima, a publicação do arquivo com os dados do *workflow* no repositório do VirSat tem apenas a função de documentação do processo, pois o *workflow* já se encontrava disponível na rede RCE para todos os usuários antes disso.

Como visto na Figura 5.4, embora o computador 3 não esteja diretamente conectado ao servidor SVN, nota-se que os dados gerados por ele durante a execução do *workflow* são acessíveis a todos os computadores da rede RCE. Assim, o usuário "logado" no computador 2 é capaz de exportar um arquivo ZIP com os dados utilizados na simulação distribuídos entre os computadores 2, 3 e 4 e em seguida publicá-lo no servidor SVN. Existe, assim, uma conexão indireta entre os computadores 1 e 3.

5.5. **Configuração da integração entre RCE e VirSat**

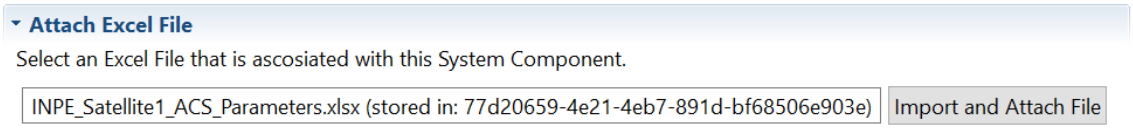
No ambiente proposto, o RCE lê os valores de parâmetros de entrada de sistemas armazenados no VirSat e depois escreve os valores calculados de parâmetros de saída no VirSat. Adicionalmente, o RCE publica um arquivo ZIP com a estrutura de pastas e arquivos utilizados no *workflow* de simulação no repositório do VirSat, para que seja possível rastrear os dados utilizados na simulação que deram origem aos parâmetros de saída.

Para atingir este objetivo, foi padronizada a pasta de instalação do VirSat em todas as máquinas como "*C:\VirSat*". Após a instalação, criação do primeiro estudo e sincronização inicial com o servidor SVN, o VirSat cria uma cópia local do repositório na pasta "*C:\VirSat\virsatws\INPE_Satellite_1\INPE_Satellite_1\svnwc*", onde "*INPE_Satellite_1*" é o nome do estudo de engenharia simultânea escolhido.

Uma das opções de cálculo de parâmetros nativa do VirSat é por meio de uma planilha Excel que é carregada para o repositório SVN na aba "Excel" do *Component View* do VirSat, mostrado anteriormente na Figura 4.22. No topo desta figura, há um campo para a importação da planilha, onde são mapeados

os parâmetros de entrada e saída do sistema. Após a importação, é exibido um código (“77d20659-4e21-4eb7-891d-bf68506e903e”) ao lado do nome do arquivo, mostrado na Figura 5.5.

Figura 5.5 – Identificação única do arquivo Excel com parâmetros do VirSat.



Fonte: Produção do autor.

Este código é uma identificação única (UUID) de objeto, conforme visto na seção 3.5.3, que permite a localização segura de um objeto no banco de dados. Neste caso, o VirSat cria, dentro da pasta “*svnwc*”, uma subpasta chamada “*fileStore*” e dentro dela cria outra pasta nomeada com o código do arquivo Excel. Dentro desta pasta, o VirSat copia o arquivo Excel com seu nome original.

Sabendo-se o caminho completo do arquivo Excel publicado no repositório SVN local que contém os valores dos parâmetros do componente do sistema, é possível criar um *script* que leia o arquivo Excel, extraia os valores dos parâmetros e os disponibilize para uso por um software de simulação, no formato adequado. Sendo variáveis deste caminho apenas o nome do estudo de engenharia simultânea e o código do arquivo Excel, basta fornecer estes dois valores de entrada para que o *script* obtenha a lista de parâmetros do componente de sistema e seus valores. Para que este *script* seja reutilizado em outros casos, é necessária também a adoção de um formato padrão de arquivo Excel, com um intervalo de células fixo para os valores de parâmetros de entrada, e outro para os parâmetros de saída.

Neste trabalho é utilizada a linguagem Python (versão 2.7.12) e a biblioteca Openpyxl, que permite a leitura e escrita de planilhas Excel, para a construção de *scripts* que leem e escrevem os parâmetros de componentes de sistema no

VirSat. Alternativamente, a troca de dados entre RCE e VirSat poderia também ser realizada por meio de arquivos XML.

Para a criação do arquivo ZIP que contém a estrutura de pastas e arquivos utilizados em um *workflow*, é utilizada a funcionalidade de *Export* de Projetos do RCE, usando o formato *Archive File*. Para a importação de um *Archive File* no RCE, basta usar a funcionalidade de *Import*. O *script* que publica o arquivo ZIP no repositório do VirSat faz uma cópia do arquivo de um computador da rede RCE para a mesma pasta onde está o arquivo Excel com os parâmetros, e adicionalmente edita o arquivo Excel, escrevendo o caminho completo do arquivo ZIP para facilitar a consulta futura.

5.6. **Procedimentos de operação do ambiente**

A lista de passos descrita a seguir descreve em detalhes a sequência de atividades que um engenheiro, em uma sessão de engenharia simultânea, executaria para analisar a variação do valor de um parâmetro de um componente do sistema em estudo:

a) no VirSat, definir os parâmetros de entrada e saída do componente do sistema que serão utilizados na simulação;

b) no VirSat, carregar a planilha Excel padrão na aba “Excel” do componente do sistema;

c) usando o VirSat, associar os valores dos parâmetros escolhidos com as células correspondentes aos parâmetros de entrada e saída na planilha Excel, e salvar a planilha;

d) anotar o código de identificação da planilha Excel (UUID) gerado pelo VirSat;

e) no RCE, criar um *workflow* de simulação, contendo no mínimo três componentes, que executam as seguintes funções:

a. executar o *script* que lê os valores dos parâmetros de entrada do *workflow* de simulação, a partir da planilha Excel, e gera arquivos no formato requerido pelo software de simulação;

b. executar o *workflow* de simulação;

c. executar o *script* que lê os valores dos resultados da simulação e e atualiza os valores dos parâmetros na planilha;

f) executar o *workflow* no RCE, informando como entradas o UUID da planilha Excel e o nome do estudo;

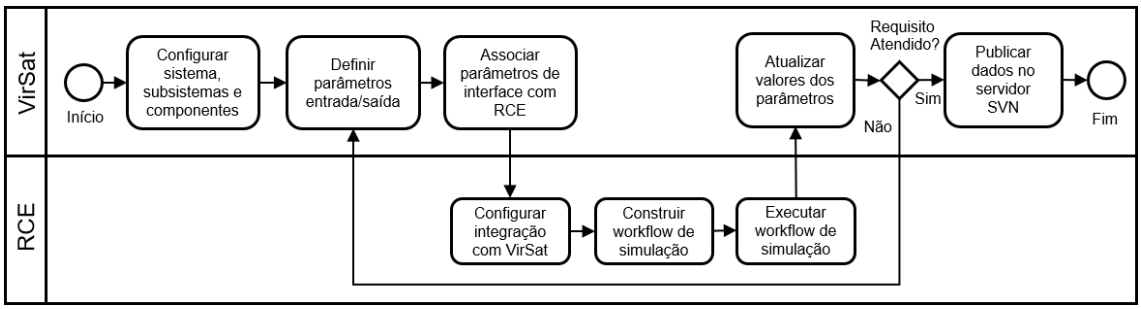
g) voltar ao VirSat e atualizar os valores dos parâmetros de saída do componente em estudo com os resultados da planilha Excel (como a associação já foi realizada no passo c, basta clicar em um botão para realizar a atualização de todos os parâmetros);

h) avaliar o impacto dos resultados nos demais componentes do sistema;

i) caso os novos valores devam ser compartilhados com os demais engenheiros do time, deve-se atualizar o repositório SVN com os novos valores dos parâmetros, usando o VirSat.

O processo descrito acima pode ser repetido várias vezes, com os engenheiros realizando estudos de balanceamento entre diversos parâmetros do sistema, até que se obtenha uma configuração final que será analisada em uma reunião de revisão de projeto (*design review*), o que geralmente significa uma mudança de fase no projeto do sistema. A Figura 5.6ilustra este processo.

Figura 5.6–Processo de operação do ambiente proposto.



Fonte: Produção do autor.

Para documentar a origem dos valores destes parâmetros publicados no VirSat, o engenheiro publica adicionalmente um arquivo ZIP do RCE contendo todos os dados utilizados na simulação, executando os passos a seguir:

a) no RCE, selecionar o projeto que contém o *workflow* e os dados usados na simulação que resultou nos valores finais dos parâmetros publicados no VirSat;

b) usar a função *Export* do RCE e selecionar a opção *Archive File* para criar o arquivo ZIP com os dados da simulação;

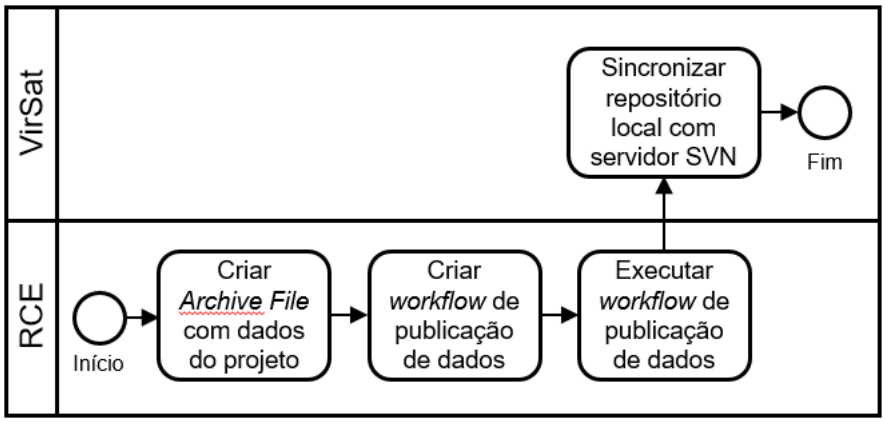
c) criar um *workflow* no RCE incluindo o componente utilizado para publicar o arquivo ZIP no VirSat;

d) executar o *workflow* no RCE, informando como entradas o UUID da planilha Excel, o nome do estudo e o arquivo ZIP. Por meio desta execução, o arquivo ZIP é copiado para o repositório SVN local do VirSat;

e) no VirSat, sincronizar o repositório local com o servidor. Com isso, o arquivo ZIP é copiado para o servidor SVN e fica disponível para consulta para os demais usuários, quando estes sincronizarem seus repositórios locais com o servidor.

A Figura 5.7 ilustra este processo.

Figura 5.7 – Processo de arquivamento de dados.



Fonte: Produção do autor.

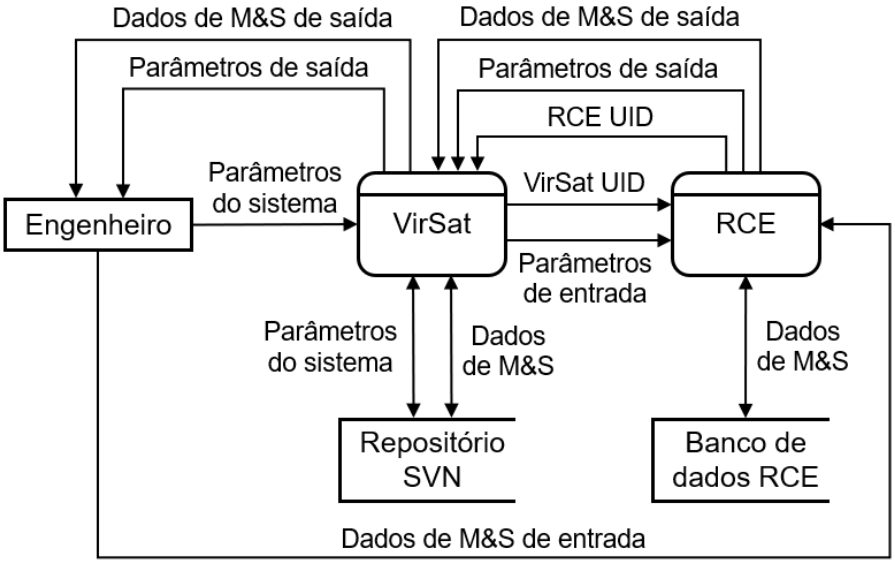
A Figura 5.8 ilustra o fluxo de dados que ocorre durante a operação do ambiente após sua configuração. Uma vez que o estudo tenha sido criado no VirSat, incluindo a hierarquia de sistema, subsistemas e componentes, um engenheiro usuário do ambiente cria parâmetros que caracterizam os subsistemas e componentes do sistema e atribui valores a eles no VirSat. O engenheiro também prepara dados de entrada de M&S, como *scripts*, modelos, arquivos de aplicações de software, que serão utilizados pelo RCE. Ao término da execução dos *workflows* de simulação no RCE, o VirSat recebe os resultados e os disponibiliza para análise pelo engenheiro.

O VirSat recebe os valores de parâmetros fornecidos pelo engenheiro e fornece para o RCE o UID do subsistema ou componente que será analisado. O UID é um código de identificação usado para localizar os valores dos parâmetros de entrada no VirSat. Ao término de cada execução de um *workflow* de simulação, o VirSat recebe do RCE os valores atualizados dos parâmetros de saída do sistema e o UID do *workflow* de simulação, que pode ser utilizado para busca e referência futuras. Quando for necessário trocar valores de parâmetros e dados de simulação com os demais engenheiros, é realizada uma sincronização com o repositório SVN. Ao final de uma fase de projeto, o VirSat recebe do RCE um conjunto de dados de M&S para

arquivamento em seu repositório, incluindo o processo utilizado para a geração dos parâmetros finais do sistema, com o intuito de preservá-los para um eventual reuso futuro para reavaliação dos resultados das simulações.

O RCE recebe do VirSat o UID do subsistema ou componente a ser analisado e, por meio dele, localiza os valores dos parâmetros de entrada no VirSat. O RCE recebe também dados de entrada de M&S fornecidos pelo engenheiro, necessários para configurar a integração entre RCE e VirSat e configurar a execução do *workflow* de simulação. Ao executar o *workflow*, o RCE sincroniza seu banco de dados com todos os dados utilizados. Após esta execução, o RCE envia para o VirSat os valores atualizados dos parâmetros de saída, assim como o UID do *workflow* de simulação. Adicionalmente, ao fim de uma revisão de projeto, o RCE envia para o VirSat um conjunto de dados de M&S para arquivamento em seu repositório.

Figura 5.8 – Diagrama de fluxo de dados do ambiente proposto.



Fonte: Produção do autor.

6 ESTUDO DE CASO ESPACIAL: IMPLEMENTAÇÃO E AVALIAÇÃO

Neste capítulo é apresentado o estudo de caso espacial, que consiste no gerenciamento dos dados e do processo de simulação do sistema de controle de atitude e órbita do satélite Amazônia-1, que foi baseado na plataforma multimissão do INPE. O propósito deste estudo é a validação do ambiente proposto no capítulo 5.

6.1. **Plataforma Multimissão**

De acordo com Moraes (2017), por volta do ano 2000 o INPE promoveu o início de estudos para o desenvolvimento de uma Plataforma Multimissão (PMM) para servir de base para os futuros satélites brasileiros. Segundo o INPE (2017b), a PMM introduz um conceito moderno de arquitetura de satélites, cujo propósito é reunir, em uma única plataforma, todos os equipamentos que realizam funções necessárias à operação de um satélite, independentemente do tipo de órbita, como apontamento, geração de energia, controle térmico, gerenciamento de dados e telecomunicação. Assim, a PMM é uma plataforma que serve como base para o desenvolvimento de satélites com diferentes tipos de carga útil, sendo flexível ao ser capaz de atender aos requisitos de diferentes tipos de missão espacial.

A grande vantagem de se desenvolver uma plataforma que atenda múltiplas missões é a redução de custos e de prazos no desenvolvimento de novas missões que a utilizem como base, pois após o primeiro processo de desenvolvimento e qualificação da plataforma, os demais processos de adaptação da plataforma às diversas missões são mais rápidos. O uso de uma plataforma já qualificada também propicia o aumento da confiabilidade das futuras missões espaciais que façam reuso da plataforma (INPE, 2017a).

Conforme informado pelo INPE (2017b), a PMM é uma plataforma genérica, com massa de 250 kg, para satélites na classe de 500kg. A PMM provê todos os recursos necessários para sustentar, em órbita, o funcionamento de

diversas cargas úteis de até 280 kg. A iniciativa para proposta e concepção da PMM foi da Agência Espacial Brasileira (AEB), que conduziu a contratação do desenvolvimento do sistema por empresas brasileiras, sendo o INPE o interveniente técnico responsável pela coordenação do projeto.

A PMM é composta de diversos subsistemas responsáveis por desempenhar funções específicas para garantir a operação do satélite em órbita conforme os requisitos da missão. De acordo com o INPE (2017b), os subsistemas da PMM são:

a) Estrutura Mecânica do Módulo de Serviço;

b) Subsistema de Controle de Atitude e Órbita e Supervisão de Bordo;

c) Propulsão;

d) Gerador solar;

e) Suprimento de Energia;

f) Controle Térmico;

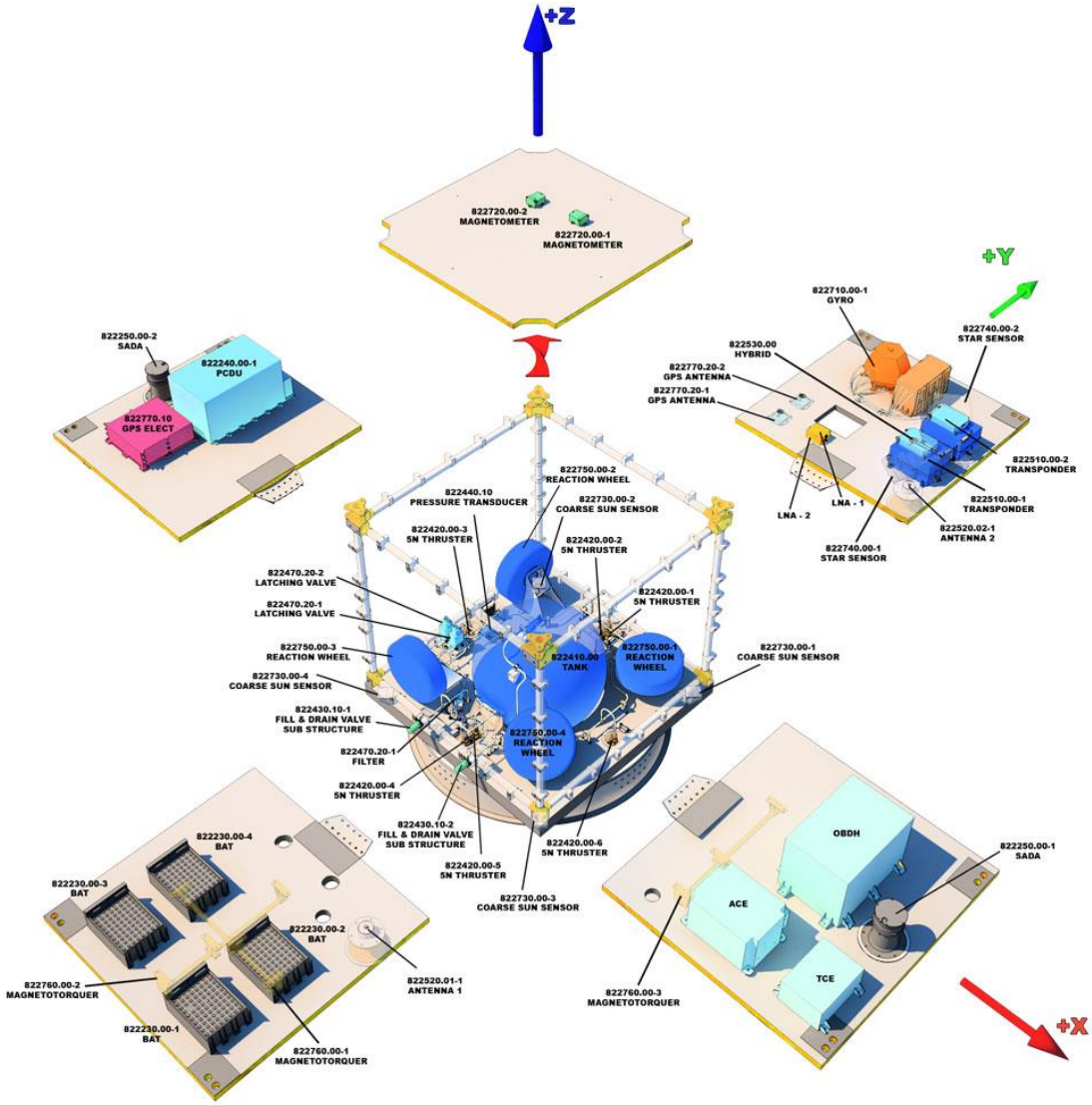
g) Telemetria e Telecomando.

