sid.inpe.br/mtc-m21c/2020/07.20.18.10-TDI

SIMULAÇÃO DE SATÉLITES COM BASE EM ADAPTAÇÃO AUTÔNOMA DE MODELOS COMPORTAMENTAIS

Jun Tominaga

Tese de Doutorado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Engenharia e Gerenciamento de Sistemas Espaciais, orientada pelos Drs. Mauricio Gonçalves Vieira Ferreira e Ana Maria Ambrosio, aprovada em 28 de maio de 2020.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34R/42SQ8J8>

INPE
São José dos Campos
2020

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GBDIR)
Serviço de Informação e Documentação (SESID)
CEP 12.227-010
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/7348
E-mail: pubtc@inpe.br

sid.inpe.br/mtc-m21c/2020/07.20.18.10-TDI

SIMULAÇÃO DE SATÉLITES COM BASE EM ADAPTAÇÃO AUTÔNOMA DE MODELOS COMPORTAMENTAIS

Jun Tominaga

Tese de Doutorado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Engenharia e Gerenciamento de Sistemas Espaciais, orientada pelos Drs. Mauricio Gonçalves Vieira Ferreira e Ana Maria Ambrosio, aprovada em 28 de maio de 2020.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34R/42SQ8J8>

INPE
São José dos Campos
2020

Dados Internacionais de Catalogação na Publicação (CIP)

Tominaga, Jun.

T595s Simulação de satélites com base em adaptação autônoma de modelos comportamentais / Jun Tominaga. – São José dos Campos : INPE, 2020.

xxii + 155 p. ; (sid.inpe.br/mtc-m21c/2020/07.20.18.10-TDI)

Tese (Doutorado em Engenharia e Tecnologia Espaciais/Engenharia e Gerenciamento de Sistemas Espaciais) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2020.

Orientadores : Drs. Mauricio Gonçalves Vieira Ferreira e Ana Maria Ambrosio.

1. Satélite artificial. 2. . Simulador operacional. 3. Sistemas especialistas. 4. Algoritmos evolutivos. 5. Programação genética. I.Título.

CDU 629.78:629.7.01

Aluno (a): *Jun Tominaga*

Título: "SIMULAÇÃO DE SATÉLITES COM BASE EM ADAPTAÇÃO AUTÔNOMA DE MODELOS COMPORTAMENTAIS"

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

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

Dra. Maria de Fátima Mattiello-Francisco

Presidente / INPE / São José dos Campos - SP

(X) Participação por Video - Conferência

(X) Aprovado ( ) Reprovado

Dr. Maurício Gonçalves Vieira Ferreira

Orientador(a) / INPE / SJCampos - SP

(X) Participação por Video - Conferência

(X) Aprovado ( ) Reprovado

Dra. Ana Maria Ambrosio

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

(X) Participação por Video - Conferência

(X) Aprovado ( ) Reprovado

Dr. Leandro Toss Hoffmann

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

(X) Participação por Video - Conferência

(X) Aprovado ( ) Reprovado

Dr. Rodrigo Rocha Silva

Convidado(a) / FATEC / Mogi das Cruzes - SP

(X) Participação por Video - Conferência

(X) Aprovado ( ) Reprovado

Dra. Marinalva Dias Soares

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

(X) Participação por Video - Conferência

(X) Aprovado ( ) Reprovado

Este trabalho foi aprovado por:

( ) maioria simples

(X) unanimidade

São José dos Campos, 28 de maio de 2020

AGRADECIMENTOS

Agradeço a todos que tiveram paciência e boa vontade, que colaboraram de alguma forma para a confecção e conclusão deste trabalho. Em especial, gostaria de registrar meus agradecimentos aos meus orientadores, pelas inúmeras reuniões, discussões, revisões, sugestões e críticas construtivas. Estas contribuições foram de vital importância para a elaboração desta Tese e, sem elas, ela não teria sido concluída. Agradeço ainda aos membros da Banca e ao pessoal da Pós-Graduação, pela compreensão e pelo empenho neste momento difícil de quarentena devido à pandemia de COVID-19, sem os quais a defesa final desta Tese não teria sido possível. Agradeço também a meus colegas de trabalho, que colaboraram na execução de minhas atividades profissionais, permitindo que pudesse obter um pouco do tempo necessário para me dedicar às atividades de pesquisa e desenvolvimento associadas à elaboração este trabalho.

RESUMO

Conforme os satélites artificiais em órbita vão envelhecendo, os respectivos modelos comportamentais do simulador operacional de tais satélites devem ser adaptados para refletir a degradação de componentes e falhas dos equipamentos embarcados. O ajuste de parâmetros e expressões é constantemente necessário para o modelo comportamental de um simulador operacional permanecer atualizado e útil. Esta atividade pode ser descrita como um problema de otimização, no qual o erro entre os valores de telemetrias recebidos do satélite e os valores dos parâmetros de simulação correspondentes deve ser minimizado. Neste contexto, são propostos um processo para adaptação autônoma de modelos comportamentais e uma arquitetura para um sistema de simulação que apoia a execução do processo. Uma característica de adaptação autônoma é dada pela definição de uma heurística para programação genética, que busca uma solução ótima entre um conjunto de soluções candidatas que são geradas iterativamente. As novas gerações de soluções candidatas são obtidas aplicando mudanças aleatórias às existentes, que são então avaliadas conforme critérios de aptidão. Visando reduzir o escopo de aplicação da programação genética e o esforço computacional, foi proposta a análise de correlação de parâmetros de regras comportamentais. Resultados obtidos com implementação de um protótipo mostram que o processo com uso de programação genética, apoiado pela arquitetura proposta, foi capaz de alcançar soluções viáveis para a adaptação de modelos comportamentais baseados em regras, e que por meio da análise de correlação pode-se reduzir o escopo do conjunto de regras a ser tratado.

Palavras-chave: Satélite artificial. Simulador operacional. Sistemas especialistas. Algoritmos evolutivos. Programação genética.

SATELLITE SIMULATION BASED ON AUTONOMOUS ADAPTATION OF BEHAVIOR MODELS

ABSTRACT

As satellites age in orbit, the behavior models of their operational simulators must be adapted to reflect onboard components degradation and equipment failures. The adjustment of parameters and expressions is constantly required for the behavior model of an operational simulator to remain adapted and useful. This activity can be described as an optimization problem, in which the error between the satellite telemetry values and the simulator parameters must be minimized. In this context, a process for behavior models adaptation and an architecture for a simulation system that supports the execution process are proposed. The autonomous adaptation trait is given by the definition of a heuristic for genetic programming, to seek an optimal solution among a set of candidate solutions that are generated iteratively. New generations of candidate solutions are obtained by applying random changes to existing ones, which are then evaluated according to fitness criteria. In order to reduce the application scope of genetic programming and computational effort, parameters correlation analysis of behavior rules was proposed. Results obtained using a prototype show that the process with use of genetic programming, supported by the proposed architecture, can achieve viable solutions for the adaptation of rule-based behavior models, ant that by means of correlation analysis the scope of the set of rules to be worked on can be reduced.

Keywords: Artificial Satellite. Operational Simulation. Expert Systems. Evolutionary Algorithms. Genetic Programming.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

LISTA DE SIGLAS E ABREVIATURAS

AOM	Adaptive Object-Model
CBERS	China-Brazil Earth Resources Satellite
CSV	Comma-Separated Values
DLR	Deutsches Zentrum für Luft- und Raumfahrt
EADS	European Aeronautic Defence and Space Company
ECSS	European Cooperation on Space Standardization
ESA	European Space Agency
FDIR	Fault Detection, Isolation and Recovery
GCS	Ground Control System
INPE	Instituto Nacional de Pesquisas Espaciais
MMP	Multi Mission Platform
NASA	National Aeronautics and Space Administration
PMM	Plataforma Multi Missão
SCD	Satélite de Coleta de Dados
SMP	Simulator Model Portability Standard
SISAO	Simulador de Satélite Operacional do Amazonia-1
TT&C	Telemetry, Tracking and Commanding
XML	Extensible Markup Language
XSCC	Xi'An Satellte Control Center

LISTA DE SÍMBOLOS

A	Ampère
°C	Graus Celsius
C	Coulomb
V	Volt

SUMÁRIO

Pág.

1 INTRODUÇÃO

Softwares de simulação são utilizados para reproduzir o comportamento de algum fenômeno físico por meio de modelos matemáticos, sem a necessidade de execução do fenômeno simulado. Se o comportamento do objeto de simulação é alterado por qualquer motivo depois que o software de simulação é entregue, então isto caracteriza, do ponto de vista de engenharia de software, em uma mudança de requisitos após o seu desenvolvimento. No caso de simuladores operacionais, construídos de acordo com especificações da fase de projeto do satélite (AMBROSIO et al., 2006), dificilmente se comportam em conformidade com satélites reais após vários anos de operação, tendo sobrevivido a diversas falhas embarcadas, tanto previstas quanto imprevistas quando concebidos.

Historicamente, temos no INPE (Instituto Nacional de Pesquisas Espaciais) operações de TT&C (Telemetry, Tracking and Commanding) de satélites artificiais há mais de 20 anos. Os satélites mais antigos controlados atualmente pelo INPE são os da família SCD (Satélite de Coleta de Dados), o SCD1, lançado em fevereiro de 1993, e o SCD2, em outubro de 1998. Ambos permanecem com suas cargas-úteis ativas fornecendo dados de missão a seus usuários. A forma de operar estes satélites hoje difere bastante de sua concepção original. Transponders não são mais ligados e desligados dentro de visada das estações terenas de TT&C. Computadores de bordo estão desativados. No caso do SCD1, as baterias estão desconectadas, assim o satélite opera somente quando iluminado, e a manutenção de atitude encontra-se degradada devido à redução drástica de sua velocidade de rotação e consequente perda de rigidez giroscópica.

Além dos satélites da família SCD, atualmente o INPE controla dois satélites de sensoriamento remoto da série CBERS (Satélite Sino-Brasileiro de Recursos Terrestres), o CBERS-4 e o CBERS-4A. O CBERS-4, lançado em 2014, finalizou a sua vida útil prevista e se encontra operacional em vida estendida, enquanto o CBERS-4A, lançado em 2019, está na fase de comissionamento, após o qual iniciará a fase operacional. Outro satélite a ser controlado pelo

INPE, o AMAZONIA-1, encontra-se em fase final de desenvolvimento e seu lançamento previsto para final de 2020. A ocorrência de falhas embarcadas causa mudanças perceptíveis no comportamento real de um satélite em voo, que devem ser refletidas no modelo comportamental de um simulador operacional que se propõe a representá-lo. Tais mudanças são tradicionalmente implementadas e validadas por meio de análise de telemetrias.

Empiricamente, durante a operação de satélites, identificaram-se duas categorias de mudanças no comportamento. Mudanças abruptas, causadas por falhas embarcadas e suas eventuais soluções, levam a chaveamentos físicos ou lógicos no funcionamento e na operação do satélite e à necessidade de novos procedimentos de voo e de solo. Mudanças graduais, causadas por envelhecimento contínuo de componentes, levam valores de parâmetros à deriva, podendo resultar em necessidade ou de redefinição dos limites normais de operação, ou à definição de novos procedimentos específicos para a monitoração deste comportamento.

A mudança de comportamento de um satélite em voo é uma realidade constante. Por exemplo, o CBERS-4, lançado em 2014 e projetado para uma vida operacional de três anos até 2017, já passou por diversas falhas embarcadas, sendo algumas delas permanentes. Entre elas, podem ser citadas a perda de um dos canais que conectam o painel solar ao circuito de carga de baterias (CBERS, 2016a) e a telemetria presa do status da chave de conexão das câmeras brasileiras ao gravador do bordo ou ao transmissor de dados (CBERS, 2016b). Em consequência de alterações em bordo, bases de dados de operação devem ser modificadas para adequar os alarmes de telemetrias de monitoração, em conformidade com solução de falhas embarcadas.

Além das falhas, a degradação de componentes em equipamentos embarcados, exige que limites de telemetrias necessitem de constantes ajustes, implicando na adequação das bases de dados operacionais às mudanças observadas no comportamento do satélite. Por exemplo, no banco de dados operacional do software de controle do CBERS-4, foram

contabilizadas 67 alterações de 07 de agosto de 2015 a 30 de abril de 2019, entre cadastramento de novos parâmetros auxiliares, ajustes de limites de monitoração e sequências de comandos alternativos, além de adaptações para novos equipamentos e cadastramento de novos endereços de rede nas estações terrenas de TT&C. Estas tarefas de alterações têm se caracterizado pela complexidade inerente, pela demora na implantação, e pela propensão do sistema resultante a conter erros logo após a implantação das correções.

No caso dos simuladores operacionais, a implementação de tais mudanças caracteriza-se por um custo elevado, em termos de recursos financeiros, mão-de-obra e tempo dedicado. Isto porque, tradicionalmente, requerem uma análise e uma modelagem do problema para, em seguida, obter-se uma modificação dos requisitos de simulação a fim de se implantar e validar a sua solução. Para a adaptação de modelos comportamentais de softwares de simulação, a abordagem tradicional é constituída por uma análise das mudanças comportamentais e pela reelaboração de requisitos, envolvendo as equipes de operação e desenvolvimento, para uma reformulação do projeto, seguida pela reconstrução do software de simulação conforme as alterações relatadas e diagnosticadas.

No INPE atualmente tanto operadores, especialistas e desenvolvedores de satélites e de software são funcionários públicos concursados com anos de experiência. Entretanto, as recentes políticas governamentais apontam para uma tendência de corte de vagas. À medida que funcionários antigos envelhecem e se aposentam sem a entrada de novos, criam-se situações em que o conhecimento é perdido juntamente com a saída de pessoal. A recente modificação das leis trabalhistas permitiria a futura contratação. O ingresso de funcionários novos para assumir funções de operadores de satélites exige treinamento antes de assumirem seus cargos. Enquanto funcionários experientes estiverem controlando e analisando os satélites a atualização de simuladores não seria urgente. De fato, no contexto atual, o treinamento de operadores tem sido realizado “*on-the-job*” com o satélite em voo, com novos procedimentos sendo instruídos à medida que falhas embarcadas determinam novos comportamentos. Quando isto ocorre, modelos comportamentais de

simuladores operacionais dos satélites SCD e CBERS têm sido mantidos inalterados e os simuladores pouco usados, dado o tamanho do esforço de atualização. Soma-se a isto também a dificuldade de obtenção de recursos, tanto humanos quanto financeiros, para a execução deste tipo de atualização.

Outra questão relacionada ao desenvolvimento de simuladores de satélites está no fato de que detalhes de implementação em software de modelos de simulação de satélites são dificilmente encontrados na literatura, provavelmente, por causa de sua importância comercial. Lee et al. (2007) submeteram uma patente para um sistema de simulação de satélite utilizando componentes embarcados, ou "*hardware-in-the-loop*". Em outra submissão para patente Lee et al. (2006) propõem uma abordagem híbrida para um simulador com componentes em hardware e software, alegando que simuladores em software são fáceis de desenvolver, mas apresentam problemas de acurácia e fidelidade em comparação à abordagem híbrida proposta por eles. O uso de "*hardware-in-the-loop*" permite a obtenção de resultados fiéis de simulação em situação nominal, mas a reprodução em solo de efeitos de degradação em ambiente espacial se torna difícil. O simulador seria inerentemente não-adaptável, ou de difícil adaptação.

A modificação autônoma de um software de simulação para sua adequação à mudança comportamental do objeto simulado permitiria uma redução drástica do esforço de manutenção. Uma modificação autônoma atribui características autoadaptativas ao modelo comportamental de um simulador de satélite. Para isto é necessária, primeiramente, uma capacidade de monitoramento do comportamento do satélite e comparação com o modelo de simulação corrente, para a correta identificação da necessidade de mudança e, em seguida, a adequada atualização do modelo de forma a reproduzir o comportamento do satélite simulado após a mudança.

Esta Tese propõe dotar capacidade autoadaptativa a um simulador de satélite com um modelo comportamental do tipo sistema especialista. Para isto, uma análise de correlações entre parâmetros e expressões de regras comportamentais é realizada para permitir a alterações de valores de

parâmetros relevantes e a elaboração de novas regras por meio de programação genética. Esta última se enquadra em um tipo de algoritmo evolutivo da categoria de otimização multiobjetivo, que busca obter soluções em forma de programas para a execução de uma tarefa ou, no caso desta Tese, a adaptação de regras comportamentais por meio da alteração de expressões.

Os algoritmos evolutivos não têm sido explorados pelos autores pesquisados para a finalidade de adaptação autônoma de modelos comportamentais de simuladores de satélites como proposta por esta Tese. A razão pode estar relacionada à sua exigência de alta capacidade computacional para um desempenho considerado minimamente satisfatório, devido à necessidade de um número significativo de soluções candidatas gerados aleatoriamente, testadas iterativamente para a obtenção de uma solução final. Visando minimizar este problema, nesta Tese, a fim de otimizar o trabalho de atualização das regras do modelo comportamental, é proposta uma identificação de regras e parâmetros relevantes do modelo e o descarte de informações irrelevantes, para em seguida, iniciar a busca por uma solução analítica e, na falha desta, utilizar a programação genética após restringir e reduzir a quantidade de regras a serem processadas e testadas.