A Figura 6.1 mostra a arquitetura da PMM.

6.2. **Satélite Amazônia-1**

De acordo com o INPE (2017c), o Amazonia-1 é o primeiro satélite de observação da terra completamente projetado, integrado, testado e operado pelo Brasil, com lançamento previsto para novembro de 2018. Este será o primeiro satélite a fazer uso da PMM como base para o desenvolvimento da missão. O satélite conta com dois módulos independentes: um Módulo de Serviço (representado pela PMM) e um Módulo de Carga Útil (câmera imageadora e equipamentos de gravação e transmissão de dados de imagens).

Figura 6.1 – Arquitetura da plataforma multimissão.



Fonte: INPE (2017c).

A missão do Amazônia-1 é gerar imagens de um mesmo ponto do planeta a cada 5 dias, permitindo aplicações como a geração de alerta de desmatamento na Amazônia, pois aumenta a probabilidade de captura de imagens úteis diante da cobertura de nuvens na região (INPE, 2017b).

De acordo com o INPE (2017c), os subsistemas da carga útil do satélite Amazônia-1 são:

a) Subsistema da Estrutura Mecânica do Módulo de Carga Útil;

b) Subsistema câmera imageadora (WFI);

c) Unidade Remota de Telemetria (RTU);

d) Conversores DC/DC;

e) Subsistema Gravador Digital de Dados (DDR);

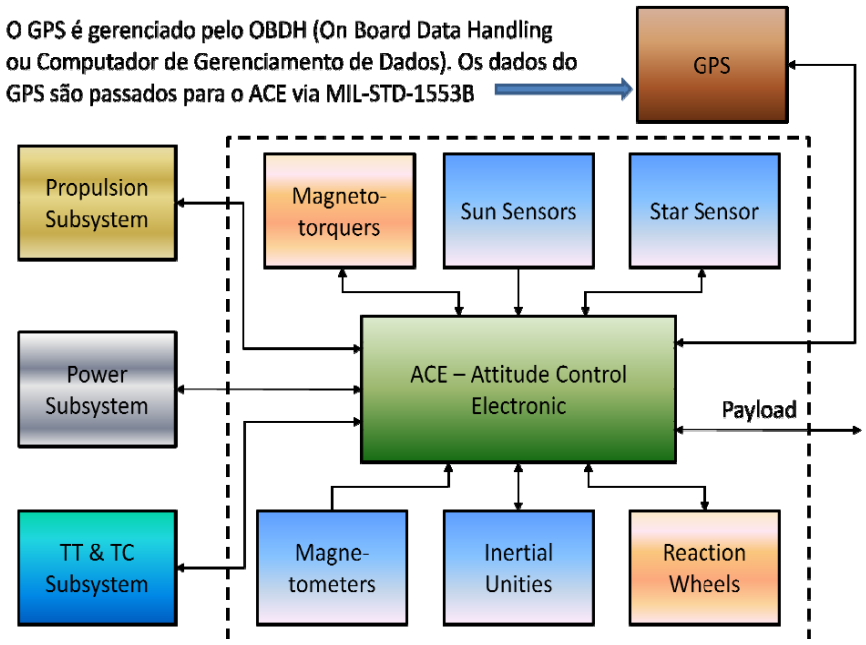
f) Subsistema Transmissor de Dados da Câmera (AWDT).

6.3. **Sistema de controle de atitude e órbita**

O subsistema ACDH (*Attitude Control and Data Handling*) é responsável pelo controle de atitude e órbita e pelo processamento e armazenamento de dados a bordo. Este subsistema é dividido em duas partes: uma dedicada ao Sistema de Controle de Atitude e Órbita (SCAO), também chamado de *Attitude and Orbit Control System* (AOCS), e outra para o processamento dos dados a bordo e gerenciamento do satélite, ou *On Board Data Handling* (OBDH).

Segundo Moraes (2017), o satélite Amazônia-1 será estabilizado em três eixos com o apontamento de uma câmera para a Terra, que deverá produzir imagens com maior frequência e maior definição, adequadas para monitorar o ambiente e gerenciar recursos naturais. O SCAO contém atuadores e sensores que permitirão satisfazer os requisitos da missão. A Figura 6.2 mostra os elementos do SCAO do satélite Amazônia-1 e a interface com três outros subsistemas, além do OBDH.

Figura 6.2 - Subsistema de Controle de Atitude e Órbita do satélite Amazônia-1.



Fonte: Tagawa (2013).

O SCAO é composto de sensores (magnetômetros, unidades inerciais, sensores solares, sensores de estrelas), atuadores (bobinas magnéticas, rodas de reação) e a *Attitude Control Electronics* (ACE) e deve prover as seguintes funções de controle de atitude e órbita (MORAES, 2017):

a) controle de atitude estabilizado em três eixos no Modo Nominal, permitindo apontamento para Terra;

b) determinação de atitude a bordo;

c) aquisição e manutenção segura de atitude após a fase de lançamento ou ocorrência de falha;

d) controle do posicionamento/velocidade dos painéis solares;

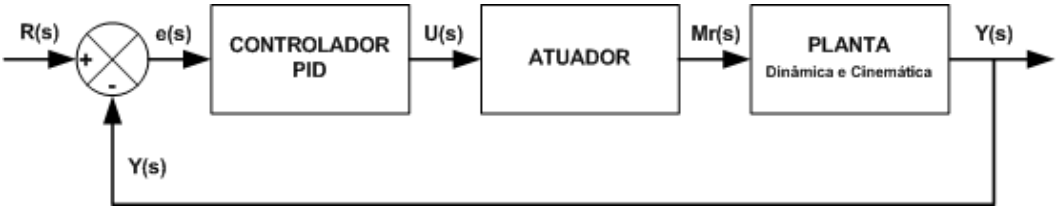
e) determinação da posição e velocidade do satélite;

f) controle dos propulsores para aquisição e manutenção de órbita;

g) dessaturação das rodas de reação.

Como mostrado no trabalho de Tagawa (2013) e descrito por Moraes (2017), a modificação da atitude do satélite é alcançada por meio de rodas de reação (atuadores) que giram com determinada velocidade angular (ω_r), impondo um torque à estrutura do satélite (M_r). Este torque induz uma velocidade angular ω. Baseado nos valores de ω, ω_r e M_r, é possível determinar a atitude do satélite em um determinado momento em relação aos ângulos em torno dos eixos de rolamento (φ), arfagem (θ) e guinada (ψ) do referencial do corpo. 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. Este sistema é ilustrado na Figura 6.3.

Figura 6.3 – Diagrama de blocos do SCAO.



Fonte: Moraes (2017).

Uma simulação envolvendo este modelo tem como entradas os ângulos de referência de rolamento, arfagem e guinada que o satélite deve atingir, assim como os demais dados de referência do satélite, mostrados na Tabela 6.1. Ao se executar a simulação, a resposta dinâmica do satélite é determinada. Devido à atuação das rodas de reação, controladas pelo SCAO, a orientação do satélite é modificada. Os ângulos de orientação, inicialmente igualados a 0°, são modificados até atingirem os valores de referência desejados.

Tabela 6.1–Dados e condições iniciais do SCAO.

Parâmetro	**Valor**	**Descrição**
h =	0,60	passo da simulação (s)
Dados do Controlador PID		
Ti x =	1,0000	Ganho Integral PID em x

Parâmetro	Valor	Descrição
Kp x =	40,5931	Ganho Proporcional PID em x
Td x =	454,1050	Ganho Derivativo PID em x
Ti y =	1,0000	Ganho Integral PID em y
Kp y =	51,7854	Ganho Proporcional PID em y
Td y =	556,9350	Ganho Derivativo PID em y
Ti z =	1,0000	Ganho Integral PID em z
Kp z =	44,3541	Ganho Proporcional PID em z
Td z =	488,6600	Ganho Derivativo PID em z
Dados e condições iniciais do Satélite/Órbita		
Isx =	295,71	Momento de Inércia do Satélite em x (kg.m2)
Isy =	501,37	Momento de Inércia do Satélite em y (kg.m2)
Isz =	364,82	Momento de Inércia do Satélite em z (kg.m2)
Xϕ ref =	0,10	Posicionamento desejado em x (radianos)
Yθ ref =	0,10	Posicionamento desejado em y (radianos)
ZΨ ref =	0,10	Posicionamento desejado em z (radianos)
Xϕ (0) =	0,10	Posicionamento inicial em x (radianos)
Yθ (0) =	0,10	Posicionamento inicial em y (radianos)
ZΨ (0) =	0,10	Posicionamento inicial em z (radianos)
ωx (0) =	0,00	Velocidade angular x inicial do satélite (rpm)
ωy (0) =	0,00	Velocidade angular y inicial do satélite (rpm)
ωz (0) =	0,00	Velocidade angular z inicial do satélite (rpm)
Mx (0) =	0,00	Torque inicial do Satélite em x (kg.m2)
My (0) =	0,00	Torque inicial do Satélite em y (kg.m2)
Mz (0) =	0,00	Torque inicial do Satélite em z (kg.m2)
ωrx (0) =	0,00	Velocidade angular x inicial da roda (rpm)
ωry (0) =	0,00	Velocidade angular y inicial da roda (rpm)
ωrz (0) =	0,00	Velocidade angular z inicial da roda (rpm)
Mrx (0) =	0,00	Torque inicial da roda em x (kg.m2)
Mry (0) =	0,00	Torque inicial da roda em y (kg.m2)
Mrz (0) =	0,00	Torque inicial da roda em z (kg.m2)
ωo =	0,001080	Velocidade orbital média (rad/s)
Dados das Rodas de Reação		
Iw =	0,0150	Momento de Inércia das rodas (kg.m2)
ωrmax =	7500,00	Velocidade angular máxima das rodas (rpm)
Mrmax =	0,60	Torque máximo das rodas (N.m)
Kw =	0,06	Ganho das rodas
Tw =	20,00	Constante de tempo das rodas (s)

Fonte: Moraes (2017).

Ainda de acordo com Moraes (2017), para que o satélite Amazônia-1 cumpra sua missão, a atitude do satélite, bem como sua taxa de variação, deve ser controlada nos três eixos em seu Modo Nominal de operação para cumprir com os seguintes requisitos:

a) precisão de apontamento: < 0.05° (3σ);

b) estabilidade:

- deriva ("*Drift*") < 0.001°/s com frequências até 2Hz;
- variação pico a pico < 0.005° com frequências de 2Hz a 10Hz.

c) "*jitter*" < 0.0005° (1σ) com frequências acima de 10Hz até 100Hz;

d) determinação de atitude: ≤ 0.005° (3σ);

e) desvio ("*off pointing*") de até 30° em torno do eixo de rolamento em 180 segundos.

6.4. **Definição do estudo de caso espacial**

Para validar o ambiente proposto no capítulo 5, é definido um estudo de caso na área espacial que consiste no gerenciamento dos dados e do processo de simulação do SCAO mostrado na seção 6.3 no contexto de uma sessão de engenharia simultânea na fase A do projeto do satélite Amazônia-1. O foco deste estudo de caso é o gerenciamento de dados e processos, e não o modelamento do sistema, que é mostrado em detalhes na dissertação de Moraes (2017).

Neste estudo de caso, o VirSat é usado para modelar a estrutura de dados do satélite, utilizando o modelo de dados conceitual proposto. Este modelo contempla as relações hierárquicas entre componentes e subcomponentes do sistema, com base na PMM, e seus parâmetros correspondentes. A seguir, é definida uma sessão de engenharia simultânea onde o requisito de "desvio de

até 30° em torno do eixo de rolamento em 180 segundos” é verificado por meio de uma simulação. É utilizado como referência nesta simulação o modelo em MATLAB / Simulink do SCAO descrito no trabalho de Moraes (2017), que é implementado em um *workflow* do RCE. Para se executar a simulação, o valor do ângulo de referência para o rolamento é atualizado no VirSat para o valor de 0.523599 radianos (correspondente a 30°) e é criado o parâmetro de saída “Tx” do SCAO com o valor do tempo necessário pelo sistema para levar o satélite até o ângulo de rolamento desejado. O *workflow* do RCE lê os valores destes parâmetros, executa a simulação no MATLAB / Simulink e atualiza o valor de Tx com o tempo necessário para executar a manobra, que deve ser menor que 180 segundos.

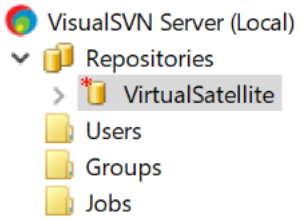
Durante a sessão, especialistas podem alterar várias vezes os valores dos parâmetros do satélite e do SCAO no VirSat e executar várias simulações no RCE até que o requisito seja cumprido. O especialista responsável pelo SCAO publica no VirSat, no momento da revisão de projeto para troca de fase, o arquivo ZIP com a exportação da estrutura de dados do *workflow* do RCE para documentação e possível reuso em projetos futuros.

6.5. **Configuração da simulação do SCAO no ambiente proposto**

6.5.1. Configuração do VirSat

O primeiro passo na configuração do ambiente é a criação de um repositório SVN no computador destinado a ser o servidor SVN. No exemplo mostrado na Figura 5.4, este papel é exercido pelo computador 1. Foi utilizada neste estudo de caso a ferramenta VisualSVN Server Manager (VisualSVN, 2017) que possibilita a configuração de um servidor SVN de forma fácil e rápida. Foi então criado o repositório SVN chamado “*VirtualSatellite*” no computador 1, conforme mostrado na Figura 6.4.

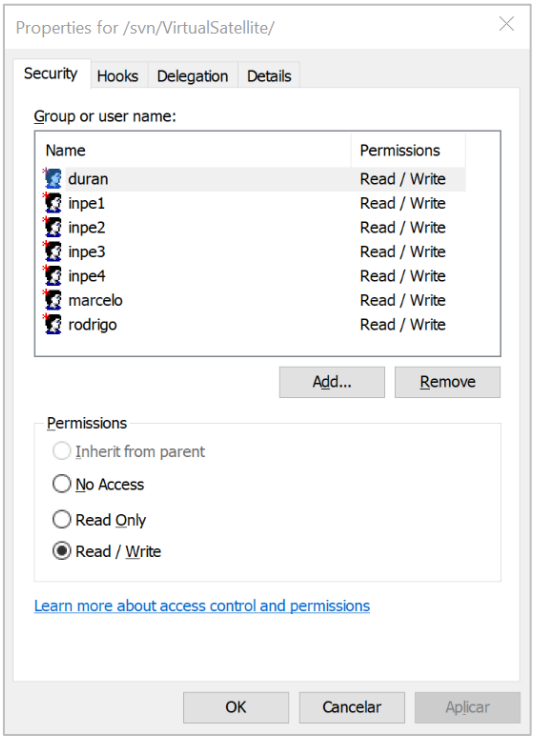
Figura 6.4 – Repositório SVN.



Fonte: Produção do autor.

Após a criação do repositório, deve-se criar os usuários que terão acesso ao repositório, bem como seus níveis de acesso, que podem ser de apenas leitura ou de leitura e escrita, como mostrado na Figura 6.5.

Figura 6.5 – Configuração de acesso ao repositório SVN.



Fonte: Produção do autor.

A seguir, o VirSat é instalado nos computadores de usuário (computadores 2 e 4, conforme mostrado na Figura 5.4) e o arquivo de propriedades do VirSat com a definição de um novo estudo de engenharia simultânea ("*INPE_PMM.properties*") é criado pelo responsável pelo estudo, em um dos

computadores de usuário. Este arquivo especifica o nome do estudo e o repositório onde será armazenado. Neste estudo de caso, foi escolhido o estudo "*INPE_PMM*" e o repositório "*VirtualSatellite*", conforme mostrado na Figura 6.6. O diretório padrão de instalação "C:\VirSat\" foi padronizado em todos os computadores.

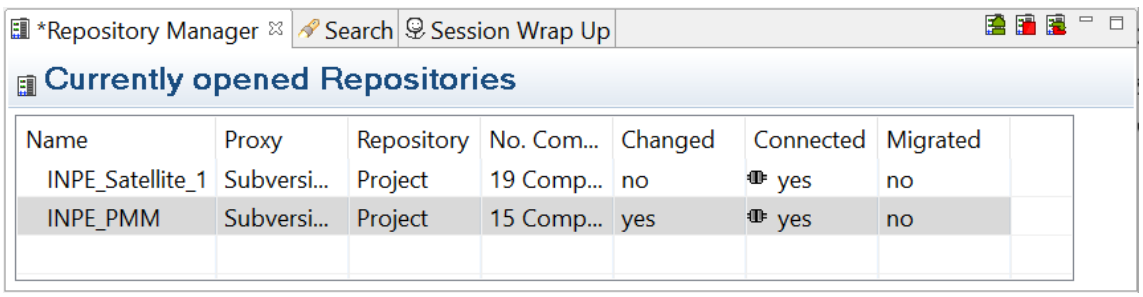
Figura 6.6 – Arquivo de propriedades do VirSat para o estudo de caso espacial.

```
virsat.persistence.repository.proxy = de.dlr.virsat.datastore.proxy.user.SVNResourceRepositoryProxy

# the name of the study in the SVN repository. Must not be empty
virsat.persistence.repository.name = INPE_PMM

# Path to the SVN repository, can be left blank for a local study without collaboration with other users
#virsat.persistence.repository.remote=
virsat.persistence.repository.remote= https://DESKTOP-126D28U/svn/VirtualSatellite

# put your credentials here or leave blank, and VirSat will ask for them
virsat.persistence.repository.user=duran
virsat.persistence.repository.pass=duran
```

Fonte: Produção do autor.

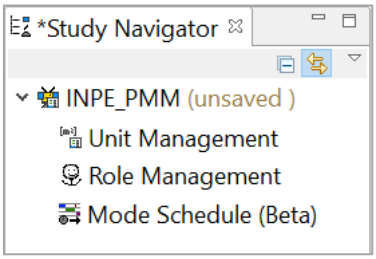
Em seguida, o VirSat é iniciado em modo de administração pelo responsável pelo estudo, por meio da execução de seu *script* de inicialização ("*VirSat.bat*") editado com a opção *"-disciplineAdmin*" na linha de comando que lança o VirSat. Por meio do *Repository Manager*do VirSat, mostrado na Figura 6.7, o arquivo "*INPE_PMM.properties*" é aberto e o VirSat cria um novo estudo chamado "*INPE_PMM*", que pode ser visto no *Study Navigator* mostrado na Figura 6.8. Usando a funcionalidade "*Commit to Repository*" do VirSat, é realizada a primeira publicação do VirSat no repositório. Com isto, é criada uma nova pasta com o nome do estudo no repositório SVN, o que é mostrado na Figura 6.9.

Figura 6.7 – Configuração do repositório SVN no VirSat.



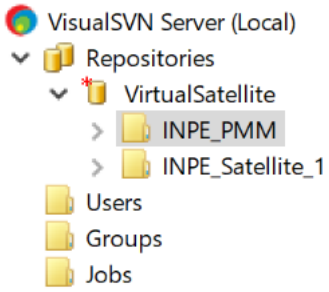
Fonte: Produção do autor.

Figura 6.8 – *Study Navigator* mostrando o novo estudo INPE_PMM.



Fonte: Produção do autor.

Figura 6.9 – Pasta com nome do estudo criada no repositório SVN.

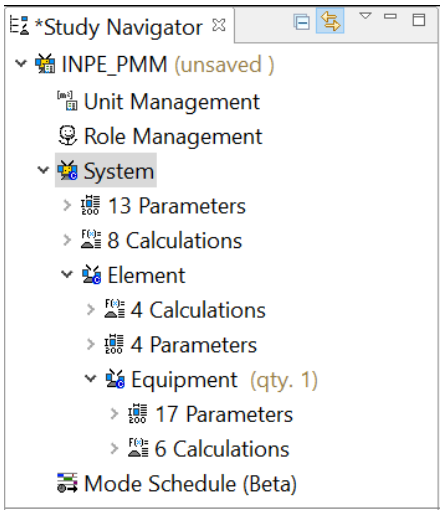


Fonte: Produção do autor.

A seguir, por meio da funcionalidade "*Templates* > *Create New Study Template*", é criado um modelo de estrutura de dados de um sistema padrão, chamado "*System*", que pode ser visto na Figura 6.10. Este sistema contém um subsistema chamado "*Element*" que, por sua vez, possui um equipamento chamado "*Equipment*". Tanto o sistema principal quando o subsistema e equipamento possuem parâmetros de projeto ("*Parameters*") e cálculos envolvendo estes parâmetros ("*Calculations*") associados.

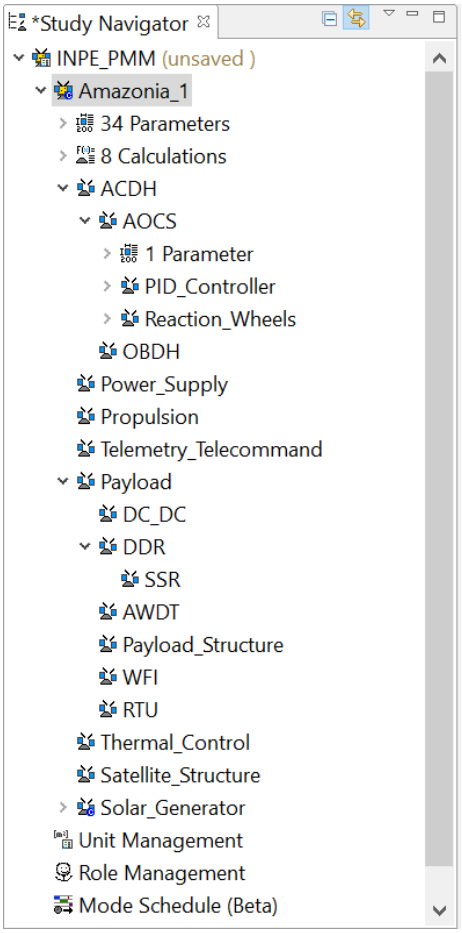
O sistema padrão "*System*", que representa o satélite, é então renomeado para "*Amazonia_1*" e novos subsistemas são criados, um para cada subsistema do satélite Amazônia-1, que é composto pelos subsistemas da PMM e subsistemas da carga útil, apresentados nas seções 6.1 e 6.2, respectivamente. Esta representação da estrutura de dados do satélite é mostrada na Figura 6.11.

Figura 6.10 – Criação do modelo de estrutura de dados.



Fonte: Produção do autor.

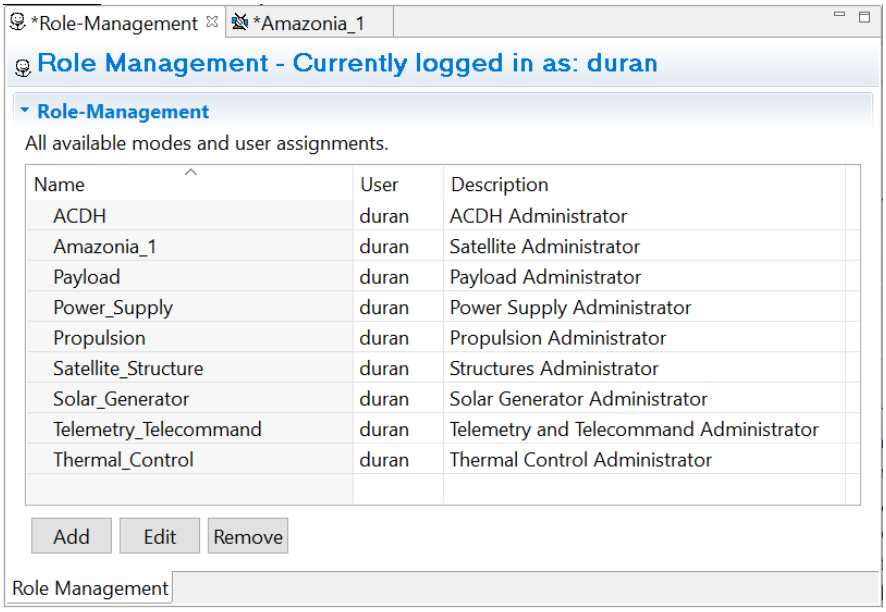
Figura 6.11 – Estrutura de dados do satélite Amazônia-1 no VirSat.



Fonte: Produção do autor.

Após a criação da estrutura de dados, é necessário atribuir, a cada subsistema, um administrador que tem a permissão de editar os dados do subsistema. Isto é realizado com a criação uma disciplina para cada subsistema, usando a funcionalidade de "*Role Management*", mostrada na Figura 6.12, e a associação de cada disciplina a um usuário administrador. Para facilitar o estudo, todos os administradores foram associados ao mesmo usuário. Em um caso real, os usuários poderiam ser distintos.

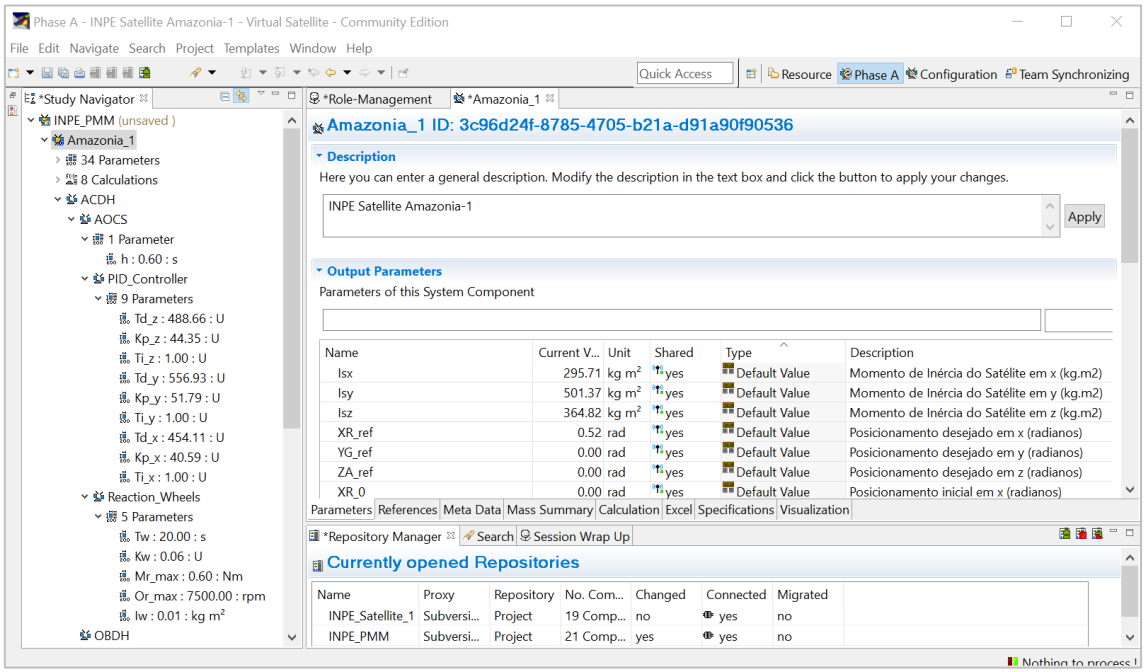
Figura 6.12 – *Role Management* no VirSat.



Fonte: Produção do autor.

No VirSat, após a criação destas disciplinas, é necessário atribuí-las aos seus respectivos subsistemas, usando a funcionalidade já mostrada na Figura 4.27. Em seguida, os parâmetros definidos na Tabela 6.1 são criados no VirSat para o sistema principal e seus subsistemas. A Figura 6.13 mostra, no *Study Navigator* (lado esquerdo), os parâmetros do SCAO (ou AOCS, "*Attitude and Orbit Control System*", em Inglês) assim como do controlador PID ("*PID_Controller*") e das rodas de reação ("*Reaction_Wheels*"). No lado direito da figura, os parâmetros do satélite Amazônia-1 são mostrados em detalhes, na visualização detalhada do componente.

Figura 6.13 – Parâmetros utilizados na simulação do SCAO.



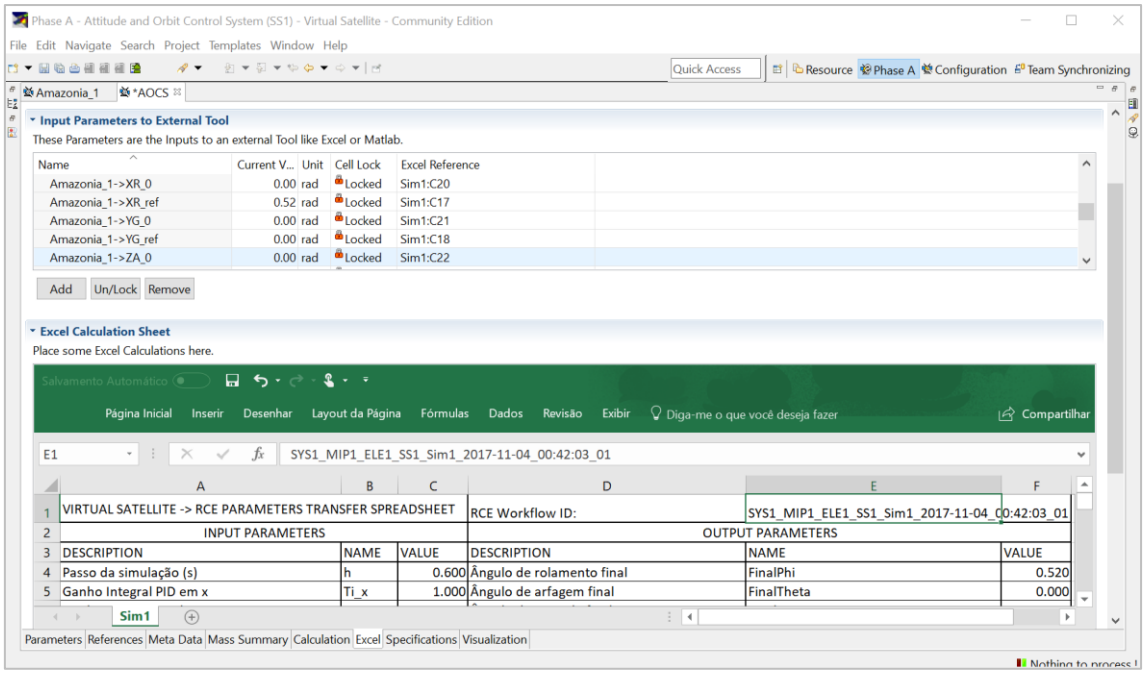
Fonte: Produção do autor.

Para verificar o requisito de “desvio de até 30° em torno do eixo de rolamento em 180 segundos”, é realizada uma simulação onde o satélite é atuado por meio do giro de rodas de reação, saindo de uma posição inicial onde o ângulo de rolamento é 0° até chegar a uma posição de referência onde este ângulo é de 30°. Nesta simulação, é verificado o tempo em que o SCAO leva para estabilizar o satélite nesta nova posição, que deve ser inferior a 180 segundos.

Para configurar esta simulação, o parâmetro “*XR_ref*” do satélite Amazonia_1 no VirSat (correspondente ao parâmetro “*Xϕ ref*” da Tabela 6.1), que representa o ângulo de rolamento de referência a ser atingido na simulação, é definido com o valor de 0.523599 radianos, o equivalente a 30°. Da mesma forma, o parâmetro “*XR_0*” (correspondente ao parâmetro “*Xϕ (0)*” da Tabela 6.1), que representa o ângulo de rolamento inicial, é definido com o valor de 0 radianos. São criados também os parâmetros “*TimePhi*” e “*FinalPhi*”. Aquele representa o tempo em segundos que o satélite leva para estabilizar-se no ângulo de rolamento de referência, enquanto este representa o ângulo de rolamento atingido ao final da simulação.

Para que os valores dos parâmetros sejam acessados pela simulação do SCAO no *workflow* do RCE, é necessário mapeá-los como parâmetros de entrada e saída em uma planilha Excel padronizada. Os parâmetros "*XR_ref*" e "*XR_0*", assim como demais parâmetros de outros subsistemas, são mapeados como parâmetros de entrada. Os parâmetros "*FinalPhi*" e "*TimePhi*", assim como outros parâmetros, são mapeados como parâmetros de saída. A Figura 6.14 e a Figura 6.15 mostram como este mapeamento é feito no VirSat.

Figura 6.14–Mapeamento de parâmetros de entrada do VirSat para simulação RCE.

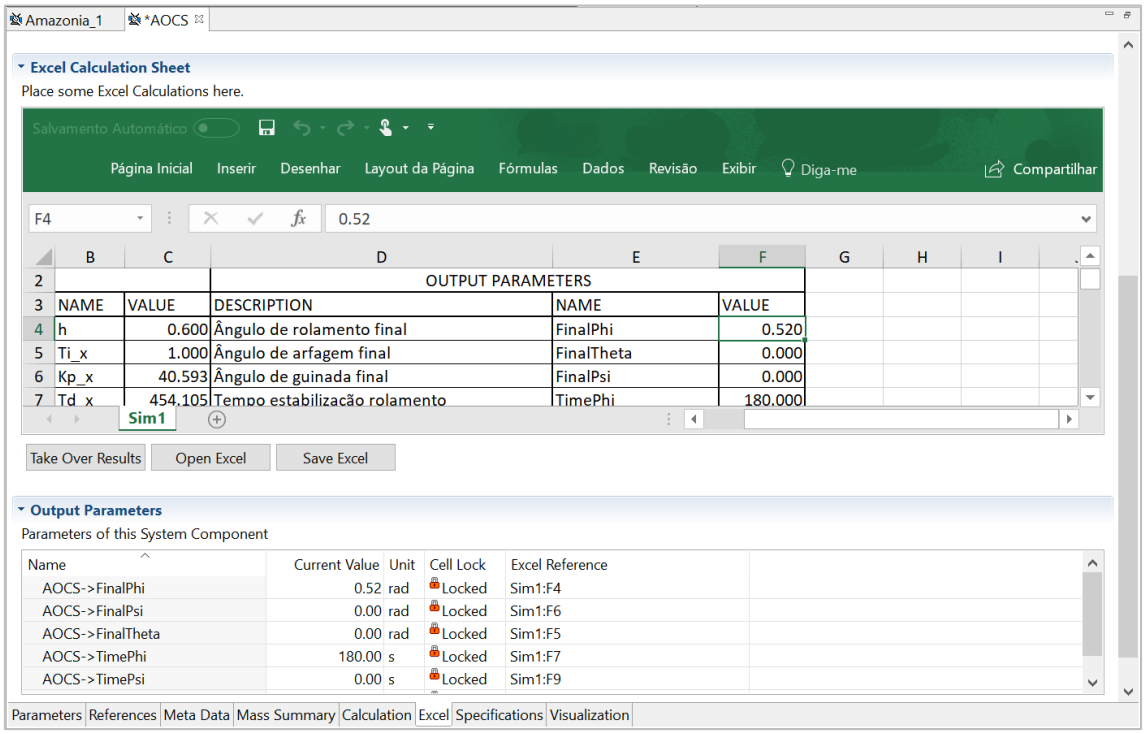


Fonte: Produção do autor.

6.5.2. Configuração do RCE

Após a definição da estrutura de dados no VirSat, o próximo passo é configuração do ambiente de processos de simulação no RCE. Para isso, o RCE foi instalado e configurado em quatro computadores conforme mostrado na Figura 5.3.

Figura 6.15 - Mapeamento de parâmetros de saída do VirSat para simulação RCE.



Fonte: Produção do autor.

Para este trabalho, foi utilizada a versão 8.0.1 do RCE que pode ser obtida na página do RCE na internet: http://rcenvironment.de/. A instalação do RCE foi realizada conforme descrito na seção 4.2.5, descompactando o arquivo ZIP baixado em uma pasta local do computador. Foi padronizada a pasta C:\RCE\ para receber os arquivos de instalação do RCE em todos os computadores.

Para executar a simulação do SCAO do satélite Amazônia-1 no RCE, é necessário desenvolver componentes para executar as seguintes funções:

a) Ler os valores dos parâmetros da simulação definidos no VirSat e escrever um *script* de inicialização do MATLAB/Simulink com estes valores;

b) Executar o MATLAB em modo “linha de comando” (sem interface gráfica), carregar o modelo Simulink e *scripts* de inicialização e executar a simulação (incluindo a geração de arquivos de saída com resultados);

c) Ler o arquivo de resultados da simulação, extrair valores de parâmetros de saída e atualizar o VirSat com os novos valores.

Para isso, são desenvolvidos dois componentes no RCE: “*DataPrep*”, para tratamento de dados de entrada e saída, e “*MATLAB*”, para execução de simulações usando *scripts* e modelos predefinidos. Ambos seguem os requisitos mostrados na seção 4.2.7 e o conceito apresentado na Figura 4.12.

Os componentes desenvolvidos podem, a princípio, ser disponibilizados em qualquer computador da rede RCE. No contexto deste trabalho, o componente “*MATLAB*” foi disponibilizado no computador 3 (“Nó de Computação”) e o componente “*DataPrep*” no computador 2 (“Máquina Cliente”), e o *workflow* que integra todos os componentes foi desenvolvido no computador 4 (outra “Máquina Cliente”). São apresentadas, a seguir, as configurações de cada componente e do *workflow* que contém a simulação.

6.5.2.1. Configuração do componente DataPrep

O componente “*DataPrep*” foi desenvolvido para executar um *script* definido pelo usuário na linguagem Python (“*UserScript*”). Este *script* lê um arquivo de entrada (“*DataFile*”) e duas variáveis de texto (“*UID*” e “*Study_Name*”), realiza seu processamento e escreve um arquivo de saída (“*OutputFile*”). A Tabela 6.2 mostra os detalhes das configurações de entrada e saída deste componente.

Conforme mencionado na seção 4.2.8, a ordem de execução de componentes em um *workflow* do RCE é orientada a dados. Para controlar esta ordem, as entradas de cada componente possuem opções de configuração que informam ao *workflow* como o dado deve ser tratado.

Na opção “*Single (consumed)*” mostrada na Tabela 6.2, a entrada é usada apenas uma única vez durante a execução do *workflow*. Caso o *workflow* possuísse uma retroalimentação (“loop”), nas iterações seguintes este dado não seria utilizado. Na opção “*Constant (not consumed)*”, a entrada fica sempre

disponível para ser utilizada pelo *workflow*, em todas as suas possíveis iterações.

Tabela 6.2 – Configuração das entradas e saídas do componente *DataPrep.*

Inputs / Outputs	**Nome**	**Tipo de Dado**	**Processamento**	**Restrição**
Input	*DataFile*	*File*	*Single (consumed)*	*Required if connected*
Input	*UID*	*Short Text*	*Constant (not consumed)*	*Required if connected*
Input	*Study_Name*	*Short Text*	*Constant (not consumed)*	*Required if connected*
Input	*UserScript*	*File*	*Single (consumed)*	*Required*
Output	*OutputFile*	*File*	-	-

Fonte: Produção do autor.

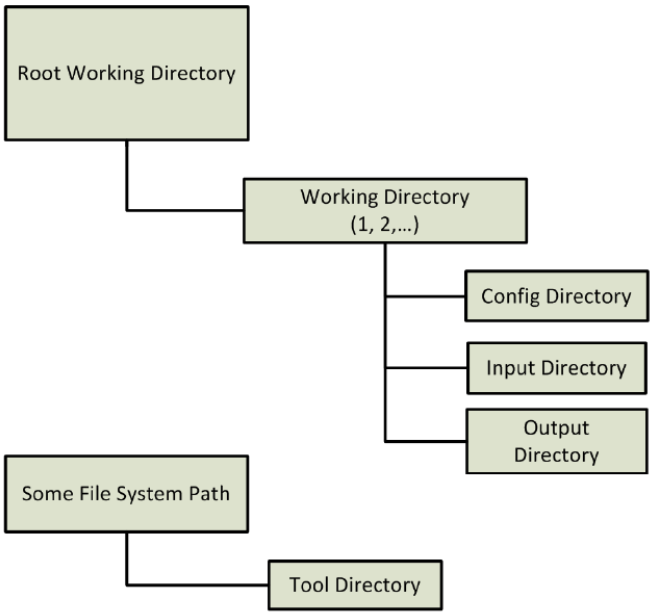
Na opção "*Required if connected*", a entrada é mandatória se existir uma conexão com outro componente que a forneça. Caso não exista uma conexão, a entrada é ignorada. Na opção "*Required",* a entrada é sempre mandatória, independentemente da existência de conexão com outro componente. Por exemplo, caso uma entrada seja considerada "*Required*" por um componente, o RCE sabe que o outro componente que fornece esta entrada deve ser executado primeiramente, pois o componente que requer esta entrada só poderá ser executado quando a mesma estiver disponível.