1.1 Objetivo da tese

O objetivo desta Tese consiste em propor um processo para adaptação autônoma de modelos comportamentais de um simulador operacional de satélites artificiais e caracterizar uma arquitetura de um sistema de simulação que apoie tal processo. Desta forma, permitir que conforme as características reais do satélite vão sendo modificadas ao longo de sua vida útil, assim também serão atualizados o simulador operacional deste satélite.

O processo proposto faz uso de programação genética para obtenção de expressões alternativas e de uma análise de correlação de regras e parâmetros para a redução do espaço de busca da programação genética e a obtenção de valores de parâmetros. A arquitetura do sistema de simulação consiste de um conjunto de componentes interdependentes distribuídos correlacionados que

apoiam o processo. Tais componentes permitem que um simulador com modelos comportamentais baseados em regras e com propriedades auto adaptativas continuamente se adapte de modo a minimizar o erro do modelo com relação aos valores reais do satélite.

No INPE o projeto do Simulador SIMCBERS (CBERS, 2006; 2012; 2016c) e seus derivados (AMAZONIA, 2019) serão diretamente beneficiados com a capacidade autoadaptável de modelos comportamentais baseados em regras proposto por este trabalho.

O desenvolvimento deste trabalho de pesquisa foi motivado pela experiência do autor em operação dos satélites e treinamento de operadores sob a responsabilidade do Centro de Controle de Satélites do INPE, tendo vivenciado as consequências da degradação dos equipamentos dos satélites em órbita e a necessidade de manutenção dos sistemas em solo, em particular os simuladores de satélites. Além disso, o autor participou da especificação do comportamento dos subsistemas do satélite de grande porte, CBERS 3&4, para o Simulador Operacional SIMCBERS. Estas especificações (CBERS, 2006; 2012; AMBROSIO et al., 2006) foram realizadas através de tabelas causa-efeito, tendo sido a forma comum melhor aceita pelos engenheiros das diferentes disciplinas envolvidas na definição do comportamento dos subsistemas de satélite do programa CBERS. Especificações dos modelos comportamentais de cada um dos 14 subsistemas do satélite CBERS 3&4 são documentadas em especificação de requisitos de software para o SIMCBERS descritos na Seção 2.1.

1.2 Metodologia da pesquisa

A hipótese desta Tese é de que a adaptação autônoma de modelos comportamentais de simuladores de satélites pode ser alcançada com sucesso.

Para comprovar essa hipótese, inicialmente foram realizados estudos acerca de simuladores operacionais de satélites e foram pesquisadas falhas embarcadas típicas e ferramentas para seu processamento, para incorporação a modelos de simuladores.

A constatação do ínfimo número de referências abertas sobre trabalhos relacionados à modelagem em simuladores de satélites levou essa pesquisa a basear-se nos trabalhos desenvolvidos no INPE, cujos mais recentes utilizam modelagem baseada em regras com armazenamento externo de dados e metadados, e que também estavam em consonância com Eickhoff (2009) e com o Memorando Técnico da ESA (ECSS, 2011c).

Estudos acerca da estrutura de expressões de regras e sua semelhança à representação em árvore sintática levaram à escolha da programação genética como técnica para a modificação de expressões de regras. Ressalvas encontradas na literatura referentes ao elevado custo computacional da programação genética levaram à elaboração de um método de análise de correlação para redução do espaço de busca.

Foram estudados trabalhos sobre métodos de adaptação de software. Tais métodos não apontaram para o uso de programação genética para adaptação de modelos de simulador de satélites, portanto, a análise de correlação de regras utilizada nesta Tese foi criada pelo autor, caracterizando uma contribuição ao estado da arte.

A validação da hipótese foi feita com provas de conceito organizadas da seguinte forma:

(1) Desenvolvimento de uma arquitetura de software para adaptação de um modelo comportamental de um simulador de satélite de acordo com telemetrias de referência do satélite simulado.

(2) Desenvolvimento da uma ferramenta para geração de novas regras mais aptas para caracterizar o novo comportamento identificado por meio de telemetrias e sua substituição no modelo comportamental do simulador.

(3) Definição de cenários simplificados para teste do algoritmo de programação genética conforme casos típicos de falhas simples.

(4) Definição de cenários complexos, baseados em um modelo simplificado de um satélite fictício para descrever o processo analítico de identificação e

alteração de parâmetros discrepantes em um modelo de simulação baseado em regras.

A ideia geral da solução consiste na **identificação de discrepâncias** em um determinado instante, medidas por meio de diferenças entre os valores das telemetrias e dos parâmetros de simulação correspondentes, maiores do que limiares pré-estabelecidos, levando à busca por um novo modelo. **Modelos candidatos** são gerados a partir de modificações do modelo original, que são testados por meio de um simulador simplificado (um componente da arquitetura do Sistema de Simulação). A saída de cada modelo candidato é comparada com telemetrias de referência e, caso não sejam identificadas discrepâncias, o modelo candidato é considerado uma solução válida para substituir o modelo original a partir do instante em que a discrepância foi identificada.

A contribuição desta tese consiste na definição de um processo de adaptação de modelos comportamentais de simuladores de satélites baseados em regras, e de uma arquitetura geral para possibilitar a sua execução. Em destaque, o processo inclui uma atividade para a qual foi definida uma heurística de programação genética para obtenção de expressões alternativas e uma análise de correlação de regras e parâmetros, para a redução do espaço de busca da programação genética e a obtenção de valores de parâmetros para as novas regras.

1.3 Organização da tese

O conteúdo desta Tese a partir deste ponto encontra-se estruturado conforme a seguir:

Capítulo 2 - Fundamentação teórica e trabalhos relacionados - lista o resultado de pesquisa de outros trabalhos que abordam ou utilizam alguma técnica ou ferramenta em comum com a proposta desta Tese, incluindo estudos acerca de simuladores operacionais de satélites, falhas embarcadas e softwares autoadaptativos.

Capítulo 3 - Processo e arquitetura - descreve em alto nível o processo e a arquitetura para a adaptação autônoma de modelos de simulador e uma

breve introdução ao protótipo desenvolvido para comprovação e obtenção de resultados.

Capítulo 4 – Adaptação autônoma – descreve por meio de um exemplo a metodologia empregada no processo para a adaptação do modelo. Inicialmente a estrutura de um modelo baseado em regras é descrita por meio de um exemplo simples, bem como a composição de uma regra comportamental por expressões e suas possíveis modificações. Em seguida, é mostrado como a programação genética é aplicada para a obtenção de uma regra alternativa a partir da original. Por fim, é descrita a abordagem analítica.

Capítulo 5 - Prova de conceito - mostra a capacidade autoadaptativa de modelos comportamentais com cenários complexos para detecção e alteração de parâmetros de simulação e regras do modelo comportamental e com cenários simples para validação do uso de programação genética.

Capítulo 6 - Conclusão - lista conclusões, contribuições, pontos de melhoria e trabalhos futuros.

Apêndice A - Protótipo do componente Reconfigurador de Modelo - Mostra detalhes da implementação do protótipo do componente de adaptação de modelos de simulação.

Apêndice B - Interfaces do componente Reconfigurador de Modelo - Descreve interfaces com outros componentes do ambiente de operação.

2 FUNDAMENTAÇÃO TEÓRICA E TRABALHOS RELACIONADOS

Na pesquisa sobre simuladores operacionais de satélites, o livro de Eickhoff (2009) se destaca pela abrangência do assunto, apesar de não entrar em detalhes específicos da modelagem de comportamentos. O memorando técnico concebido pela Agência Espacial Europeia (ESA) (ECSS, 2010) é a principal referência por apresentar uma arquitetura típica de simuladores operacionais definida com base em muitos anos de experiência, porém, outros artigos científicos descrevendo arquiteturas de simuladores de satélites também foram pesquisados.

Com intuito de estudar as soluções que abordassem modelagem de falhas em simuladores, foram pesquisadas falhas embarcadas típicas e ferramentas para seu processamento para incorporação a modelos de simuladores. Em geral, a modelagem e a análise de falhas são realizadas por meio de relações de causa e efeito, que caracterizam regras de sistema especialista. (GUIOTTO et al., 2003) mostrou a adaptação de modelos comportamentais a falhas simples previstas em projeto.

No arcabouço geral dos simuladores de satélites os modelos comportamentais são em grande número representados em regras de causa-efeito. Estudos sobre a estrutura de expressões de regras e sua semelhança à representação em árvore sintática levaram à escolha da programação genética como técnica para a modificação de expressões de regras.

Devido ao pequeno número de referências abertas relacionadas à modelagem em simuladores de satélites e ao histórico de trabalhos em pesquisa e desenvolvimento de simuladores de satélites desenvolvidos no INPE, esta pesquisa apresenta e incorpora os conhecimentos adquiridos neste instituto. E quanto a evolução dos simuladores, o conceito de software adaptativo foi explorado.