Quando um componente é executado no RCE, é criada uma estrutura de diretórios contendo um diretório de trabalho raiz ("*Root Working Directory*") no qual são criados vários diretórios de trabalho ("*Working Directory*"), um para cada execução do componente no *workflow*. Dentro deste diretório são criados os diretórios de configuração, entrada e saída de dados ("*Config Directory*", "*Input Directory*" e "*Output Directory*", respectivamente). Todos os diretórios de trabalho raiz são criados dentro de um diretório temporário que pode ser escolhido pelo usuário durante a instalação do RCE. Cada componente ainda conta com um diretório de ferramenta ("*Tool Directory*") que é o diretório onde

está instalada a ferramenta principal que será executada pelo componente. Esta estrutura é mostrada na Figura 6.16.

Segundo o conceito apresentado na Figura 4.12, todo componente no RCE possui *scripts* de pré-processamento, processamento e pós-processamento. O *script* de pré-processamento trata os dados de entrada para adequá-los ao processamento principal (caso seja necessário). O *script* de processamento usa a ferramenta localizada no "*Tool Directory*" para realizar o processamento principal do componente. O *script* de pós-processamento faz o tratamento dos dados de saída (caso seja necessário).

Figura 6.16 – Estrutura de diretórios de trabalho do RCE.



Fonte: Adaptado de RCE (2017).

O *script* de pré-execução do componente "*DataPrep*" faz a cópia do arquivo de dados ("*DataFile*") de qualquer diretório onde ele se encontre para o diretório de trabalho do componente e também escreve as variáveis de texto ("*UID*" e "*Study_Name*"), digitada pelo usuário durante a execução do *workflow*, em arquivos texto que serão utilizados posteriormente.

O *script* de execução do componente "*DataPrep"* executa a ferramenta *Python*, cujo diretório do seu executável é informado na opção "*Tool Directory*", durante a configuração do componente. Como entrada desta ferramenta, é utilizado o *script* definido pelo usuário na linguagem Python ("*UserScript*").

O *script* de pós-execução do componente "*DataPrep*" copia o arquivo de saída "*m-file-init.m*" (caso ele exista) do diretório de trabalho para o diretório de saída, e então define este arquivo como o arquivo de saída oficial do componente. O arquivo "*m-file-init.m*" é um *script* de inicialização da simulação do MATLAB que contém os valores de todos os parâmetros do AOCS definidos no VirSat.

6.5.2.2. Configuração do componente MATLAB

O componente MATLAB foi desenvolvido para executar uma simulação usando o MATLAB/Simulink. O componente tem como entradas um *script* "*m-file-init*", que inicializa as variáveis da simulação, um *script* "*m-file-run*", que executa a simulação propriamente dita, e um modelo opcional "*Simulink Model*" que pode ser usado na simulação. A Tabela 6.3 mostra as entradas e saídas do componente.

Tabela 6.3 - Configuração das entradas e saídas do componente MATLAB.

Inputs / Outputs	**Nome**	**Tipo de Dado**	**Processamento**	**Restrição**
Input	Simulink *Model*	*File*	*Single (consumed)*	*Required if connected*
Input	m-file-init	*File*	*Single (consumed)*	*Required*
Input	m-file-run	*File*	*Single (consumed)*	*Required*
Output	*output_file*	*File*	-	-

Fonte: Produção do autor.

O *script* de pré-execução do componente MATLAB copia os três arquivos de entrada dos locais onde eles se encontram para o diretório de trabalho do

componente, e cria um *script* “*init_run.m*” por meio da concatenação dos *scripts* “*m-file-init*” e “*m-file-run*”.

O *script* de execução lança o MATLAB utilizando parâmetros especiais que fazem com que a sua interface gráfica não seja mostrada, e seja executado o *script* “*init_run.m*” criado no passo anterior. Ao término da execução, o MATLAB é encerrado. Assim, a simulação no MATLAB é executada por meio de uma única linha de comando, sem a necessidade de interação com sua interface gráfica, que são alguns dos requisitos de integração de ferramentas do RCE mostrados na seção 4.2.7.

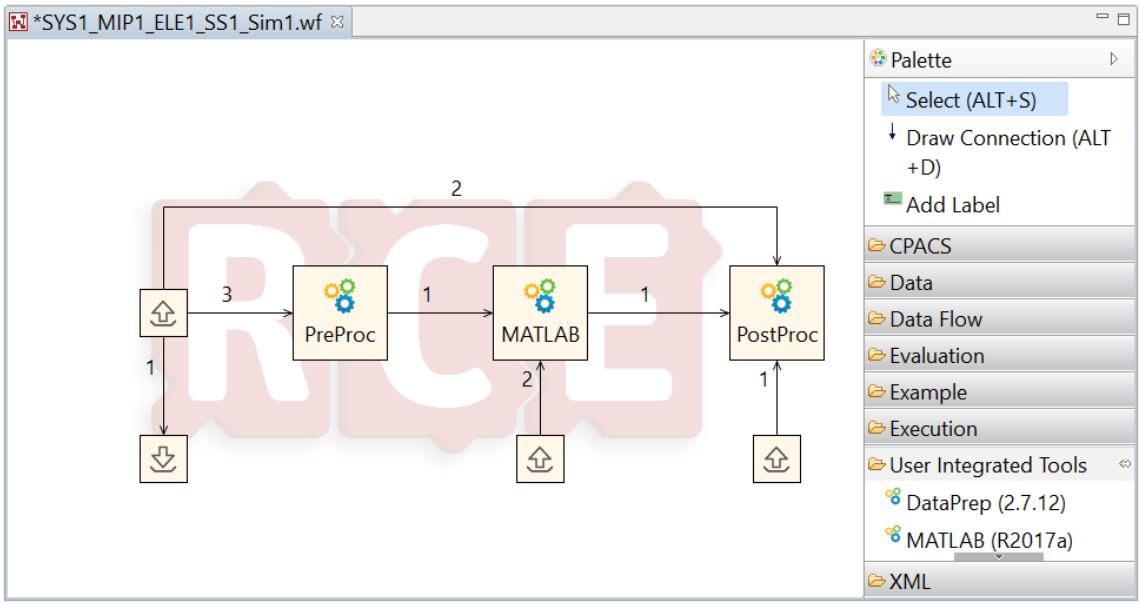
Quando o *script* de execução é lançado, a simulação no MATLAB é iniciada e demora alguns minutos para ser concluída. Entretanto, do ponto de vista do *script* de execução, a atividade já foi concluída e o RCE passa diretamente para o *script* de pós-execução, que faz o tratamento dos resultados. Como a simulação ainda está sendo executada, os resultados ainda não estão disponíveis, e o componente acusaria um erro devido a falta do arquivo de resultado. Assim, é necessário que o *script* de pós-execução execute uma rotina que fique verificando regularmente se a simulação já acabou, e faça o tratamento dos resultados somente após a disponibilização do arquivo de resultado. Quando este arquivo é encontrado, ele é copiado do diretório de trabalho para o diretório de saída e é formalmente associado como uma saída do componente MATLAB.

6.5.2.3. Configuração do *workflow* de simulação

Uma vez definidos os componentes “*DataPrep*” e “*MATLAB*”, eles são integrados por meio de um *workflow* no RCE, criado no *Workflow Editor* mostrado na Figura 6.17. O componente “*DataPrep*” é instanciado duas vezes e renomeado como “*PreProc*” e “*PostProc*”. O componente padrão do RCE chamado “*Input Provider*” (quadrado menor com seta apontando para cima) é instanciado três vezes e renomeado como “*PrePostProc Inputs*”, “*MATLAB Inputs*” e “*PostProc Inputs*”. Estes componentes fornecem os arquivos

necessários à simulação. O componente padrão do RCE chamado "*OutputWriter*" (quadrado menor com seta apontando para baixo) é instanciado uma vez e renomeado como "*Workflow ID Writer*". Este componente escreve valores de variáveis em arquivos predefinidos.

Figura 6.17 – *Workflow* de simulação no RCE.



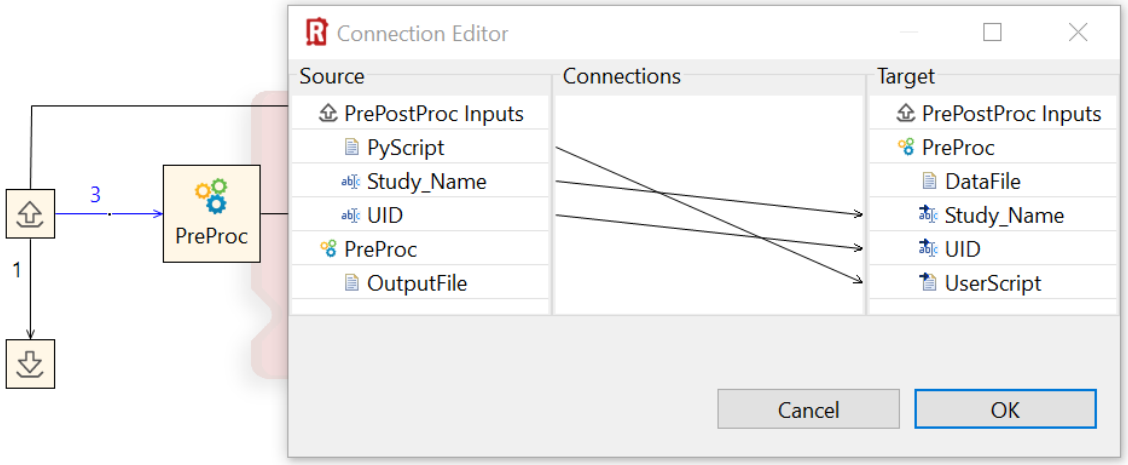
Fonte: Produção do autor.

Após a instanciação dos componentes, é necessário definir as conexões entre seus dados de entrada e saída. Isto é realizado por meio do "*Connection Editor*". A Figura 6.18 mostra um exemplo da definição de duas conexões entre os componentes "*PrePostProc Inputs*" e "*PreProc*". Neste editor, os dados de saída são mostrados à esquerda, classificados como "*Source*" e os dados de entrada são mostrados à direita, classificados como "*Target*". Neste exemplo, a saída "*PyScript*" do componente "*PrePostProc Inputs*" está associada com a entrada "*UserScript*" do componente "*PreProc*".

É possível notar que, na Figura 6.17, apenas as conexões superficiais entre os componentes são mostradas. Os números exibidos nas linhas que unem os componentes representam o número de conexões entre eles. Um mapa

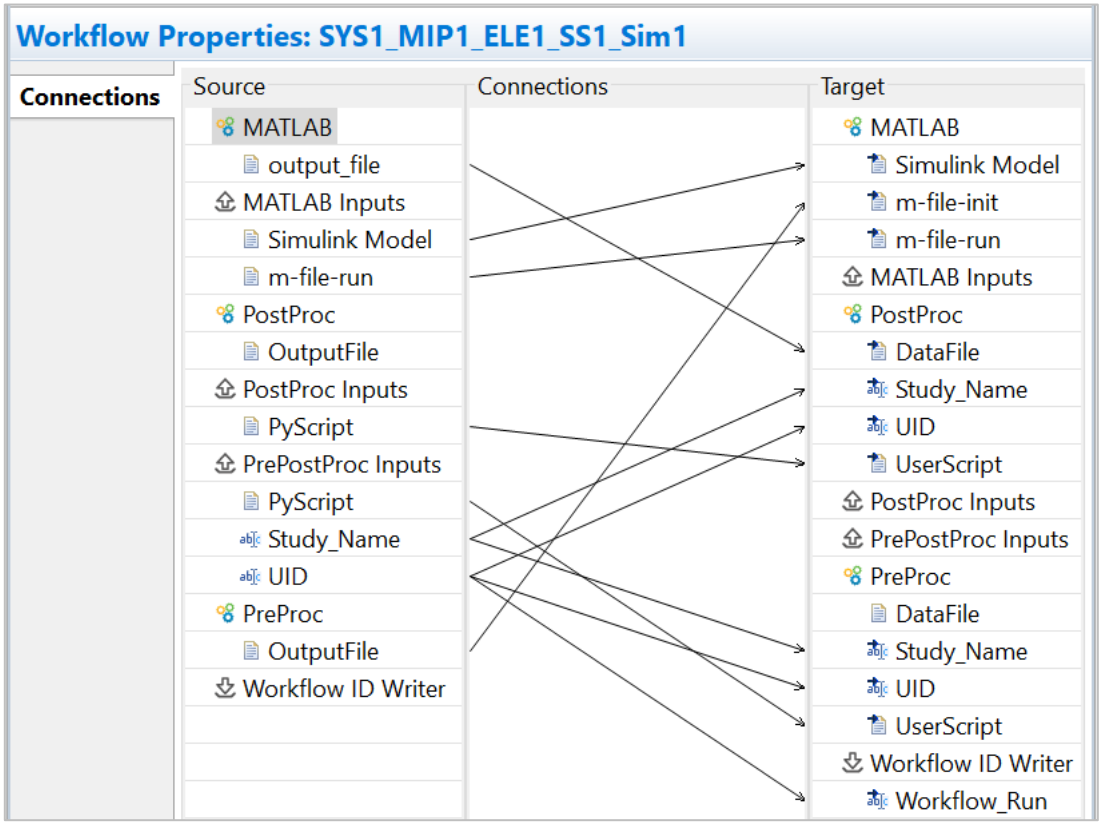
completo com todas as conexões de dados do *workflow* pode ser visualizado por meio da aba "*Properties*" do *workflow*, mostrado na Figura 6.19.

Figura 6.18 – *Connection Editor* do RCE.



Fonte: Produção do autor.

Figura 6.19 – Conexões entre dados de entrada e saída de componentes do *workflow.*



Fonte: Produção do autor.

6.6. **Execução da simulação do SCAO no ambiente proposto**

Em uma sessão de engenharia simultânea durante a fase A do desenvolvimento de um satélite, o engenheiro responsável pelo SCAO deve fazer a verificação do requisito de que o satélite deveria executar uma manobra de rotação de 30° em relação ao seu eixo de rolamento em menos de 180 segundos. Para tanto, ele executa uma simulação usando o ambiente proposto.

No VirSat, cada engenheiro responsável por um determinado componente do satélite executa o comando "*Commit to Repository*", para atualizar o repositório de dados com os valores mais atualizados dos seus respectivos parâmetros. O engenheiro responsável pelo SCAO executa o comando "*Update from Repository*" para atualizar sua cópia local do repositório com estes valores. Em seguida, salva a planilha Excel que contém o mapeamento dos parâmetros da simulação e anota o UUID que representa sua identificação no repositório de dados.

No RCE, o engenheiro responsável pelo SCAO abre o *workflow* com a simulação do sistema, fornece os arquivos de entrada por meio dos componentes "*PrePostProc Inputs*", "*MATLAB Inputs*" e "*PostProc Inputs*" e o executa com o comando "*Run > Execute Workflow*". No início da execução, o *workflow* solicita ao usuário que forneça o UUID da planilha Excel que contém os valores dos parâmetros do SCAO e o nome do estudo. Uma vez fornecidos estes dados, o componente PreProc executa o *script* "*GetVirSatInputParams.py*", que por sua vez localiza a planilha Excel no repositório do VirSat, lê os nomes e valores dos parâmetros e escreve um arquivo de saída "*m-file-init.m*", que será usado posteriormente pelo componente "*MATLAB*" para inicializar os valores das variáveis da simulação. Adicionalmente, o componente "*Workflow ID Writer*" é executado, escrevendo no diretório do projeto um arquivo nomeado com a identificação da execução do *workflow*, que será útil em um passo seguinte.

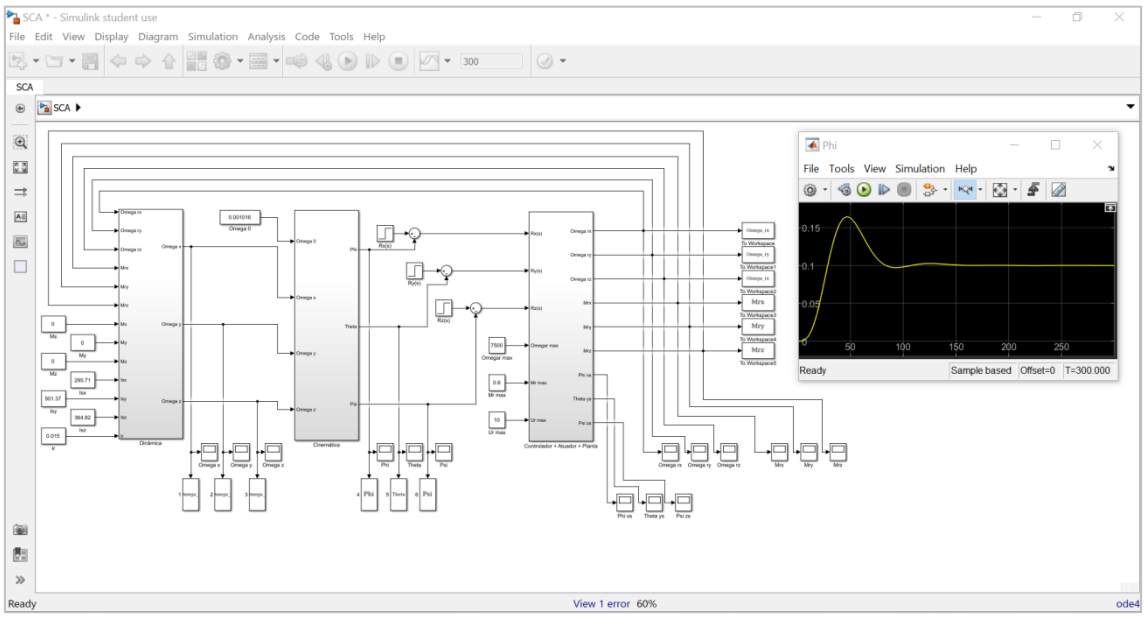
A seguir, o componente “*MATLAB*” é executado. O *script* “*m-file-run.m*” e o modelo Simulink do SCAO são fornecidos pelo usuário como dados de entrada, por meio do componente “*MATLAB Inputs*”. O arquivo de entrada “*m-file-init.m*” é fornecido automaticamente pelo componente PreProc. O componente “*MATLAB*” concatena os dois *scripts*“*m-file-run.m*” e “*m-file-init.m*”, gerando o *script* “*init_run.m*”, e lança a aplicação MATLAB, que executa este *script*.

No modelo Simulink, os valores dos parâmetros definidos no *scriptinit_run.m,* provenientes do VirSat, são atribuídos a variáveis correspondentes do modelo. Dentre estas variáveis, estão dados do satélite e de sua órbita, do controlador e dos atuadores (rodas de reação). São definidos nestes parâmetros os ângulos iniciais de orientação do satélite e os ângulos finais desejados. O modelo Simulink do SCAO contém, além do modelamento do SCAO, os modelos de cinemática e dinâmica do satélite. Ao ser executada a simulação, o comportamento dinâmico do satélite é determinado e é verificado se, a partir dos dados fornecidos, o SCAO foi capaz de orientar o satélite no ângulo desejado, dentro do intervalo de tempo determinado pelo requisito. O arquivo de saída da simulação é o “*MatlabOut.txt*”, que contém os valores das variáveis de saída “*FinalPhi*” (ângulo de rolamento final) e “*TimePhi*” (tempo para atingir o ângulo de rolamento final), em formato texto. A simulação no MATLAB/Simulink é executada pelo RCE em segundo plano, sem mostrar a interface gráfica e sem intervenção do usuário. Entretanto, após a execução da simulação, o usuário pode verificar manualmente seus resultados. A Figura 6.20mostra a tela do Simulink com o resultado de uma das simulações, um gráfico mostrando a evolução do valor de *Phi* (ângulo de rolamento).

A seguir, o componente “*PostProc*” é executado. Este componente tem como arquivos de entrada o arquivo “*MatlabOut.txt*” (fornecido pelo componente “*MATLAB*”), o texto “*UID*” (fornecido pelo componente PreProc) e o arquivo “*UpdateVirSatOutputParams.py*” (fornecido pelo usuário por meio do componente “*PostProc Inputs*”). Ao ser executado, o componente salva no arquivo “*UID.txt*” a identificação da planilha Excel fornecida no início da

execução do *workflow* e executa o *script* “*UpdateVirSatOutputParams.py*”. Este *script* lê a identificação da planilha Excel do arquivo “*UID.txt*”, o nome do estudo do arquivo “*Study_Name.txt*”, os resultados da simulação do arquivo “*MatlabOut.txt*” e atualiza a planilha Excel no repositório local do VirSat, incluindo os valores dos parâmetros de saída calculados.

Figura 6.20 – Resultado de uma simulação no Simulink.



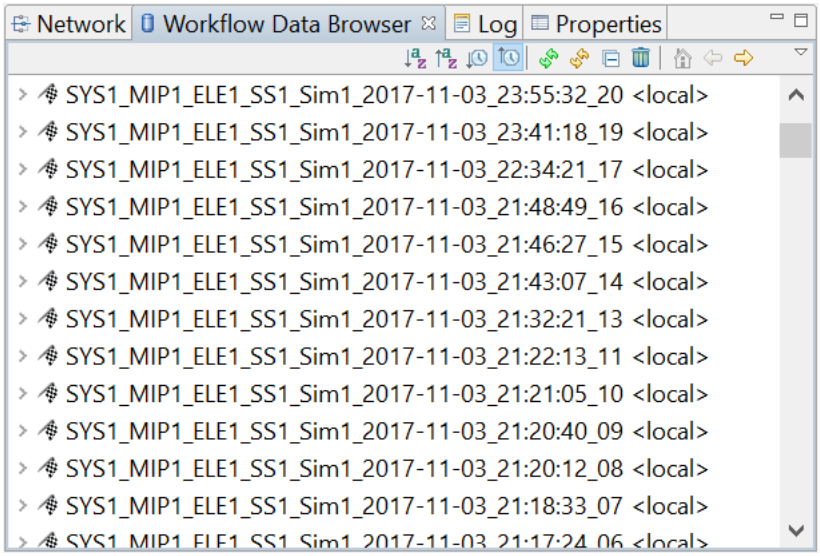
Fonte: Produção do autor.

O engenheiro responsável pelo SCAO abre então o VirSat, atualiza os valores dos parâmetros de saída do SCAO com os valores da planilha e executa o comando “*Commit to Repository*” para atualizar o repositório de dados com os valores atualizados dos parâmetros, disponibilizando-os para os demais engenheiros do time.

Caso o requisito não tenha sido atendido, os engenheiros podem interagir e alterar os valores dos parâmetros do satélite e executar novas simulações, tantas quantas forem necessárias, até que o requisito tenha sido atendido. Quando isto acontecer, uma consulta ao modelo Simulink atualizado com os valores mais recentes dos parâmetros resultará em um gráfico similar ao mostrado na Figura 6.20, mostrando o atendimento aos requisitos.

Ao término da execução de um *workflow*, seus dados e metadados ficam disponíveis a todos os usuários da rede RCE por meio do recurso "*Workflow Data Browser*", mostrado na Figura 6.21.

Figura 6.21 – Janela do *Workflow Data Browser* do RCE.

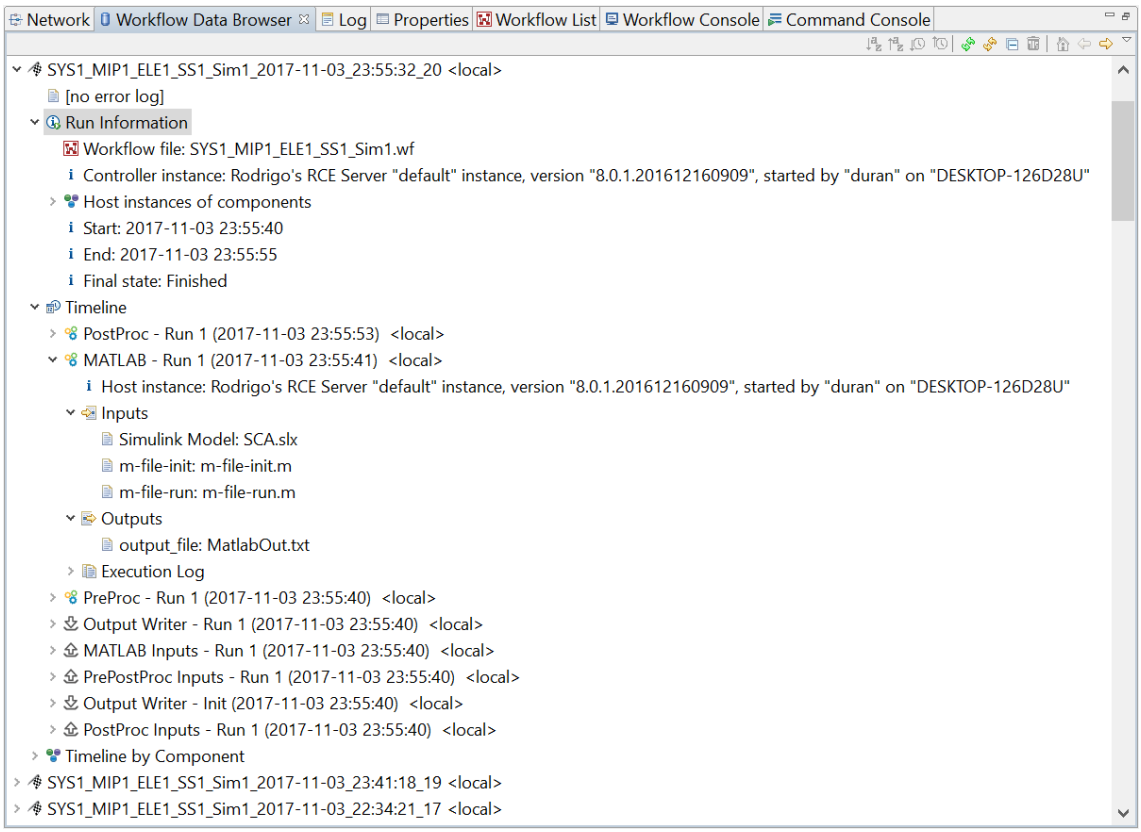


Fonte: Produção do autor.

O "*Workflow Data Browser*" mostra uma lista com os nomes originais dos *workflows* seguidos da data e hora completos de sua execução. Devido a limitações deste recurso, a busca por *workflows* é limitada ao seu nome. Assim, é necessário adotar uma convenção de nomenclatura para facilitar a busca. A convenção adotada é a mesma definida pela ECSS (2010b) para nomenclatura de parâmetros, mostrada na seção 3.6.1. Neste caso, a convenção é adaptada para nomenclatura de *workflows* de simulação.

Por meio do "*Workflow Data Browser*" é possível acessar todos os arquivos envolvidos na execução do *workflow* de simulação e acessar vários metadados como informações do usuário que executou o *workflow*, horário de início e de fim da simulação, etc. A Figura 6.22 mostra um exemplo de visualização de dados de um *workflow* já executado, mostrando o componente "*MATLAB*" e seus dados de entrada e saída.

Figura 6.22 – Visualização detalhada do *Workflow Data Browser* do RCE.



Fonte: Produção do autor.

6.7. **Publicação do *workflow* no VirSat**

Conforme mencionado na seção 2.3, somente linhas de base aprovadas são colocadas em controle de configuração ao final de cada revisão de projeto. Estudos, análises, modelos e demais artefatos não diretamente associados à linha de base aprovada não são formalmente controlados. No ambiente proposto, considera-se o repositório de dados do VirSat como a referência oficial para os dados referentes à fase A do projeto espacial. Assim, ao efetuar-se a revisão de projeto da fase A, os *workflows* do RCE que deram origem aos valores finais dos parâmetros de cada subsistema no VirSat são publicados no repositório de dados do VirSat.

Para isso, é utilizado o recurso de exportação (“*Export*”) do RCE, que gera um arquivo ZIP com toda a estrutura de dados de um *workflow*. Posteriormente, se

for necessário executar novamente este *workflow* e os dados tiverem sido apagados do RCE, é possível realizar uma operação de importação de dados, fornecendo o arquivo ZIP como entrada, o que resulta na restauração de todos os dados do *workflow* e na possibilidade de executá-lo novamente, seja para verificação de valores de parâmetros ou para cálculo de valores que não haviam sido simulados antes. A Figura 6.23 mostra um exemplo deste recurso, na geração de um arquivo ZIP para a simulação do subsistema AOCS do satélite Amazônia-1.

Por convenção, o arquivo ZIP deve ser nomeado com o mesmo nome da execução do *workflow* publicado no VirSat. A Figura 6.25 mostra um exemplo deste nome, que é composto do nome do *workflow* seguido da data e hora de execução. Um arquivo com este nome é gerado automaticamente, por conveniência, no diretório do projeto, a cada execução do *workflow*, pelo componente "*Workflow ID Writer*".

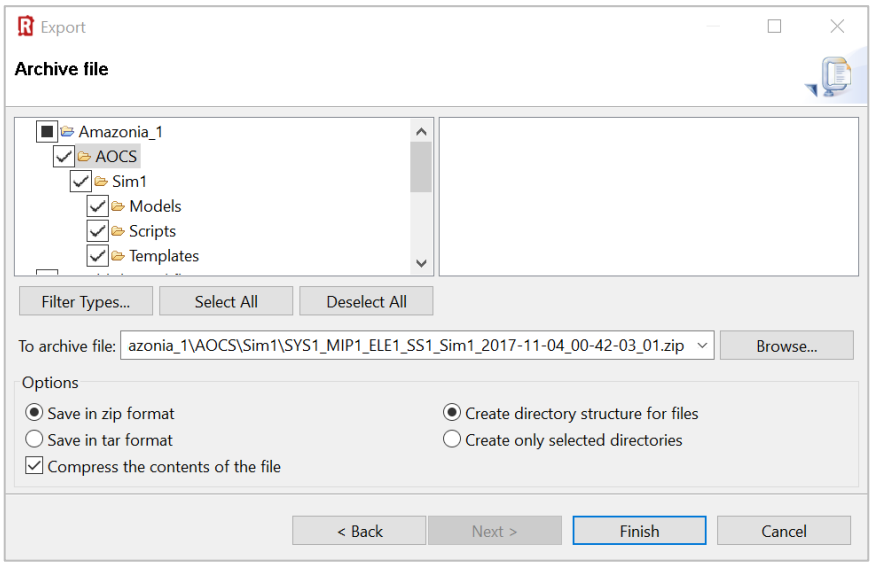
Para a publicação do arquivo ZIP no repositório do VirSat, é utilizado um novo *workflow* do RCE desenvolvido para esta finalidade, mostrado na Figura 6.24. Este *workflow* simples é composto de dois componentes nativos do RCE: "*Input Provider"* e "*Script"* (instanciado com o nome de "*Publish*"). O componente "*Input Provider"* fornece ao componente "*Publish*" três entradas: o arquivo ZIP a ser publicado ("*Archive_File*"), a identificação do subsistema ("*UUID*") a que o arquivo ZIP se refere e o nome do estudo no VirSat ("*Study_Name*").

Ao se executar o *workflow*, o arquivo ZIP é copiado do diretório de projeto do RCE para o repositório de dados do estudo no VirSat.

A planilha Excel que contém os parâmetros do subsistema AOCS no VirSat possui a identificação de qual execução do *workflow* que deu origem aos valores dos parâmetros de saída. Por conveniência, o componente "*PostProc*" também adiciona um *hyperlink* nesta identificação, de modo que, ao clicar nela, é aberta uma janela do *Windows Explorer* mostrando o diretório, dentro do repositório VirSat, que contém o arquivo ZIP. Isto é ilustrado na Figura 6.26,

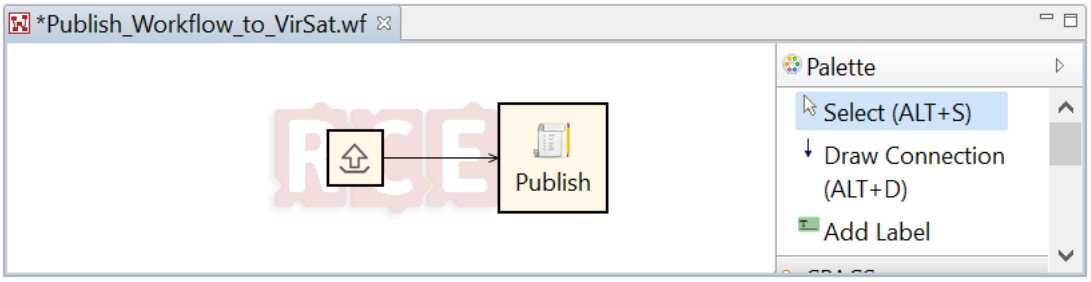
onde a célula “E1” da planilha contém a identificação do *workflow* e, ao clicar na célula, é aberta outra janela com o caminho completo até o arquivo ZIP correspondente.

Figura 6.23 – Exportação de *workflows* para arquivos ZIP no RCE.



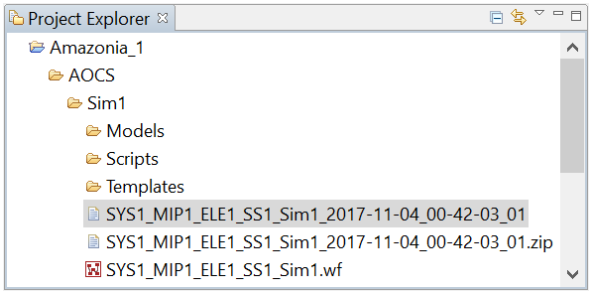
Fonte: Produção do autor.

Figura 6.24 – *Workflow* RCE para publicação de dados no VirSat.



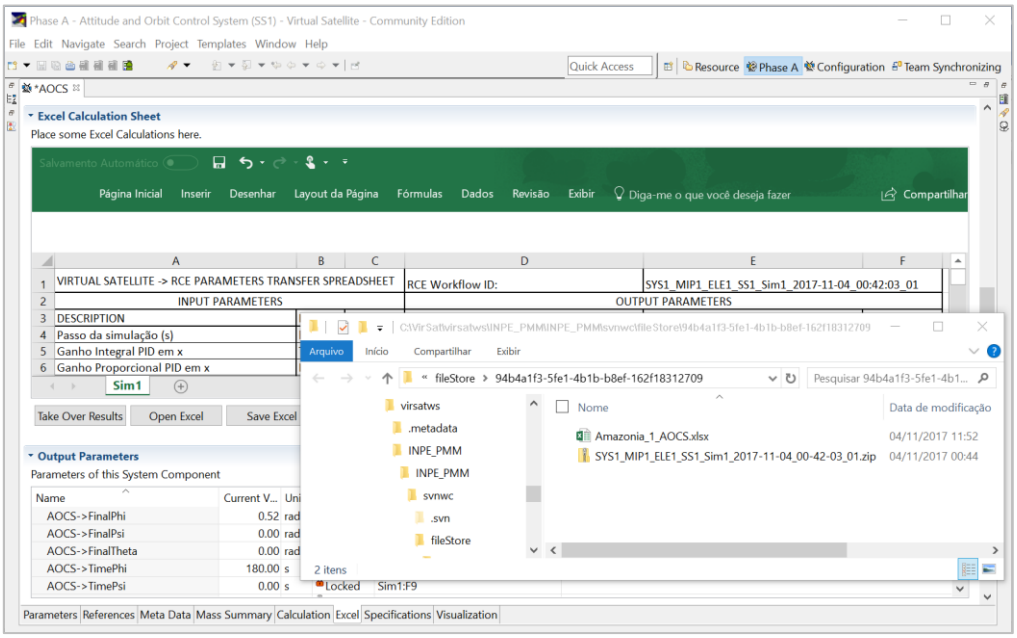
Fonte: Produção do autor.

Figura 6.25 – *Project Explorer* do RCE com a identificação da execução da simulação.



Fonte: Produção do autor.

Figura 6.26 – Identificação do *workflow* RCE no VirSat.



Fonte: Produção do autor.

Durante a fase A, enquanto as simulações estiverem sendo executadas e o arquivo ZIP ainda não tiver sido publicado, qualquer engenheiro envolvido no estudo pode ter acesso aos dados da simulação por meio da rede RCE. Para isto, basta acessar o "*Workflow Data Browser*" do RCE e realizar uma busca pela identificação da execução do *workflow,* a mesma mostrada na planilha.

6.8. **Resultados do estudo de caso espacial**

Por meio da execução do estudo de caso espacial, foi possível avaliar o ambiente de gerenciamento de dados e processos de simulação proposto. O controle de acesso a usuários no VirSat restringe a atualização de dados dos subsistemas apenas aos seus responsáveis. Para não prejudicar a característica colaborativa do ambiente, não foi configurado o controle de acesso aos dados dos *workflows* no RCE, embora seja possível restringir o acesso a componentes específicos.

O controle de versão dos dados publicados no VirSat é feito por meio do servidor SVN. Cada *upload* ou *commit* de dados neste servidor cria uma nova

revisão dos dados. Caso seja necessário recuperar uma versão antiga, basta configurar o servidor para voltar à versão desejada e atualizar o repositório local.

A evolução dos dados no RCE é controlada por meio do *Workflow Data Browser*, que guarda todas as versões dos dados utilizados em execuções de *workflows* de simulação no banco de dados do RCE. Quando uma simulação é executada, são utilizados dados localizados no *Workspace*, uma área local no computador do usuário, não controlada por sistemas de controle de versão. Entretanto, todos os dados utilizados são copiados para o banco de dados do RCE e disponibilizados para visualização no *Workflow Data Browser.* Caso os dados do *Workspace* sejam modificados para a execução de uma nova simulação, e seja necessário consultar os dados antigos utilizados na execução anterior, basta consultar o *Workflow Data Browser* para visualizar os dados utilizados em execuções anteriores do mesmo *workflow*.

Durante o processo de execução das simulações, metadados como o autor da simulação, componente analisado, data, hora e máquina onde a simulação foi executada, são automaticamente capturados pelo ambiente e ficam disponíveis para consulta posterior.

O RCE, por meio do *Workflow Data Browser*, permite que se verifique as entradas e saídas de cada componente do *workflow*, assim como a ordem em que foram executados. Isto permite que se obtenha um *pedigree* de dados, facilitando a rastreabilidade futura. No VirSat, na visualização de cada componente, existe a aba “*References*” onde é possível visualizar quais outros componentes referenciaram parâmetros do componente em questão.

A aderência do RCE ao modelo de dados proposto para o ambiente se traduz na leitura e escrita de valores dos parâmetros de componentes dos subsistemas, que são objetos deste modelo. Mesmo o RCE não sendo baseado no mesmo modelo de dados, foi possível criar um adaptador que interage com o modelo de dados proposto.

No ambiente proposto, o VirSat cumpre tanto a função de gerenciar a estrutura de dados do sistema quando armazenar e gerenciar uma parte de seus dados. Embora as normas espaciais não indiquem a necessidade de um software para gerenciar o modelo de dados (ECSS, 2011f), notou-se que é interessante a utilização de um software com esta função. Este software não necessariamente precisaria realizar a função adicional de armazenar e controlar a versão dos dados do sistema, mas poderia conter *links* entre cada objeto e os diferentes sistemas de SPDM responsáveis pelo seu gerenciamento, de forma a centralizar a visualização e compartilhamento de dados do sistema.

O ambiente apresentou também algumas deficiências, como a capacidade de busca limitada do RCE e VirSat, que permite apenas a busca por palavras-chave contidas nos arquivos armazenados em seus bancos de dados locais. O *Workflow Data Browser* do RCE permite a busca pelo nome de *workflows* em toda a rede RCE, não limitado ao computador do usuário, entretanto esta funcionalidade não é eficiente para localizar dados específicos gerados por outros usuários.

Por fim, a disponibilização de *workflows* no RCE e a possibilidade de referenciá-los em componentes do VirSat amplia a capacidade de cálculo de parâmetros do sistema e alavanca as fases iniciais de projeto de sistemas espaciais com a introdução de simulações e dos recursos para gerenciá-las, evidenciando assim os ganhos obtidos com a utilização do ambiente.