2.1 Arquiteturas de simuladores operacionais de satélites

Simuladores de satélites têm diferentes aplicações em uma missão espacial. Na engenharia de operações, simuladores de satélites são usados em validação do segmento solo, desenvolvimento, preparação e validação de

planos de voo, desenvolvimento de conceito de operações e verificação de operações fora do nominal em casos de falha. Segundo a ECSS ("*European Cooperation on Space Standardization*"), simuladores operacionais têm como objetivos validar procedimentos de controle em voo, treinar a equipe de controle em voo e dar suporte durante operações à solução de problemas e manutenção (HENDRICKS & EICKHOFF, 2005; ECSS, 2010).

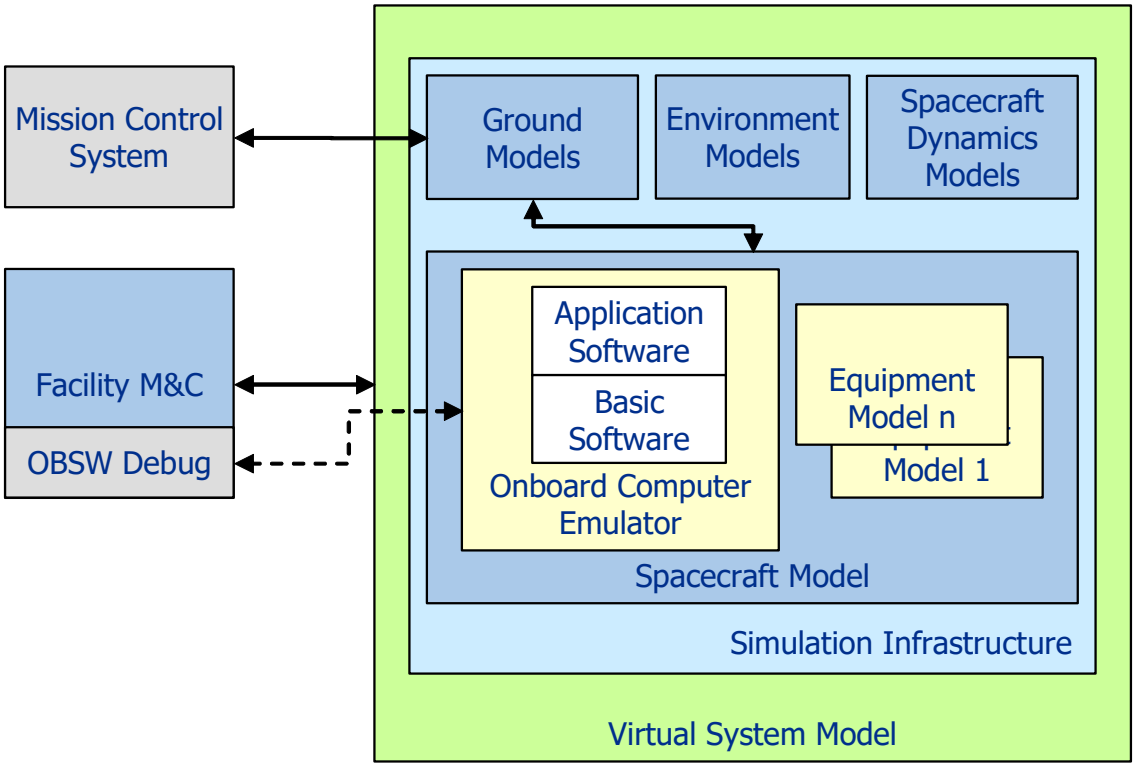
O primeiro simulador operacional de satélite do INPE foi desenvolvido para apoiar a operação do satélite SCD-1 (INPE, 1990) e mais tarde foi adaptado para apoiar a operação do SCD2. A experiência com o desenvolvimento deste simulador foi aproveitada para a especificação do simulador do satélite CBERS-1 (XSCC & INPE, 2000). Um resumo das características dos simuladores desenvolvidos no INPE entre 1992 até 2002 é apresentado em AMBROSIO et al. (2006). Simuladores antigos têm modelos codificados em funções, sub-rotinas ou classes que não permitem sua adaptação dinâmica sem recompilação. Simuladores mais recentes do INPE como os do Satélite CBERS-4 (CBERS, 2012) e Amazônia-1 (AMAZONIA, 2019) utilizam modelagem baseada em regras com armazenamento externo de expressões em arquivo XML ("*Extended Markup Language*").

Devido ao alto custo de desenvolvimento de simuladores para missões espaciais, a ESA definiu uma arquitetura de referência para o desenvolvimento de simuladores de satélites baseada no padrão SMP ("*Simulator Model Portability Standard*") da norma ECSS (2011a). O formato de dados do metamodelo encontra-se em ECSS (2011b). A definição de componentes do modelo é descrita em ECSS (2011c). Esta padronização permite a implementação de modelos de simulação para diferentes missões utilizando uma mesma referência de arquitetura base e interfaces pré-definidas, reduzindo esforços de desenvolvimento e pesquisa.

Uma arquitetura típica de simuladores operacionais segundo a norma ECSS é mostrada na Figura 2.1. Nesta arquitetura o simulador operacional é composto por: (i) um núcleo ou infraestrutura de simulação em tempo real, (ii) modelos de subsistemas, com capacidade de simular funções de software embarcado,

comportamentos de malhas elétrica e térmica e resposta de cargas-úteis a telecomandos por meio de telemetrias de serviço, além de (iii) modelos ambientais, (iv) modelos de órbita e (v) modelos dos sistemas de solo. O simulador operacional provê interfaces de monitoração e controle com o sistema de solo de TT&C, indicado na figura como "*Mission Control System*" (MCS) e interfaces de configuração, teste e validação do próprio simulador (ECSS, 2010).

Figura 2.1. Arquitetura de simulador operacional.



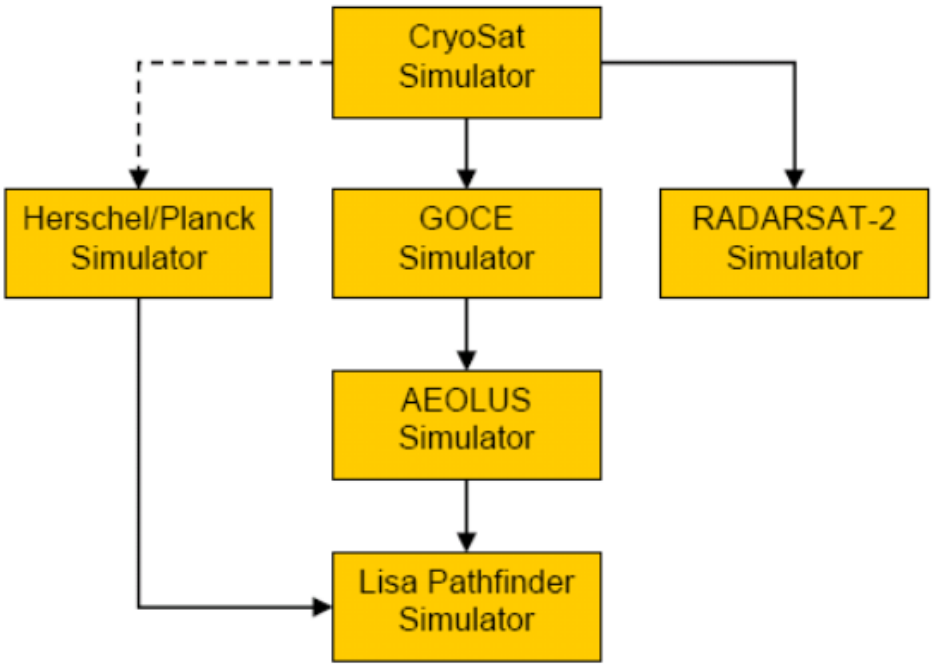
Fonte: ECSS (2010).

Dentro desta filosofia de reutilização de partes de um simulador, um ambiente simulador utilizado para apoio ao desenvolvimento e verificação baseado em modelo é o MDVE ("*Model-based Development and Verification Environment*") desenvolvido pela empresa EADS ("*European Aeronautic Defence and Space Company*") Astrium (HENDRICKS & EICKHOFF, 2005). Este ambiente utiliza modelos, bases de dados e procedimentos integrados e compartilhados ao longo de diversas fases do projeto de uma missão espacial cuja estrutura seria compartilhada por diversas missões. Na arquitetura do MDVE o centro de

controle de missão (MCC) utiliza um sistema de controle de missão (MCS) conectado a um simulador de satélite em software que se comunica com um simulador de computador embarcado com "*hardware-in-the-loop*". O ambiente MDVE é usado pela Universidade de Stuttgart para suas missões de satélites universitários, como na missão Flying Laptop (BRANDT et al., 2008; FRITZ et al., 2010). O MDVE é ainda utilizado nas missões CryoSat, GOCE, Aeolus e LISA Pathfinder da Agência Espacial Europeia (ESA), além da missão TerraSAR do Centro Aeroespacial Alemão (DLR) (EICKHOFF et al., 2007).