7 ESTUDO DE CASO AERONÁUTICO: IMPLEMENTAÇÃO E AVALIAÇÃO

Neste capítulo, são apresentadas as oportunidades de aplicação das normas da ECSS na área aeronáutica, identificadas durante a condução deste trabalho. É apresentado o estudo de caso aeronáutico, que consiste na aplicação de conceitos das normas ECSS na especificação de um ambiente de gerenciamento de dados e processos de simulação.

7.1. **Introdução**

Os processos de desenvolvimento de aeronaves geram uma grande quantidade de dados, não apenas pela área de engenharia, mas também por áreas como manufatura, testes de voo e publicações técnicas. Estes dados são gerados por diversas aplicações de software que leem e escrevem dados em formatos e tipos diferentes. É um grande desafio organizar todas essas informações para que qualquer pessoa possa procurar os dados certos e recuperá-los de forma eficiente.

Conforme visto na seção 3.5, um modelo de dados conceitual pode ser utilizado como ponte entre diversas aplicações de software desenvolvidas com base em modelos de dados diferentes, fornecendo uma referência comum para troca de dados. Isto é possível por meio da extensão destas aplicações, com a criação de adaptadores de interface compatíveis com um mesmo modelo de dados conceitual.

Neste estudo de caso aeronáutico proposto, é analisado como a introdução dos conceitos propostos pela ECSS (2011f) durante as fases de desenvolvimento em projetos de aeronaves poderia beneficiar os processos de gerenciamento de dados e processos de engenharia.

7.2. **Normas de gerenciamento de dados aeronáuticos**

Assim como a ECSS, órgãos regulamentadores aeronáuticos como o *Federal Aviation Administration* (FAA) não possuem documentos normativos que especifiquem um determinado processo de gerenciamento de dados e processos aplicado ao desenvolvimento de produtos. Entretanto, ambos possuem documentos não-normativos que tratam deste assunto. Enquanto a ECSS tem memorandos técnicos, o FAA tem as *Advisory Circulars (*AC*),* onde são propostos meios aceitáveis para o cumprimento de normas de aeronavegabilidade, ou demonstração de conformidade com as mesmas. Geralmente de natureza informativa, as ACs não são vinculativas nem reguladoras (APPROACH, 1985).

Existem apenas duas ACs que tratam dos dados gerados durante o desenvolvimento de produtos aeronáuticos: a AC 21-48: *Using Electronic Modeling Systems as Primary Type Design Data* (FAA, 2010) e a AC 20-179: *Certification Data Retention Agreements and Government Records* (FAA, 2013). Enquanto que os memorandos técnicos da ECSS como o ECSS-E-TM-10-23A: *Space Engineering – Space System Data Repository* (ECSS, 2011f) apresentam recomendações detalhadas de como configurar um repositório de dados de sistemas espaciais, as ACs apresentam apenas orientações básicas. A AC 21-48 contém apenas um conjunto de questões que devem ser respondidas pelo desenvolvedor do produto aeronáutico, caso sejam usados sistemas eletrônicos para o armazenamento de dados de certificação de produto. Para cada questão, o FAA lista os requisitos mínimos para a resposta esperada do desenvolvedor, sem, no entanto, indicar os meios de cumprimento a serem utilizados. A AC 20-179 contém orientações de como armazenar dados de certificação, caso o desenvolvedor do produto aeronáutico assine um acordo de retenção de dados com o FAA.

As questões levantadas pelo FAA na AC 21-48 são mostradas a seguir:

a) como a integridade dos dados eletrônicos de projeto de tipo será garantida ao longo do ciclo de vida do produto?

b) como os usuários irão acessar os dados eletrônicos de projeto de tipo?

c) como o usuário determinará o *status* de aprovação do dado eletrônico de projeto de tipo utilizado para certificação?

d) como será estabelecida a configuração do produto final?

e) como o sistema de modelamento 3D irá garantir que os dados aprovados pelo FAA e os dados liberados para fabricação e inspeção são facilmente distinguíveis uns dos outros?

f) como os dados eletrônicos de projeto de tipo serão repassados para o FAA caso o certificado de aeronavegabilidade seja suspenso ou o seu proprietário interrompa suas operações?

g) como os dados eletrônicos de projeto de tipo serão repassados para fornecedores e outros usuários externos?

h) como os dados eletrônicos de projeto de tipo serão usados para suportar a aeronavegabilidade continuada?

i) como o FAA, o *National Transportation Safety Board* (NTSB), e outras agências reguladoras terão acesso aos dados eletrônicos de projeto de tipo?

j) como as características dos dados eletrônicos de projeto de tipo correspondem ao formato original em papel?

k) como os usuários serão treinados a usar o sistema de modelamento em 3D?

l) que atributos são necessários nos arquivos de dados?

As normas do FAA, denominadas *Federal Aviation Regulations* (FAR), tiveram origem década de 1960 (FAA, 2018), quando o uso de computadores não era disseminado como nos dias de hoje e os dados de certificação eram primariamente disponibilizados ao FAA em forma de relatórios em papel. Atualmente, embora as normas sejam ainda as mesmas, os dados, em sua maioria, residem em sistemas computacionais. O propósito das duas ACs mencionadas é prover orientações de como utilizar dados em formato eletrônico na certificação de aeronaves.

No mundo aeronáutico, o foco dos órgãos regulamentadores é na garantia do acesso aos dados de certificação da aeronave durante toda vida útil da mesma. Assim, em casos como a investigação de acidentes aéreos, para determinação de responsabilidade civil, os órgãos regulamentadores têm a garantia de acesso aos dados de projeto para determinar se a origem do acidente foi devida a alguma falha de projeto.

7.3. **Sinergias identificadas entre as normas espaciais e aeronáuticas**

Após a análise das normas da ECSS e do FAA, nota-se que existe um grande potencial de aplicação das normas espaciais em casos de uso aeronáuticos, especialmente o memorando técnico ECSS-E-TM-10-23A: *Space Engineering – Space System Data Repository* (ECSS, 2011f). Conforme visto na seção 3.5.2, o memorando estabelece um conjunto de passos para o estabelecimento de um Repositório de Dados de Sistemas Espaciais para um projeto espacial. Estes passos podem ser adaptados para o estabelecimento de um “Repositório de Dados de Sistemas Aeronáuticos” equivalente, para um projeto aeronáutico, da seguinte forma:

a) atribuir o papel de *autoridade de projeto* para um Repositório de Dados de Sistemas Aeronáuticos a ser criado. Assim, qualquer aplicação de software que venha a se integrar ao repositório, tanto consumindo quanto fornecendo dados, deverá se adequar às regras

do repositório, que se torna a referência central para todas as aplicações;

b) definir o escopo do Repositório de Dados de Sistemas Aeronáuticos para o projeto, definindo o que será envolvido, por exemplo: quais disciplinas de engenharia, quais fases do ciclo de vida, quais partes da cadeia cliente-fornecedor, etc;

c) definir o modelo de dados conceitual do projeto. Este modelo de dados deve incluir a representação de todos os dados definidos no escopo do repositório;

d) definir a arquitetura do Repositório de Dados de Sistemas Aeronáuticos, definindo, por exemplo: quais aplicações e repositórios de dados, quais interfaces de troca de dados entre repositórios, qual a abordagem para controle de configuração de dados, etc;

e) derivar o modelo de dados conceitual “local” de cada aplicação a partir do modelo de dados conceitual global do projeto;

f) implementar e manter a infraestrutura do Repositório de Dados de Sistemas Aeronáuticos.

Geralmente, softwares comerciais de gerenciamento de dados possuem um modelo de dados fixo, com regras definidas, que não permite mudanças. Desta forma, ao se deparar com a necessidade de criar um modelo de dados conceitual de uma aeronave, o seu desenvolvedor pode optar por criar um modelo de dados compatível com o modelo de dados oferecido por uma ferramenta comercial, ou pode investir no desenvolvimento de sua própria ferramenta de gerenciamento de dados, que permitiria uma liberdade maior na definição de um modelo de dados ideal. Neste caso, poderia ainda reusar um software *framework* existente, ao invés de iniciar um desenvolvimento completamente novo. O que se observa na prática, contudo, é a adequação do

modelo de dados da aeronave ao modelo de dados específico da ferramenta comercial, pois empresas aeronáuticas tendem a focar no desenvolvimento de seus produtos, e não no desenvolvimento de software não diretamente relacionado a eles.

É importante também mapear todos os interessados no projeto de desenvolvimento de uma nova aeronave e as trocas de dados previstas entre eles, para que o modelo de dados possa distinguir dados do desenvolvedor da aeronave e de seus fornecedores, por exemplo.

7.4. **Adaptação do ambiente para o estudo de caso aeronáutico**

Conforme visto na seção 5, o ambiente de gerenciamento de dados e processos de simulação proposto envolve uma aplicação específica para o desenvolvimento de satélites, o VirSat, que usa um modelo de dados compatível com o SEIM definido no memorando técnico ECSS-E-TM-10-25A (ECSS, 2010b). Assim, o ambiente proposto não é diretamente aplicável na área aeronáutica. Entretanto, os mesmos conceitos utilizados em sua elaboração podem ser reusados na configuração de ambientes de gerenciamento de dados e processos na área aeronáutica.

O conceito-chave na definição do ambiente proposto é a utilização de um modelo de dados conceitual que serve como referência comum para todas as aplicações que fazem parte do repositório de dados de sistemas espaciais. Mesmo que estas aplicações não tenham sido desenvolvidas com base no mesmo modelo de dados, é possível desenvolver adaptadores de interface compatíveis com o modelo de dados conceitual.

No intuito de simular como seria a adaptação do ambiente a um modelo de dados mais rígido, geralmente encontrado em softwares comerciais de gerenciamento de dados de engenharia, propõe-se a utilização do CPACS, um modelo de dados desenvolvido pelo DLR para a comunicação entre diferentes

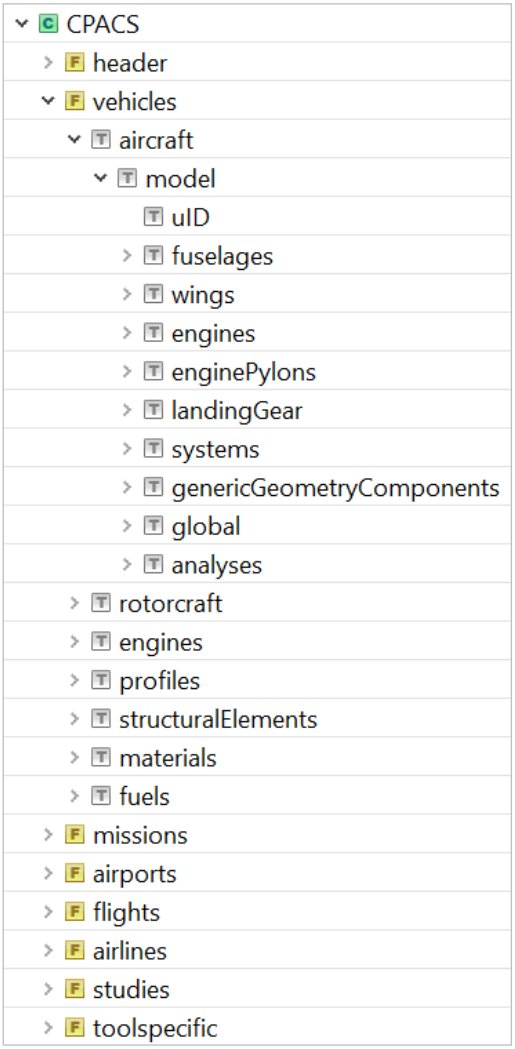
disciplinas durante o desenvolvimento de aeronaves, já mencionado na seção 4.3.

O CPACS é um modelo paramétrico de uma aeronave completa, incluindo uma estrutura hierárquica de componentes que pode ser usada para representar não apenas dados geométricos da aeronave como também dados de processo, como parâmetros a serem usados em simulações. Estes parâmetros incluem, por exemplo, dados de configuração para geração de malha de elementos finitos a partir da geometria, *links* para localização de dados de apoio em repositórios de dados, etc. O CPACS foi concebido como um modelo de dados enxuto para apoiar as fases iniciais de concepção da aeronave, assim como estudos de otimização multidisciplinar (NAGEL et al., 2012).

A Figura 7.1 mostra os primeiros níveis da hierarquia de componentes do CPACS. Neste exemplo, é mostrado o componente *aircraft* se desdobrando em seus componentes, como *fuselages*, *wings* e *engines*. No CPACS, é possível definir um sistema de coordenadas global e vários sistemas de coordenadas locais. Em uma montagem, um componente que aparece várias vezes no modelo não precisa ser criado múltiplas vezes. É possível criar apenas um componente central de referência e várias instâncias deste componente, por meio de transformações entre sistemas de coordenadas. Para isto, é importante que cada componente receba uma identificação única (UID), que é usada nestas referências.

Os componentes mostrados na Figura 7.1 desdobram-se ainda em vários outros níveis, cada vez mais detalhados. É possível representar, por exemplo, detalhes da estrutura da asa como as nervuras (*ribs*), mostradas na Figura 7.2. Por ser capaz de representar uma aeronave neste nível de detalhe, o CPACS configura-se como um promissor padrão de interface entre ferramentas utilizadas no desenvolvimento de aeronaves.

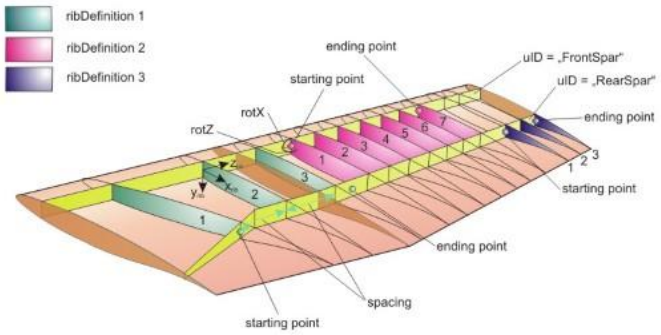
Figura 7.1 – Exemplo da estrutura de dados do CPACS.



Fonte: Produção do autor.

Figura 7.2 – Exemplo de representação de nervuras de asa no CPACS.

	ribDefinition 1	ribDefinition 2	ribDefinition 3
ribReference	RearSpar	FrontSpar	RearSpar
etaStart	0.2	0.45	0.9
etaEnd	0.4	0.8	1.0
ribStart	FrontSpar	FrontSpar	RearSpar
ribEnd	RearSpar	RearSpar	trailingEdge
ribRotationReference	RearSpar	FrontSpar	
x-Rotation	90	90	
z-Rotation	90	90	
spacing	2		
numberOfRibs		7	3
ribCrossingBehavior	cross	cross	cross



Fonte: CPACS, 2017.

O CPACS é implementado como uma “definição de esquema” (*Schema Definition*) da linguagem XML, no formato de um arquivo com a extensão XSD (cpacs_schema.xsd). Tais definições são usadas para definir os elementos, atributos, tipos de dados e regras que um arquivo XML baseado neste esquema deve cumprir (W3C, 2018).

Aplicações que utilizem o CPACS como padrão para troca de dados devem ser capazes de ler e escrever arquivos XML compatíveis com a definição de esquema do CPACS. Por exemplo, na definição de nervuras, a Figura 7.2 mostra uma tabela com as *tags* do arquivo XML necessárias para a descrição das mesmas. Por exemplo, a *tag* “*ribStart*” da RibDefinition 1 apareceria com o seguinte formato no arquivo XML do CPACS:

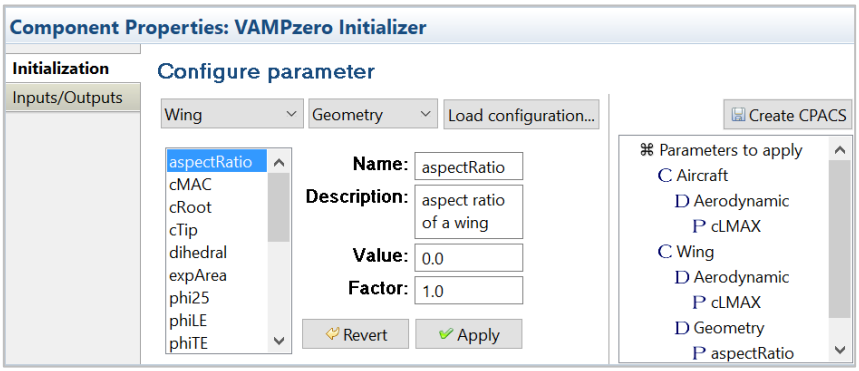
<ribStart externalDataDirectory="" externalDataNodePath="" externalFileName="" xsi:type="stringBaseType">**FrontSpar** </ribStart>

O RCE possui os seguintes componentes dedicados ao tratamento de arquivos XML compatíveis com o esquema de dados do CPACS:

a) ***CPACS Writer***: a partir de um arquivo XML no padrão CPACS fornecido como entrada, permite mapear valores específicos de elementos (como o “*ribStart*” mostrado acima) e substituí-los por outros valores também fornecidos como entrada. Da mesma forma, é possível extrair valores de elementos mapeados e escrevê-los em um arquivo texto;

b) ***VAMPzero Initializer***: escreve um arquivo CPACS para ser utilizado pela ferramenta de projeto conceitual de aeronaves VAMPzero (um acrônimo para ***V****irtual* ***A****ircraft* ***M****ultidisciplinary Analysis and Design* ***P****rocesses*). Possui uma interface gráfica amigável onde é possível navegar por vários elementos do esquema CPACS, determinar seus valores e escreve um arquivo XML correspondente. A Figura 7.3 mostra um exemplo da interface do VAMPzero Initializer;

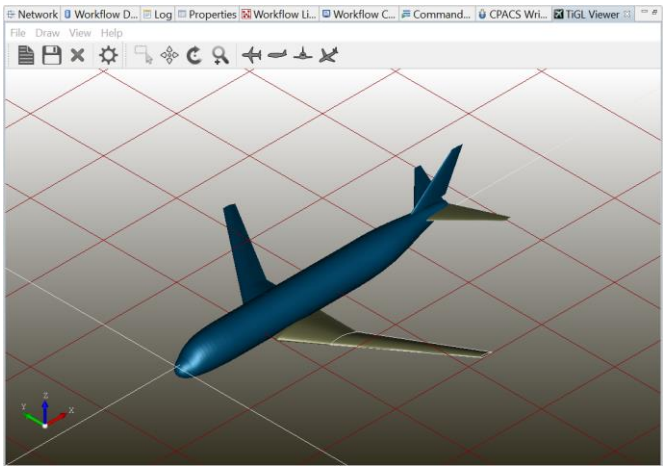
Figura 7.3 – Interface do *VAMPzero initializer*.



Fonte: Produção do autor.

c) ***TiGL Viewer***: permite visualizar graficamente, em 3D, um arquivo XML no padrão CPACS. A Figura 7.4 mostra um exemplo da interface do TiGL Viewer;

Figura 7.4 – Interface do *TiGL Viewer*.



Fonte: Produção do autor.

d) ***XML Loader***: a partir de um modelo de arquivo XML fornecido como entrada, permite mapear valores específicos de elementos, substituí-los por outros valores também fornecidos como entrada e escrever o novo arquivo XML resultante;

e) ***XML Merger***: permite a fusão de dois arquivos XML, um de referência e outro com dados a serem integrados ao de referência, com base em

um terceiro arquivo, também fornecido, que contém o mapeamento entre os dois primeiros.

Desta forma, propõe-se que o ambiente para o estudo de caso aeronáutico utilize apenas o RCE e aplicações de software integradas pelo usuário na forma de componentes do RCE, todos adotando o CPACS como um modelo de dados físico.

7.5. **Definição do estudo de caso aeronáutico**

No estudo de caso aeronáutico proposto, uma configuração preliminar de aeronave comercial é criada durante a fase de estudos conceituais e disponibilizada como referência para engenheiros responsáveis por diversas disciplinas realizarem suas análises. Uma destas análises consiste na simulação de decolagem da aeronave para determinação do tempo total de decolagem e comprimento de pista necessários para o cumprimento das normas de desempenho do FAR Part 25, incluindo a §25.113, "*Takeoff distance and takeoff run*". Esta norma estabelece, entre outros requisitos, que a distância de decolagem é a diferença de distância horizontal entre a aeronave parada e o ponto onde ela se encontra a uma altura de 35 pés. No trabalho de Lynn (1994), é mostrada uma formulação para a determinação da distância e tempo de decolagem, entre outros parâmetros. Um código na linguagem Fortran é disponibilizado nesta referência para a execução das simulações, que têm como entrada os parâmetros mostrados na Tabela 7.1.