A Figura 2.2 ilustra simuladores de diferentes missões espaciais que foram desenvolvidos utilizando este conceito, todas derivadas do simulador para o satélite CryoSat (SEBASTIÃO et al., 2008).

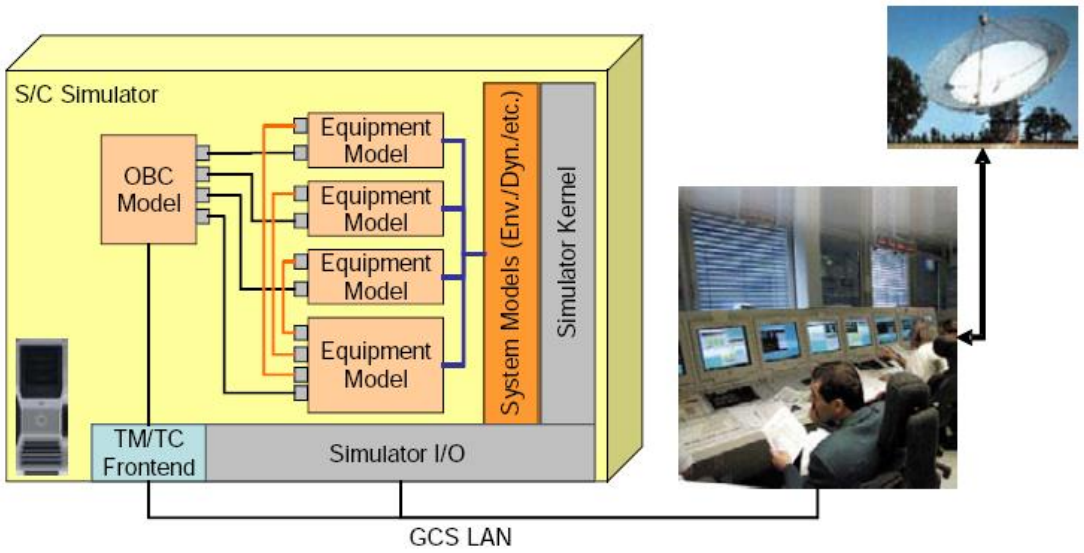
Figura 2.2. Simuladores de satélites derivados do CryoSat.



Fonte: Sebastião et al. (2008).

A Figura 2.3 mostra uma forma alternativa de representar uma arquitetura típica de simuladores operacionais.

Figura 2.3. Simulador para suporte a operações de veículos espaciais.

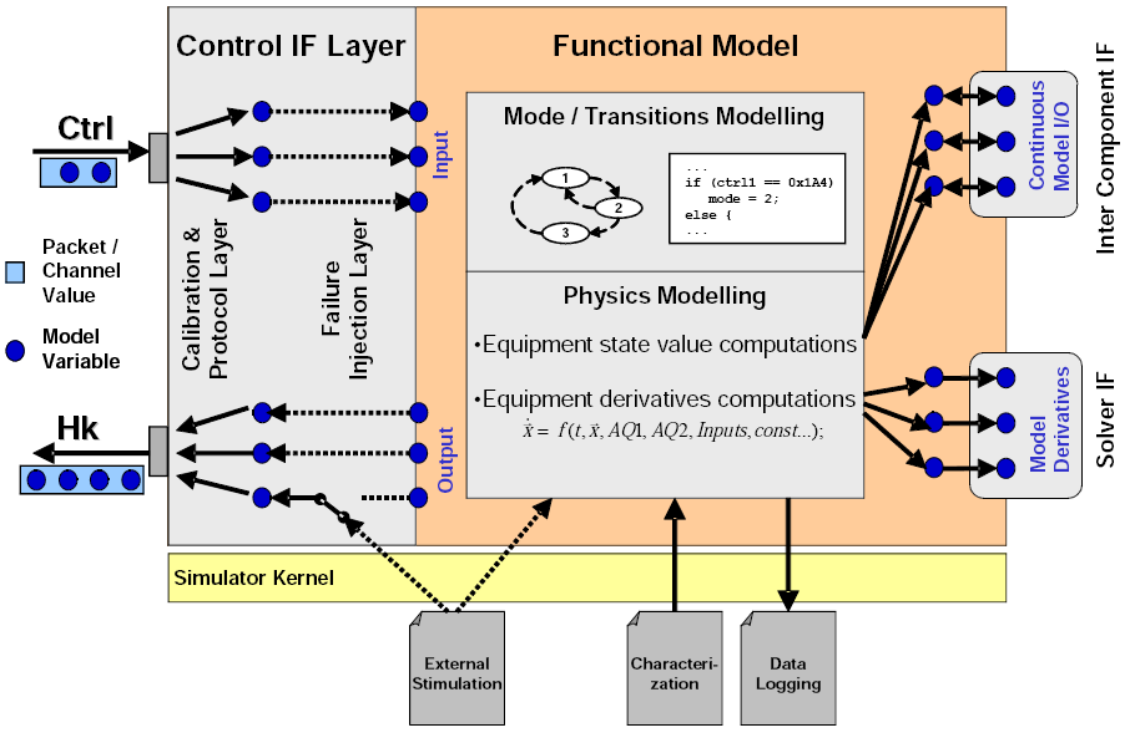


Fonte: Eickoff (2009).

No simulador operacional anterior (Figura 2.3), a emulação de funções de software embarcado é realizada por meio de um modelo de equipamento específico, responsável pela função de supervisão de bordo (OBC = Onboard Computer), bem como o processamento de dados de monitoração (telemetrias) e controle (telecomandos). Os cálculos de balanços elétrico e térmico são realizados por meio de modelos de equipamentos, interconectados entre si e estimulados por sinais externos. As fontes de estímulos externos são provenientes de modelos externos, tais como ambiental, dinâmica de voo, etc. O controle da simulação é realizado a partir de um núcleo simulador ("*Simulator Kernel*"), que se comunica com o sistema de solo de TT&C (GCS = "*Ground Control System*") por meio de rede de área local (GCS LAN) (EICKHOFF, 2009). A Figura 2.4 mostra a estrutura de um modelo de equipamento genérico que compõe a arquitetura exibida na Figura 2.3.

Na Figura 2.4, um modelo de equipamento interage com outros componentes do simulador por meio de interfaces de dados de controle ("*Ctrl*") e de monitoração ("*Hk*") ligados ao sistema de solo de TT&C.

Figura 2.4. Estrutura de um modelo de equipamento.



Fonte: Eickhoff (2009).

O comportamento é regido por modelos funcionais que simulam transições de estados discretos e computação de parâmetros físicos contínuos. O núcleo de simulação é responsável por prover estímulos externos provenientes de outros modelos (EICKHOFF, 2009).

Outro trabalho de base pesquisado para elaboração desta Tese foi o de Kang et al. (1995), que descreve um simulador denominado ARTSS (“*Advanced Real-Time Satellite Simulator*”). Neste simulador um núcleo de tempo real possui interfaces com operadores de TT&C e de simulação, sendo acoplado a um modelo comportamental do veículo espacial. Os modelos de serviço são implementados em software, enquanto o modelo de carga-útil adota a abordagem “hardware-in-the-loop”.

O modelo comportamental do ARTSS é composto por unidades divididas por subsistemas. A unidade do subsistema de controle de atitude e órbita (AOCSU) utiliza uma estrutura de controle em malha fechada, composta por um componente de dinâmica de voo, incluindo órbita e atitude, que possui

interfaces com a eletrônica de controle de atitude (ACE) por meio de sensores e atuadores. O modelo da unidade de suprimento de energia elétrica (EPSU) simula funções de painéis solares, motor de painéis solares, bateria, regulador de carga, eletrônica de suprimento de energia e montagem de “pull-up” da bateria, em função de estados de carga de carga ou descarga e iluminação ou eclipse. A unidade de controle térmico (THCSU) simula as posições de chaves de aquecedores em resposta a temperaturas calculadas com base em cálculo de ângulos solares. A unidade de TT&C (TCRSU) simula status de chaveamento do subsistema, e a unidade de carga-útil (PLDU) implementa interface de comunicação com o hardware do equipamento transponder de banda Ku.

A Seção 4.1 desta Tese descreve um conjunto de regras que simula o status de chaves de equipamentos como os descritos no artigo para THCU e TCRU do ARTSS. Na Seção 5.1.1 desta Tese é dado um exemplo de um conjunto de regras que simula o comportamento dinâmico de suprimento de energia de um satélite, semelhante à descrição do EPSU.