Tabela 7.1 – Parâmetros da simulação de decolagem.

Parâmetro	Descrição
D	Densidade do ar durante a decolagem
W	Peso da aeronave
A	Área da asa da aeronave
CLmax	Coeficiente de sustentação máximo da aeronave
CLgrd	Coeficiente de sustentação para o segmento de corrida em solo
CLair	Coeficiente de sustentação para o segmento de subida no ar

CDgrd	Coeficiente de arrasto para o segmento de corrida em solo
CDair	Coeficiente de arrasto para o segmento de subida no ar
MUgrd	Coeficiente de fricção de rolagem
MUbrk	Coeficiente de fricção de frenagem
LAMBDA	Ângulo de deflexão do propulsor
K	Margem de *stall*
TEB	Tempo entre falha de motor e frenagem
OBSHT	Altura do obstáculo
PLOSS	Fração de potência disponível em caso de falha no motor
T3	Três valores de potência para determinação de curva potência/velocidade dos motores
V3	Três valores de velocidade para determinação da curva potência/velocidade dos motores
TSTEP	Intervalo de tempo para escrita dos dados de saída
TROT	Tempo requerido para rotação da aeronave

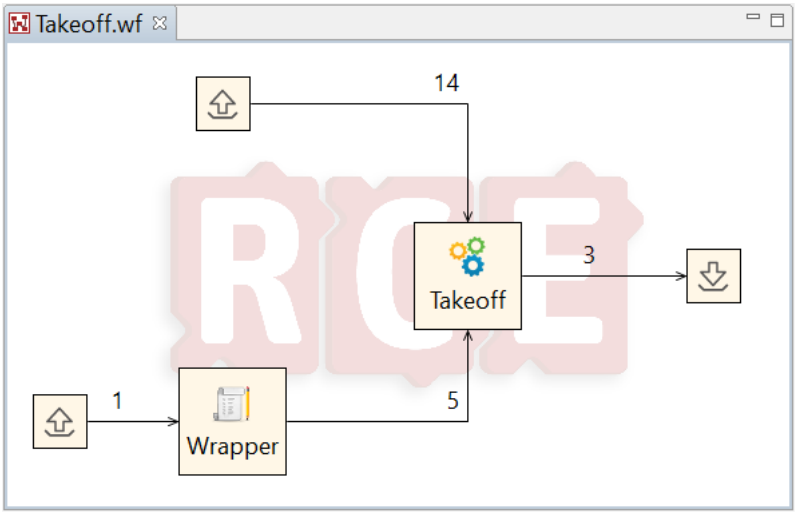
Fonte: Adaptado de Lynn (1994). (tradução nossa).

No estudo de caso aeronáutico proposto, após a publicação no RCE do arquivo CPACS com as definições iniciais da aeronave, este é compartilhado com os demais engenheiros envolvidos no estudo. Cada engenheiro é responsável por uma disciplina de projeto específica e realiza simulações por meio de *workflows* no RCE, a partir de dados de entrada que incluem os extraídos do arquivo CPACS de referência. Neste trabalho, é exemplificado o *workflow* de simulação de decolagem da aeronave. Ao término da primeira rodada do estudo, cada engenheiro salva seus resultados em um arquivo CPACS contendo apenas um subconjunto do arquivo original, e em seguida todos são fundidos pelo engenheiro responsável pelo estudo em um arquivo único, que se torna a nova referência para a próxima rodada do estudo. Nesta fase, cada engenheiro analisa as possíveis mudanças em seus dados de entrada, devidas às análises dos demais engenheiros, e programa as próximas análises para levar em consideração estas mudanças. Este processo se repete até que os requisitos de projeto da aeronave tenham sido atingidos.

7.6. **Configuração da simulação de decolagem no RCE**

Para a simulação da decolagem no RCE, foi criado o *workflow Takeoff* mostrado na Figura 7.5. O *workflow* é composto do componente principal, também denominado *Takeoff*, criado especificamente para esta simulação. Este componente executa o programa "*takeoff.exe*" disponibilizado por Lynn (1994), que tem como entrada um arquivo texto com dezenove valores de parâmetros do modelo de decolagem e como saída um arquivo texto com os valores de distância e tempo de decolagem.

Figura 7.5 – *Workflow* de simulação de decolagem.



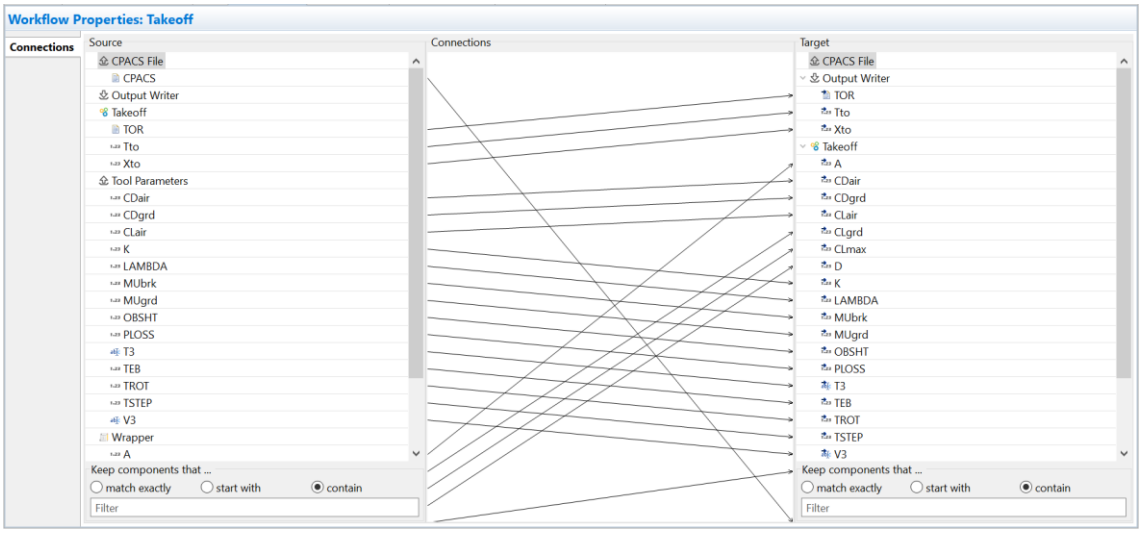
Fonte: Produção do autor

Dos parâmetros de entrada, cinco estão presentes no arquivo XML no padrão CPACS disponibilizado aos engenheiros e as demais são específicas da simulação de decolagem. Assim, foi criado um componente *Wrapper* que extrai os dados do arquivo no padrão CPACS e os disponibiliza para a escrita de um arquivo no formato requerido pelo programa "*takeoff.exe*".

As quatorze entradas adicionais são fornecidas pelo componente *Tool parameters* (do tipo *Input Provider*). A Figura 7.6 mostra as conexões detalhadas entre os componentes do *workflow*.

O componente *Takeoff* possui um *script* de pré-execução que lê todas as variáveis de entrada e gera um arquivo no formato requerido pelo programa "*takeoff.exe*". Após a execução do programa, um *script* de pós-execução lê o arquivo de saída e extrai os valores da distância e tempo de decolagem, associando-os, respectivamente, às variáveis de saída Xto e Tto. A Figura 7.7mostra a janela do *Workflow Data Browser* do RCE com os dados de entrada e saída do componente *Takeoff*.

Figura 7.6 – Conexões entre componentes do *workflow Takeoff.*

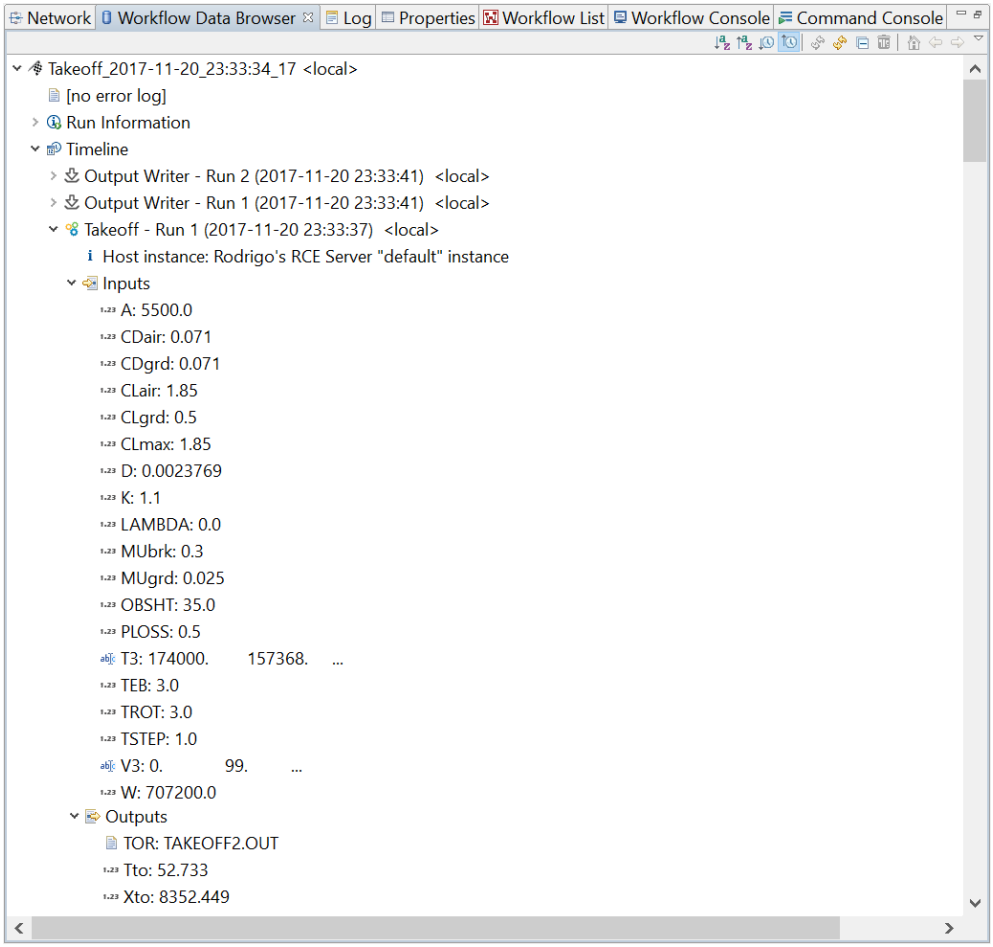


Fonte: Produção do autor.

7.7. **Resultados do estudo de caso aeronáutico**

Por meio da execução do estudo de caso aeronáutico, foi possível avaliar a adaptação do ambiente proposto a um novo modelo de dados. Nota-se que, embora o RCE tenha funcionalidades para processar arquivos no formato do CPACS, não existem as funcionalidades de criação e gerenciamento da estrutura de dados, que precisam ser realizadas fora do RCE. Embora o componente *VAMPzero Initializer* seja capaz de gerar arquivos no formato CPACS, não é possível criar todos os componentes previstos no modelo, nem editá-los posteriormente.

Figura 7.7 – *Workflow Data Browser* com dados de saída do *workflow Takeoff.*



Fonte: Produção do autor.

Assim como mostrado na seção 6.8, nos resultados do estudo de caso espacial, a evolução dos dados utilizados nos *workflows* no RCE é controlada por meio do *Workflow Data Browser*, que guarda todas as versões dos dados no banco de dados do RCE. Ao se executar as simulações, seus metadados são automaticamente capturados pelo ambiente e ficam disponíveis para consulta posterior. Utilizando-se o *Workflow Data Browser*, é possível verificar as entradas e saídas de cada componente do *workflow*, assim como a ordem em que foram executados, permitindo assim a obtenção de um *pedigree* de dados, facilitando a rastreabilidade futura.

Dado que o modelo de dados proposto é um modelo físico com regras bem definidas, todos os componentes que forem desenvolvidos para serem executados neste ambiente têm de ser capazes de, pelo menos, ler dados no formato CPACS e, se necessário, também escrever resultados de saída neste formato.

Para a criação de um modelo de dados consistente, é importante a consideração de todas as fases do projeto de desenvolvimento de uma nova aeronave. Se o modelo é criado para atender apenas às necessidades de fases iniciais, ele se torna incapaz de representar outros tipos de dados e metadados provenientes de fases posteriores, e torna-se necessário revisar o modelo de dados.

No caso de cada fase de desenvolvimento da aeronave ter um modelo de dados diferente, seus repositórios de dados serão diferentes também, o que onera a rastreabilidade dos dados pois, neste caso, é necessário realizar buscas em vários repositórios diferentes. Assim, a abordagem mais eficiente é criar, no início do projeto, um modelo de dados que inclua todas as fases de desenvolvimento do produto.

Sendo o CPACS um modelo de dados concebido para as fases iniciais de projeto de aeronaves e para auxiliar estudos de otimização multidisciplinar, sua utilização em outras fases do ciclo de vida do projeto de aeronaves estaria condicionado à evolução do modelo para a inclusão de outros metadados típicos de fases posteriores do projeto.

Os principais ganhos percebidos na utilização do ambiente no caso de uso aeronáutico foram a obtenção da rastreabilidade dos dados utilizados nos *workflows* de simulação e a facilidade de troca de dados entre aplicações de software por meio de um modelo de dados padrão como o CPACS.

8 CONCLUSÕES, LIÇÕES APRENDIDAS E TRABALHOS FUTUROS

O ambiente proposto neste trabalho, com a integração das aplicações RCE e VirSat, é uma alternativa viável para o gerenciamento de dados e processos de simulação nas fases iniciais de desenvolvimento de um sistema espacial no INPE, pois demonstra o atendimento aos requisitos mostrados na seção 4.5, conforme mostrado a seguir:

a) **O ambiente deve permitir a distribuição de aplicações de software geradas por um usuário para os demais usuários:** conforme mostrado nas seções 6.5.2.1, 6.5.2.2 e 7.6, foram criados componentes de software apropriados para o contexto das simulações necessárias e disponibilizados na rede RCE para os demais usuários;

b) **O ambiente deve permitir o reuso de aplicações de software geradas por um usuário pelos demais usuários:** conforme mostrado nas seções 6.5.2.3 e 7.6, os componentes de software disponibilizados na rede RCE foram reusados para a construção de *workflows* de simulação;

c) **O ambiente deve permitir a construção de *workflows* de simulação que contenham o encadeamento de aplicações de software desenvolvidas por seus usuários:** conforme mostrado nas seções 6.5.2.3 e 7.6, *workflows* de simulação foram criados por meio do encadeamento dos componentes de software disponibilizados na rede RCE;

d) **O ambiente deve permitir o armazenamento de dados gerados em *workflows* de simulação em repositórios de dados acessíveis a todos os seus usuários:** conforme mostrado nas seções6.6, 6.7 e 7.7, dados usados em *workflows* de simulação ficam disponíveis no banco de dados da rede RCE, acessíveis a todos os seus usuários

por meio do *Workflow Data Browser*. Adicionalmente, parte destes dados são também arquivados no repositório do VirSat para referência futura;

e) **O ambiente deve permitir que um usuário consulte dados de *workflows* de simulação executadas por qualquer usuário:** por meio do *Workflow Data Browser*, qualquer usuário pode pesquisar dados de *workflows* de simulação executados na rede RCE, desde que tenha a identificação correta do *workflow* para a realização da busca. Esta identificação é guardada na planilha utilizada como interface entre o RCE e o VirSat;

f) **O ambiente deve permitir o modelamento da hierarquia de subsistemas e componentes de um sistema:** conforme mostrado na seção 6.5.1, é possível modelar a hierarquia de subsistemas e componentes de um sistema por meio do VirSat, que segue o modelo de dados conceitual mostrado na Figura 4.17;

g) **O ambiente deve permitir o modelamento de parâmetros de sistemas, subsistemas e componentes:** conforme mostrado na seção 6.5.1 e na Figura 6.13, parâmetros de sistemas, subsistemas e componentes podem ser modelados no VirSat, visto que fazem parte de seu modelo de dados conceitual;

h) **O ambiente deve permitir que dados de parâmetros de sistemas, subsistemas e componentes sejam usados como entradas de *workflows* de simulação:** conforme mostrado na seção 6.5.2 e na Figura 6.15, valores de parâmetros de sistemas, subsistemas e componentes modelados no VirSat são utilizados pelo RCE como dados de entrada para a execução de *workflows* de simulação;

i) **O ambiente deve permitir que dados gerados por *workflows* de simulação sejam utilizados para atualizar valores de parâmetros**

de sistemas, subsistemas e componentes: conforme mostrado na seção 6.6, após a execução do *workflow* de simulação no RCE, o engenheiro acessa o VirSat e sincroniza os valores dos parâmetros de sistemas, subsistemas e componentes publicados pelo RCE na planilha de integração com os valores correspondentes no VirSat;

j) **O ambiente deve permitir que apenas usuários autorizados sejam capazes de executar modificações em dados de sistemas armazenados no ambiente:** conforme mostrado na seção 6.5.1 e na Figura 6.12, por meio da funcionalidade de *Role Management* do VirSat, é possível restringir o acesso de escrita de dados em cada subsistema, por meio da determinação de um usuário administrador, responsável pela atualização dos dados. Demais usuários do mesmo estudo podem apenas visualizar e consumir os dados publicados;

k) **O ambiente deve permitir o registro da evolução dos valores de parâmetros de sistemas, subsistemas e componentes:** conforme mostrado na seção 6.8, por meio da restauração de versões anteriores do repositório SVN do VirSat, é possível ter acesso a qualquer versão anterior dos valores dos parâmetros de sistemas, subsistemas e componentes;

l) **O ambiente deve permitir o registro da evolução dos dados de *workflows* de simulação utilizados para atualizar os valores de parâmetros de sistemas, subsistemas e componentes:** conforme mostrado na seção 6.8, por meio do *Workflow Data Browser* no RCE é possível verificar a evolução dos dados utilizados em *workflows* de simulação. Todas as versões de dados utilizados em *workflows* de simulação ficam armazenados no banco de dados do RCE;

m) **O ambiente deve permitir o arquivamento de *workflows* de simulação usados para a atualização de valores de parâmetros de sistemas, subsistemas e componentes:** conforme mostrado na

seção 6.7, após a execução de vários *workflows* de simulação no RCE, é possível arquivar no repositório SVN do VirSat um arquivo ZIP com todos os dados do *workflow* que gerou os parâmetros finais dos sistemas, subsistemas e componentes utilizados em uma revisão de projeto;

n) **O ambiente deve permitir que *workflows* de simulação arquivados sejam reexecutados para reavaliar valores de parâmetros de sistemas, subsistemas e componentes:** conforme mostrado na seção 6.7, após o arquivamento no VirSat do arquivo ZIP contendo os dados do *workflow*, é possível importá-lo no RCE e reexecutar o *workflow,* utilizando os mesmos valores dos parâmetros de entrada ou valores diferentes, caso seja necessário avaliar novos cenários.

O ambiente proposto ainda se configura como uma solução de baixo custo, pois não requer a compra de licenças de software, dado que o RCE e o VirSat são disponibilizados gratuitamente pelo DLR. Entretanto, são necessárias muitas horas de dedicação de profissionais especializados para a sua configuração.

Embora seja uma alternativa viável, o ambiente é focado nas fases 0 e A e não se aplica às demais fases do ciclo de vida de sistemas espaciais. Para se aplicável a todas as fases, é necessário o desenvolvimento de um modelo de dados mais sofisticado e de uma aplicação de software que o gerencie. Tais desenvolvimentos estão atualmente sendo conduzidos pelo DLR e resultarão na versão 4.0 do VirSat.

Embora tenha sido possível implantar o ambiente usando o RCE e VirSat, limitações em algumas funcionalidades, como as ferramentas de busca e a integração com planilhas, acabam limitando o potencial destas ferramentas. As ferramentas de busca do RCE e VirSat limitam-se à busca por elementos de texto dentro de arquivos armazenados nos repositórios locais. Uma ferramenta

de busca mais abrangente, utilizando metadados associados aos dados de simulações distribuídas em todas as instâncias da rede RCE, ainda não foi desenvolvida pelo DLR. A ferramenta *Workflow Data Browser* do RCE permite a busca pelo nome do *workflow* em todas as instâncias de rede RCE, mas não permite buscas mais detalhadas, utilizando outros metadados. A integração com planilhas Excel no VirSat, embora funcione, pode apresentar problemas de desempenho que tornam lenta a associação de parâmetros definidos no VirSat a células individuais das planilhas Excel.