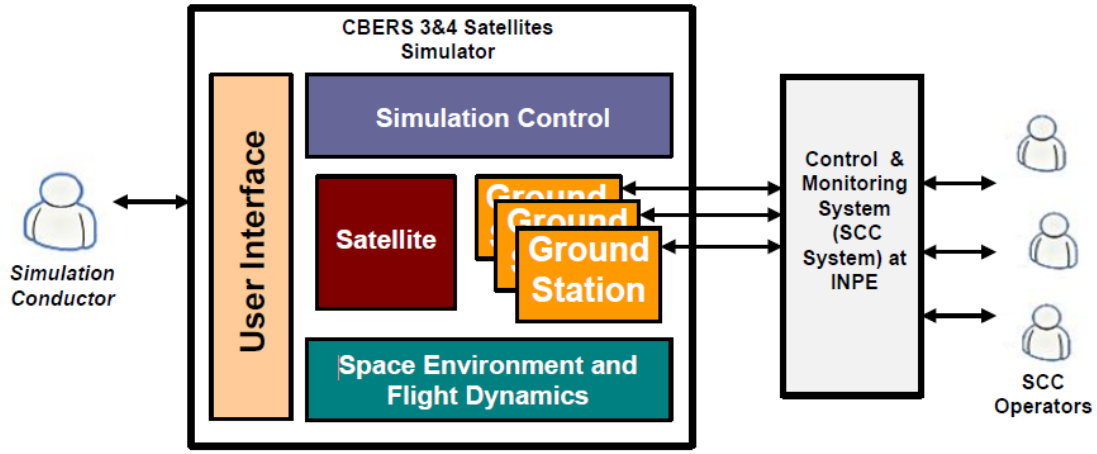
O SIMCBERS (CBERS, 2012; 2016c) e o SATSIM (TOMINAGA, 2010) são simuladores de satélites operacionais baseados em regras. Apesar de bastante diferentes em termos de funcionalidade, suas arquiteturas têm semelhanças quanto à adoção de um núcleo de processamento multimissão e modelos comportamentais externos específicos por missão com codificação de informação baseada em regras. Esta codificação é fixa e não é autoadaptável.

O SIMCBERS é um simulador operacional em software da segunda geração dos satélites da família CBERS (China Brazil Earth Resources Satellite). Esta família de satélites é constituída por três satélites (CBERS-3, CBERS-4 e CBERS-4A), mas o SIMCBERS foi concebido e adaptado apenas para o CBERS-3 e o CBERS-4. Seus objetivos incluem o treinamento de operadores e a validação de sistemas de solo de TT&C.

A Figura 2.5 mostra a arquitetura geral do SIMCBERS com seus modelos ambientais de espaço e dinâmica de voo (SPACE ENVIRONMENT and FLIGHT DYNAMICS) e estações terrenas (GROUND STATION) e satélite

(SATELLITE), que agrega os modelos comportamentais de subsistemas que o compõem. O controle da simulação é realizado por um núcleo controlador (SIMULATION CONTROL) que se comunica com o usuário condutor da simulação (SIMULATION CONDUCTOR) por meio de interface (USER INTERFACE) e com o usuário operador (SCC OPERATOR) do software do sistema de solo de TT&C (SCC SYSTEM) por meio de modelos das estações (GROUND STATION).

Figura 2.5. Visão geral do simulador SIMCBERS.



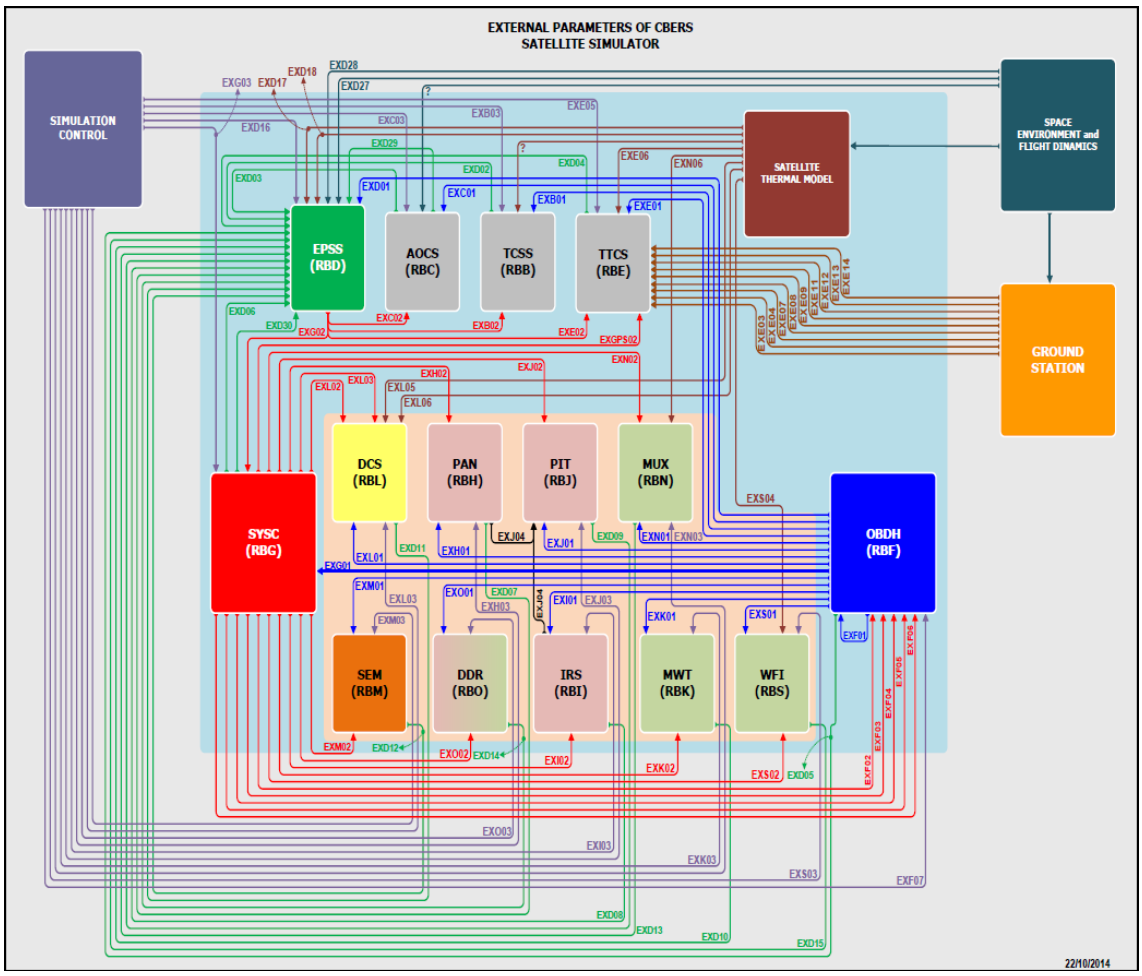
Fonte: CBERS (2016d).

A

Figura 2.6 mostra os modelos que compõem o SIMCBERS. O retângulo azul no centro da figura corresponde ao satélite, dentro dos quais existem modelos do sistema térmico (SATELLITE THERMAL MODEL) e de subsistemas de suprimento de energia elétrica (EPSS), controle de atitude e órbita (AOCS), controle térmico (TCSS), telemetria, rastreio e comando (TTCS), circuitos e conexões de dados e energia do sistema (SYSC) supervisão de bordo (OBDH), além dos subsistemas de cargas úteis de coleta de dados (DCS), monitor de ambiente espacial (SEM), câmeras pancromática e multiespectral de alta resolução (PAN), multiespectral de média resolução (MUX), infravermelho (IRS) e de visada larga (WFI), transmissores de dados das câmeras PAN + IRS (PIT) e MUX + WFI (MWT) e gravador de dados digital (DDR), cujas especificações encontram-se respectivamente em (CBERS, 2014b; 2016e;

2016f; 2016g; 2016h; 2017a; 2017b; 2017c; 2017d; 2017e; 2017f; 2017g; 2017h; 2017i).

Figura 2.6. Modelos componentes do simulador SIMCBERS.



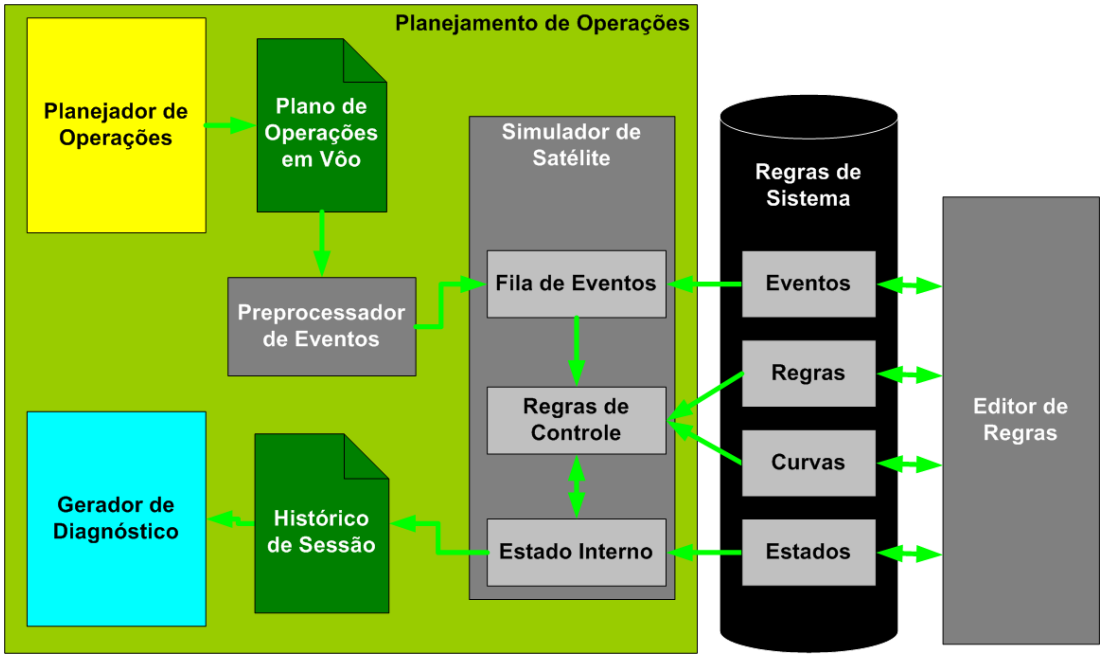
Fonte: CBERS (2016c).

Um projeto do INPE de simulador operacional de satélites derivado do SIMCBERS é o SISAO. Ele simula o satélite Amazonia-1 por meio de modelos de componentes em software baseado em regras e utiliza também um computador de bordo na forma de "*hardware-in-the-loop*" (AMAZONIA, 2017; 2019).

O SATSIM é um projeto acadêmico de um simulador de satélite genérico desenvolvido pelo autor, cuja finalidade foi a validação de planos operacionais em voo. A Figura 2.7 mostra a arquitetura em nível de sistema de adaptação do SATSIM. Ela possui um núcleo de processamento que trabalha com valores de

parâmetros de simulação para composição do estado inicial, que são atualizados a cada passo de simulação por meio de regras de controle para o comportamento do satélite e disparo de eventos como execução de telecomandos empilhados em uma fila de eventos (TOMINAGA, 2010).

Figura 2.7. Arquitetura de sistema do SATSIM.



Fonte: Tominaga (2010).

O SATSIM utiliza um modelo comportamental baseado em regras descrito inteiramente em formato XML, facilitando a edição de regras de sistema do modelo comportamental por usuários. A edição de regras é uma atividade manual de atualização de comportamento, de tal forma que esta ferramenta se beneficiaria com a introdução de uma capacidade autoadaptativa.

2.2 Representação de falhas embarcadas para Simuladores de Satélites

A norma ECSS (2009a) compila uma lista extensiva de modos de falha possíveis em componentes embarcados em missões espaciais. A maior parte dos modos de falhas em voo se resume a ocorrências de curto-circuito (SC = "*short circuit*") ou circuito-aberto (OC = "*open circuit*") de componentes eletrônicos. Outros modos de falhas possíveis incluem vazamentos ou rupturas mecânicos, desconexões de contatos, chaves presas, valores fixos de entrada

e saída, desvios de desempenho, fenômenos de eventos singulares (SEP), medidas erradas de sensores e falhas funcionais.

Um exemplo de modo de falha embarcado típico consiste na saída presa de uma linha de telemetria, fazendo-a assumir em um valor fixo. A Figura 2.8 mostra este modo de falha listado entre outros tipos previstos para modos de falha de microcircuitos.

Figura 2.8. Modos de falha possíveis para microcircuitos.

08. MICROCIRCUITS (family/group 08 xx)		
Type	Failure modes	Remarks
08 10 microprocess/microcontrol/peripher 08 20 memory SRAM 08 21 memory DRAM 08 22 memory PROM 08 23 memory EPROM 08 24 memory EEPROM 08 29 memory others 08 30 programmable logic 08 40 ASIC technologies digital	Any single output SC to V+/V- Any single output stuck to 0/1 Any single output in high impedance Any single input SC to V+/V- Any single input SC to 0/1 OC of any single power supply V+ to V- SC SEP	OC of any single power supply including ground pin For complex IC's (ASIC, FPGA, µP,...), a functional FMEA/FMECA is performed taking into account the physical implementation . SEP effect analysis performed in the FMEA/FMECA is based on the output of the radiation analyses
08 41 ASIC technologies linear 08 42 ASIC technologies mixed analog/digital	Any single functional failure Any single output SC to V+/V- Any single output stuck to 0/1 Any single output in high impedance Any single input SC to V+/V- Any single input SC to 0/1 OC of any single power supply V+ to V- SC SEP	OC of any single power supply includes ground SEP effect analysis performed in the FMEA/FMECA is based on the output of the radiation analyses
08 50 linear operational amplifier 08 51 linear sample and hold amplifier 08 52 linear voltage regulator 08 53 linear voltage comparator 08 54 linear switching regulator 08 55 linear line driver 08 56 linear line receiver 08 57 linear timer 08 58 linear multiplier 08 59 linear switches 08 60 linear multiplexers/demultiplexer 08 61 linear analog to digital converter 08 62 linear digital to analog converter 08 69 linear other functions 08 80 logic families 08 90 other functions 08 95 microwave monolithic integrated circuits (MMIC)		For linear integrated circuit SET worst case effect is considered when sensitivity identified trough radiation analyses (generally temporary effect)

Fonte: ECSS (2009a).

Outro tipo de modo de falha típico possível em bordo consiste na chave de um relê se prender em uma posição fixa. As Figuras 2.9 e 2.10 listam possíveis modos de falhas para relês.

Figura 2.9. Modos de falha possíveis para relês.

09. RELAYS (family/group 09 xx)		
Type	Failure modes	Remarks
09 01 non latching 09 02 latching	Relay stuck in one position Coil Open Circuit 2 open contacts (relay stuck in intermediate position) 2 contacts in opposite position Short Circuit between fix contacts Short Circuit between coil and one contact (epsilon) Short Circuit between contact and structure (epsilon)	See details in Figure G-1, Figure G-2, Figure G-3 hereafter. Failure modes only applicable to electromechanical devices. For other devices performing same function (e.g. thermally actuated micro-machined relays), identify alternate possible failure modes and consider them according to the technology of the relay

Fonte: ECSS (2009a).

Figura 2.10. Modos de falha aplicáveis conforme o tipo de relê.

Failure modes	Mono-stable relays (type J412, T12, GP5 or equivalent)	Bi-stable relays (type J422, TL12, GP250 or equivalent)	Bi-stable relays (type EL210 or equivalent)	Bi-stable relays (type GP3 or equivalent)
Relay stuck in OFF position:				
- coil Open Circuit	A	A	A	A
- contact stuck OFF	A	A	A	A
Relay stuck in ON position:				
coil Open Circuit	N/a	A	A	A
contact stuck ON	A	A	A	A
Coil short circuit	N/A	N/A	N/A	N/A
2 open contacts (relay stuck in intermediate position)	N/A	A (2)	N/A	A (1)
2 contacts in opposite positions	A (1)	A (1)	N/A	A (1)
Short circuit between fix contacts	A (1)	A (1)	N/A	A (1)
Short circuit between coil and one contact	A (1)	A (1)	N/A	A (1)
(1): Negligible probability of occurrence. To be considered in the FMECA for traceability aspects. (2) : Not applicable for GP250 A: applicable N/A: not applicable				

Fonte: ECSS (2009a).

A Figura 2.10 lista modos de falhas aplicáveis e não aplicáveis categorizados de acordo com o tipo de relê utilizado, também de segundo a norma ECSS. Este tipo de informação é útil para associar componentes embarcados e equipamentos construídos a partir deles, de modo a prever falhas possíveis e prováveis ou, conforme a aplicação, descartar falhas impossíveis e improváveis.

Uma ferramenta utilizada para análise qualitativa de falhas hipotéticas é a FMEA ("*Failure Modes and Effects Analysis*"). A FMEA pode ser estendida para FMECA ("*Failure Modes, Effects and Criticality Analysis*") para incluir informações referentes à probabilidade de ocorrência e a severidade de ocorrência de falhas. Estas ferramentas relacionam pontos de falha simples com sintomas observáveis, podendo ser utilizadas para monitoramento de saúde e diagnóstico em caso de ocorrência de falhas (ECSS, 2009b; WERTZ et al., 2011).