O Repositório de Dados de Sistemas Espaciais definido pelas normas da ECSS é um repositório virtual, constituído de vários repositórios físicos que trocam dados entre si utilizando um mesmo modelo de dados. Esta é uma abordagem que pode ser considerada conveniente, pois permite que cada aplicação mantenha sua estrutura particular de dados, otimizada para seu melhor desempenho, ao mesmo tempo que habilita a troca dados com as demais aplicações.

Criar um modelo de dados que abranja todo o ciclo de vida de um sistema espacial é uma tarefa desafiadora, pois é preciso prever, no início do seu projeto, todas as informações que serão geradas e como elas se relacionam entre si. Geralmente, cada tipo de sistema dá origem a um modelo de dados particular, devido às características intrínsecas dos mesmos. Entretanto, aplicações comerciais de gerenciamento de dados e processos geralmente possuem um modelo de dados genérico e de difícil, senão impossível, adaptação para casos específicos.

O que se observa é que, no setor espacial, organizações como o DLR e consórcios internacionais preferem investir em pesquisas para o desenvolvimento de aplicações de software que sejam capazes de gerenciar dados e processos baseados em modelos de dados previamente definidos. No setor aeronáutico, a tendência é da utilização de ferramentas *commercial off-the-shelf* (COTS), desenvolvidas por fornecedores de software que possuem

clientes também de outras áreas, como automotiva e de bens de consumo. Assim, os modelos de dados desta classe de ferramentas costumam ser mais rígidos e permitem pouca margem para customizações, pois o software precisa ser genérico o suficiente para atender a vários tipos diferentes de usuários e o fabricante do software precisa ter controle sobre as aplicações, para que possa prestar suporte adequado. No caso de customizações excessivas, a migração para novas versões do software se torna difícil, ou até mesmo inviável.

8.1. **Sugestões para trabalhos futuros**

Como sugestões de novas pesquisas envolvendo o gerenciamento de dados e processos de simulações, pode-se citar:

a) o desenvolvimento de um software para criação do modelo de dados conceitual de um produto e o gerenciamento de sua evolução ao longo do seu ciclo de vida. Utilizando-se este software, seria possível criar um modelo de dados preliminar nas fases iniciais do projeto espacial e revisá-lo durante as fases seguintes. Este software também poderia ser usado para instanciar uma estrutura de dados real de um produto, a partir do modelo de dados criado, e permitir a criação de conexões com repositórios de dados do sistema espacial. Assim, este software também seria um ponto central de acesso a todos os dados do produto, agindo como um grande “índice” de dados do projeto. Caso o modelo de dados evolua durante o projeto, este software manteria a compatibilidade dos dados já cadastrados com os dados publicados usando o modelo revisado;

b) a extensão do RCE para a inclusão de um gerenciador da estrutura de dados do CPACS, a exemplo do que foi realizado com o VirSat, que possui uma ferramenta (*Study Navigator*) onde é possível instanciar elementos do modelo de dados e controlar sua evolução. Assim, seria possível, de maneira mais simples, criar, editar e

compartilhar estruturas de dados de aeronaves completas usando o RCE;

c) o estudo de um modelo de dados mais abrangente que o CPACS, ou a proposta de melhoria do mesmo, de forma a atender todas as fases do ciclo de vida de um produto aeronáutico;

d) o estudo de como a definição de um modelo de dados de um sistema no início de seu ciclo de vida pode auxiliar o processo de *Engenharia de Sistemas*;

e) o desenvolvimento de uma nova interface no RCE para execução remota de aplicações que possuem interface gráfica com o usuário, para a entrada de dados durante a execução das mesmas;

f) o aprimoramento das ferramentas de busca do RCE e VirSat;

g) o desenvolvimento de componentes do VirSat 4.0 em parceria com o DLR;

h) o desenvolvimento de uma interface no RCE para a visualização gráfica do *pedigree* de dados gerados por um *workflow*;

i) a expansão do *pedigree* de dados do RCE, com a introdução da possibilidade de referenciar dados utilizados em outros *workflows* como entradas em novos *workflows*, e manter a rastreabilidade dos mesmos entre *workflows*.

REFERÊNCIAS BIBLIOGRÁFICAS

APPROACH. Norfolk: Naval Safety Center, v.31, n. 3, 1985. 25 p.

BLACKBURN, M. R.; BUSSER, R.; NAUMAN, A. **Removing requirement defects and automating test**. Software Productivity Consortium NFP, 2001. Disponível em: <https://www.researchgate.net/publication/252757817_Removing_Requirement_Defects_and_Automating_Test> Acesso em: 22 set. 2017.

BOEHM, B. A spiral model of software development and enhancement. **ACM SIGSOFT Software Engineering Notes**, v. 11, n. 4, p. 14-24, August 1986.

BRASIL. Portaria Nº 5.149, de 14 de novembro de 2016. **Regimento Interno do Instituto Nacional de Pesquisas Espaciais**, Art. 75, p. 32. Disponível em: <http://pesquisa.in.gov.br/imprensa/jsp/visualiza/index.jsp?data=16/11/2016&jornal=1&pagina=27&totalArquivos=136>. Acesso em: 18 jan. 2018.

CHAGAS, R. A. J.; LOURO, A. C.; SOUSA, F. L.; SANTOS, W. G. Satellite simulator for verification of mission operational concepts in pre-phase A studies. In: INTERNATIONAL SYSTEMS & CONCURRENT ENGINEERING FOR SPACE APPLICATIONS CONFERENCE, 7., 2016, Madrid, Spain. **Proceedings...**Madrid: UFM, 2016.

CPACS. **Releases · DLR-LY_CPACS · GitHub**. DLR-LY, 2017. Disponível em: <https://github.com/DLR-LY/CPACS/releases>. Acesso em: 13 nov. 2017.

DESHMUKH, M.; WOLFF, R.; FISCHER, P. M.; FLATKEN, M.; GERNDT, A. Interactive 3D visualization to support concurrent engineering in the early space mission design phase. In: CEAS AIR & SPACE CONFERENCE, 5., 2015, Delft. **Proceedings…** Delft: Netherlands Association of Aerospace Engineers, 2015. oai:elib.dlr.de:101924

DLR. **Institute of space systems**. 2017a. Disponível em: <http://www.dlr.de/irs/en/desktopdefault.aspx>. Acesso em: 30 jul. 2017.

DLR. **Virtual Satellite 3.**2017b. Disponível em: <https://software.dlr.de/p/virsat/home/>. Acesso em: 13 out. 2017.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **ECSS-S-ST-00C.** ECSS system. Description, implementation and general requirements. Noordwijk: ESA-ESTEC Requirements and Standard Division, 2008.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **ECSS-E-ST-40**. Space engineering.Software. Noordwijk: ESA-ESTEC Requirements and Standard Division, 2009a.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **ECSS-M-ST-40 Rev.1.**Space project management.Configuration and information management. Noordwijk: ESA-ESTEC Requirements and Standard Division, 2009b.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **ECSS-M-ST-10C Rev.1.**Space project management - Project planning and implementation. Noordwijk: ESA-ESTEC Requirements and Standard Division, 2009c.

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

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **ECSS-E-TM-10-25A.**Space engineering.Engineering design model data exchange (CDF). Noordwijk: ESA-ESTEC Requirements and Standard Division, 2010b.

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

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

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

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **ECSS-E-TM-40-07 Volume 4A.** Space engineering. Simulation modeling platform – Volume 4: C++ Mapping. Noordwijk: ESA-ESTEC Requirements and Standard Division, 2011d.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **ECSS-E-TM-40-07 Volume 5A.** Space engineering. Simulation modeling platform – Volume 5: SMP usage. Noordwijk: ESA-ESTEC Requirements and Standard Division, 2011e.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **ECSS-E-TM-10-23A.** Space engineering.Space System Data Repository. Noordwijk: ESA-ESTEC Requirements and Standard Division, 2011f.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **ECSS-P-00C.** ECSS. Standardization objectives, policies and organization. Noordwijk: ESA-ESTEC Requirements and Standard Division, 2013a.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **ECSS-E-ST-10C-Rev.1.**Space engineering.System engineering general requirements. Noordwijk: ESA-ESTEC Requirements and Standard Division, 2017.

FEDERAL AVIATION ADMINISTRATION (FAA). **A Brief History of the FAA**. 2018. Disponível em: <https://www.faa.gov/about/history/brief_history/#birth>. Acesso em: 16 jan. 2018.

FEDERAL AVIATION ADMINISTRATION (FAA). **AC 21-48.** Using Electronic Modeling Systems as Primary Type Design Data. 2010. Disponível em: <http://www.faa.gov/regulations_policies/advisory_circulars/index.cfm/go/document.information/documentID/442631>. Acesso em: 01 ago. 2017.

FEDERAL AVIATION ADMINISTRATION (FAA). **AC 20-179.** Certification Data Retention Agreements and Government Records. 2013. Disponível em: <https://www.faa.gov/regulations_policies/advisory_circulars/index.cfm/go/document.information/documentID/1021206>. Acesso em: 11 nov. 2017.

FELDMAN, S. SHERMAN, C. **The high cost of not finding information -** an IDC white paper. 2001. Disponível em: <http://www.ejitime.com/materials/IDC%20on%20The%20High%20Cost%20Of%20Not%20Finding%20Information.pdf>. Acesso em: 01 ago. 2017.

FISCHER, P.; DESHMUKH, M.; MAIWALD, V.; QUANTIUS, D.; MARTELO GOMEZ, A.; GERNDT, A. **Conceptual data model -** a foundation for successful concurrent engineering. In: INTERNATIONAL SYSTEMS & CONCURRENT ENGINEERING FOR SPACE APPLICATIONS CONFERENCE, 7., 7., 2016., Madrid, Spain. **Proceedings...** Madrid: UFM, 2016a.

FISCHER, P.M.; LÜDTKE, D.; LANGE, C. et al. Implementing model-based system engineering for the whole lifecycle of a spacecraft. **CEAS Space**

Journal, v. 9, n. 3, p. 351-365, 2016b. Disponível em: <https://doi.org/10.1007/s12567-017-0166-4>. Acesso em: 15 outubro 2017.

FORSBERG, K. MOOZ, H. The relationship of system engineering to the project cycle. **INCOSE International Symposium**, v.1, n. 1 p. 57–65, October 1991.DOI: 10.1002/j.2334-5837.1991.tb01484.x

INTERNATIONAL COUNCIL OF SYSTEMS ENGINEERING (INCOSE). **Systems engineering handbook**: a guide for system life cycle processes and activities. 4. ed. Hoboken, NJ, USA: John Wiley & Sons, 2015. 290p. ISBN: 9781118999400.

INTERNATIONAL COUNCIL OF SYSTEMS ENGINEERING (INCOSE). **What is Systems Engineering?** 2016. Disponível em: <http://www.incose.org/AboutSE/WhatIsSE>. Acesso em: 11 nov. 2016.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (INPE). **Plataforma Multimissão (PMM)**. 2017a. Disponível em: <http://www.inpe.br/amazonia-1/sobre_satelite/pmm.php>. Acesso em: 21 out. 2017.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (INPE). **AMAZONIA -** sobre o satélite. 2017b. Disponível em: <http://www.inpe.br/amazonia-1/sobre_satelite/>. Acesso em: 21 out. 2017.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (INPE). **AMAZONIA -** carga útil. 2017c. Disponível em: <http://www.inpe.br/amazonia-1/sobre_satelite/carga_util.php>. Acesso em: 28 out. 2017.

JACKSON, C. **The second incarnation of MSC software's SimManager**. ENGINEERING.COM, 2012. Disponível em: <http://www.engineering.com/DesignSoftware/DesignSoftwareArticles/ArticleID/4674/The-Second-Incarnation-of-MSC-Softwares-SimManager.aspx>. Acesso em: 29 jul. 2017.

LARSON, W. J.; WERTZ, J. R. **Space Mission Analysis and Design**. 3. El Segundo, CA, USA: Microcosm Press, 1999. 976p. ISBN(1-881883-10-8).

LYNN, S. **Summary report for an undergraduate research project to develop programs for aircraft takeoff analysis in the preliminary design phase**. Blacksburg, Virginia:.Virginia Polytechnic Institute and State University,1994. Disponível em: <http://www.dept.aoe.vt.edu/~mason/Mason_f/MRsoft.html#TakeOff>. Acesso em: 15 nov. 2017.

MCAFFER, J.; LEMIEUX, J.-M. **Eclipse rich client platform:** designing, coding, and packaging Java(TM) applications. Boston: Addison-Wesley Professional, 2005. ISBN: 0321334612.

MOORE, G. Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff. **IEEE Solid-State Circuits Society Newsletter**, v. 11, n. 5, p. 33-35, 2006.Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4785860>. Acesso em: 01 ago. 2017.

MORAES, A. L. O. **Apreciação, simulação e implementação da arquitetura ima distribuída (dima) e sua aplicação a sistemas espaciais.** 2017. 501 p. IBI: <8JMKD3MGP3W34P/3NR9H6S>. (sid.inpe.br/mtc-m21b/2017/05.05.17.37-TDI). Dissertação (Mestrado em Engenharia e Gerenciamento de Sistemas Espaciais) - Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2017. Disponível em: <http://urlib.net/8JMKD3MGP3W34P/3NR9H6S>. doi: 10.1109/N-SSC.2006.4785860

NAFEMS**. SPDM International Conference**. 2013. Disponível em: <http://www.nafems.org/events/congress/2013/summary/spdm>. Acesso em: 29 jul. 2017.

NAFEMS. **Simulation Data Management Working Group (SDMWG)**. 2016. Disponível em: <http://www.nafems.org/about/technical-working-groups/simulation-data-management>. Acesso em: 11 nov. 2016.

NAFEMS. **About NAFEMS**. 2017a. Disponível em: <http://www.nafems.org/about/about_nafems>. Acesso em: 29 jul. 2017.

NAFEMS**. NAFEMS events**. 2017b. Disponível em: <http://www.nafems.org/events/nafems>. Acesso em: 29 jul. 2017.

NAFEMS. **What is simulation data management?** 2017c. Disponível em: <https://www.nafems.org/downloads/sdmwg/nafems_wt02_-_what_is_simulation_data_management.pdf>. Acesso em: 27 set. 2017.

NAGEL, B.; BÖHNKE, D.; GOLLNICK, V.; SCHMOLLGRUBER, P.; RIZZI, A.; LA ROCCA, G.; ALONSO, J. Communication in aircraft design: can we establish a common language? In: CONGRESS OF THE INTERNATIONAL COUNCIL OF THE AERONAUTICAL SCIENCES, 28., 2012, Brisbane, Australia. **Proceedings…** Brisbane, Australia: ICAS, 2012. Paper ICAS 2012-1.9.1.

NORRIS, M. **Business value from simulation data management – a decade of production experience**. NAFEMS, 2012. e-book. Disponível em:<https://www.nafems.org/publications/browse_buy/browse_by_topic/data/business_value_from_simulation_data_management_a_decade_of_production_experience>. Acesso em: 29 jul. 2017.

NORRIS, M. The Value of Simulation Process & Data Management - Lessons from a Decade of Production Experience. NAFEMS European Conference on Simulation Process and Data Management, 2010.Disponível em: <http://pages.mscsoftware.com/rs/mscsoftware/images/SimManager-Value.pdf>. Acesso em: 01 ago. 2017.

OSGI ALLIANCE. **OSGi The dynamic module system for java**. 2016. Disponível em: <https://www.osgi.org/developer/architecture/>. Acesso em: 23 out. 2016.

RCE. **RCE user guide.** 2017. Disponível em: <http://software.dlr.de/updates/rce/8.x/products/standard/releases/latest/documentation/windows/user_guide.pdf>. Acesso em: 10 out. 2017.

ROYCE, W. W. Managing the development of large software systems. In: IEEE WESCON, 1970, New York. **Proceedings…**New York: IEEE, 1970. p. 1-9.

SHAMIEH, C. **Systems Engineering for Dummies®**: IBM Limited Edition. Hoboken, NJ, USA: John Wiley & Sons, Inc, 2012. 68 p. ISBN 978-1-118-24414-2.

SCHAUS, V.; FISCHER, P. M.; L¨UDTKE, D.; BRAUKHANE, A.; ROMBERG, O.; GERNDT, A. Concurrent engineering software development at german aerospace center - status and outlook. In: INTERNATIONAL WORKSHOP ON SYSTEM AND CONCURRENT ENGINEERING FOR SPACE APPLICATIONS, 4., 2010, Lousane. **Proceedings…** European Space Agency (ESA), 2010.

SCHUMANN, H.; BERRES, A.; MAIBAUM, O.; RÖHNSCH, A.DLR'S virtual satellite approach. In: INTERNATIONAL WORKSHOP ON SIMULATION ON EUROPEAN SPACE PROGRAMMES, 10., 2008, Noordwijk. **Proceedings…** 2008a.

SCHUMANN, H.; BRAUKHANE, A.; GERND, A.; GRUNDMANN, J. T.; HEMPEL, R.; KAZEMINEJAD, B.; ROMBERG, O.; SIPPEL, M. Overview of the new concurrent engineering facility at DLR. In: INTERNATIONAL WORKSHOP ON SYSTEM & CONCURRENT ENGINEERING FOR SPACE APPLICATIONS (SECESA), 3., 2008, Rome. **Proceedings…**ESA, 2008a.

SCHUMANN, H.; WENDEL, H.; BRAUKHANE, A.; BERRES, A.; GERNDT, A.; SCHREIBER, A. Concurrent systems engineering in aerospace: from excel-

based to model driven design. In: CONFERENCE ON SYSTEMS ENGINEERING RESEARCH, 8., March 17-19, 2010, Hoboken, NJ. **Proceedings…** 2010.

SEIDER, D.; LITZ, M; SCHREIBER, A.; FISCHER, P. M.; GERNDT, A. Open source software framework for applications in aeronautics and space. In: IEEE AEROSPACE CONFERENCE, 2012, Big Sky, MT, USA. **Proceedings...**IEEE Xplore Digital Library, 2016, p. 1-11. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6187340>. Acesso em: 02 abr. 2015.

SEIDER, D.; BASERMANN, A.; MISCHKE, R.; SIGGEL, M.; TRÖLTZSCH, A.; ZUR, S. Ad hoc collaborative design with focus on iterative multidisciplinary process chain development applied to thermal management of spacecraft. In: CEAS AIR & SPACE CONFERENCE, 4., 2013, Linköping. **Proceedings...**Linköping University Electronic Press, 2016, ISBN 78-91-7519-519-3.

SOMMERVILLE, I. Software reuse. In:___ (Ed.). **Software engineering**. 9. ed. Boston: Pearson Education, 2011. 773 p. ISBN-13: 978-0-13-703515-1.

SOUZA, M. L. O. **Engenharia de requisitos.** São José dos Campos: INPE, mar. 2017. CSE-204-4: Engenharia de Requisitos.

TAGAWA, G. B S. **Estudo de um controlador de atitude em um simulador de Aviônica Modular Integrada (IMA) aplicado ao Satélite Amazônia-1.** 2013. 303 p. IBI: <8JMKD3MGP7W/3DGNKP2>. (sid.inpe.br/mtc-m19/2013/02.08.16.21-TDI). (INPE- 02.08.16.21-TDI). Dissertação (Mecânica Espacial e Controle) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2013. Disponível em: <http://urlib.net/8JMKD3MGP7W/3DGNKP2>.

VISUALSVN. **Visual SVN server**. 2017. Disponível em: <https://www.visualsvn.com/server/getting-started/>. Acesso em: 28 out. 2017.

WINNER, R. I. PENNELL, J. P. BERTRAND, H. E. SLUSARCZUK, M. M. G. **The Role of Concurrent Engineering in Weapons System Acquisition**. Alexandria: IDA, 1988. 197 p. (IDA REPORT R-338). Disponível em: <http://www.dtic.mil/get-tr-doc/pdf?AD=ADA203615>. Acesso em: 13 jan. 2018.

WORLD WIDE WEB CONSORTIUM (W3C). **SCHEMA**. 2018. Disponível em: <https://www.w3.org/standards/xml/schema>. Acesso em: 16 jan. 2018.