A partir da FMEA/FMECA, realiza-se a análise de FDIR ("*Failure/Fault Detection, Identification/Isolation and Recovery*"), tendo como objetivo demonstrar a conformidade do projeto com requisitos de tolerância a falhas do projeto e, ao mesmo tempo, proteger a missão de perdas prematuras por causa de falhas previstas (ECSS, 2009b). Funções de FDIR implementam ações de detecção, identificação, isolamento e recuperação de falhas, de forma autônoma em bordo, ou por intervenção de solo.

Em solo, a detecção e a identificação de falhas são realizadas a partir de análise de telemetrias, por meio de expressões lógicas preferencialmente simples, utilizando valores de telemetrias monitoradas como termos. O isolamento de falhas deve salvaguardar a integridade da missão evitando a propagação da falha a outras partes do veículo espacial, por meio de comandos para desabilitar o equipamento danificado. A recuperação de falhas deve permitir a retomada de operações de missão interrompidas por causa de falhas, por exemplo, por meio de comandos para ativação de equipamento redundante ou pela criação de nova sequência de comandos que permitam recuperar a falha (ECSS, 2008).

Após a recuperação, telemetrias de monitoração são verificadas para garantir a execução correta do procedimento. Os modos de falha e os procedimentos associados são tabelados em forma de regras causais, que explicitam precondições necessárias para a execução de procedimentos (XSCC, 2014; CBERS, 2014a). A Figura 2.11 mostra um exemplo de procedimento de correção de falha do CBERS-4, listando as telemetrias de monitoração para detecção de uma anomalia e um grupo de telecomandos para envio, bem como um parâmetro de monitoração a posteriori para confirmar a execução da correção.

Figura 2.11. Exemplo de procedimento de correção de falha do CBERS-4.

D03	Bat temp over temperature range
Describe	any battery group over temperature range
justice	TMD024 TMD025
	TMB110 TMB111 TMB112 TMB113
method	If (TMD024>13°C || TMD025>13°C) & (TMB110>13°C || TMB111>13°C || TMB112>13°C || TMB113>13°C),
	then alarm, send telecommand as following at once, and notice to CAST. AND INDE
	1) Send TCD28
	2) Send TCD30
	3) Send telecommand group 0 (turn off playload)
result	TMD024<10°C
	TMD025<10°C

Fonte: CBERS (2014a).

No exemplo anterior (Figura 2.11), telemetrias de temperaturas internas das baterias 1 (TMD024) e 2 (TMD025) e as temperaturas externas de baterias do pack A (TMB110), pack B (TMB111), pack C (TMB112) e pack D (TMB113) são utilizadas para identificar se a temperatura de alguma bateria está excessivamente alta ou não. Em caso de temperaturas altas, telecomandos deverão ser enviados para solucionar o problema. Neste caso, a verificação de sua execução correta se dá através da monitoração das telemetrias temperatura das baterias (TMD024 e TMD025), que deverão abaixar (< 10 °C). Este procedimento de monitoração e correção é um exemplo de regra, composto por expressões de precondição e efeito.

À medida que ocorre o envelhecimento do satélite em voo, tais procedimentos devem ser atualizados ao longo da missão e sua sobrevida (CBERS, 2015).

O procedimento ilustrado (Figura 2.11) foi atualizado uma vez. O procedimento original é mostrado na Figura 2.12 para efeito de comparação. Nesta versão original, apenas as telemetrias internas das baterias (TMD024 e TMD025) eram utilizadas para a identificação da falha e seus valores foram alterados de 20°C para 13°C. Os telecomandos a serem enviados também foram alterados.

Figura 2.12. Exemplo de procedimento obsoleto de correção de falha do CBERS-4.

FLD03 – Ni-Cd battery over temperature range

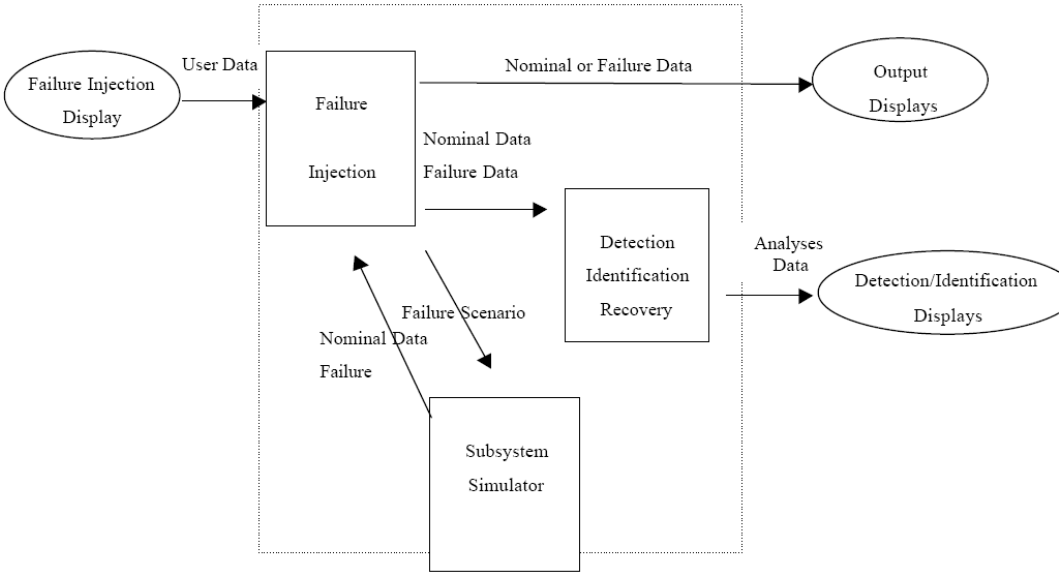
Fault ID	FLD-03	**Fault Name**	Ni-Cd battery over temperature range
Subsystem	EPSS	**Fault Degree**	II
Others SSs linked	Thermal		
Recovery Method	TC and AUT	**Fault Heritage**	CBERS2
Fault Indicatives	Battery temperature (TMD024 and TMD025)		
Fault phase	Any		
Recovery Plan	If overheating is due to BHC failure, recovery of BHC circuit is automatic. BHC will switch to redundant circuit in case of MAIN circuit failure after battery temperature reachs 20+/- 1 degrees Celsius. Verify the battery compartment temperature by its neighbor temperature sensors. If failure is due to EOC, the two immediate actions are to turn on the long trickle charge. If this method doesn't work, decrease the EOC voltage level.		
TC and TM	**Step** | **TC ID** | **TC Name** | **TM** **To turn on the long trickle charge** 1 | TCD028 | BAT1/2 Long Trickle Charge On | TMD159 shall read "1" (>3.5V) **To decrease the EOC voltage level for BAT1** 1 | TCD010 | BCC1 NEXT CHARGE CURVE UP | TMD028 (The value depends on the previous selected curve). See AD1 for details) **To decrease the EOC voltage level for BAT2** 1 | TCD011 | BCC2 NEXT CHARGE CURVE UP | TMD029 (The value depends on the previous selected curve). See AD1 for details)		
Special Remarks	To obtain temperature of battery compartment with thermal subsystem.		
Recovery Risks	Thermal runaway in case the plan is not executed at the right time.		

Fonte: XSCC (2014).

Comparativamente, esta Tese permite o uso de programação genética para explorar novos modos de falha não previstos em FDIR.

Em Guiotto et al. (2003) é definida uma ferramenta embarcada de processamento de falhas, chamada SMART-FDIR, a qual utiliza técnicas de lógica nebulosa para executar tarefas de detecção, identificação e recuperação de falhas. A ferramenta é associada a um simulador, como mostra a arquitetura ilustrada na Figura 2.13.

Figura 2.13. Arquitetura da ferramenta SMART-FDIR.



Fonte: Guiotto et al. (2003).

Observa-se a presença do simulador de subsistema como parte integrante do SMART-FDIR, o qual é capaz de processar cenários de falhas previamente previstas por meio de FDIR, alterando o modelo de uma configuração nominal para uma configuração de falha. Este simulador recebe cenários de falha como entrada e retorna dados de falha nominal como saída. O simulador pode variar, podendo ser simples ou de alta fidelidade.

2.3 Modelos comportamentais baseados em regras em simuladores de satélites

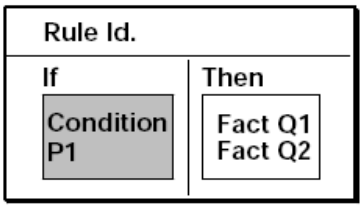
Uma possível abordagem para se gerar modelos de equipamentos para simulação consiste no uso de sistemas especialistas ou sistemas baseados em conhecimento. Sistemas especialistas utilizam uma máquina de inferência para avaliar conhecimentos abstratos, na forma de regras ou casos, armazenados em bases de conhecimento (EICKHOFF, 2009).

Modelos comportamentais baseados em regras têm suas origens no conceito de sistemas especialistas. Um sistema especialista baseado em regras consiste em um software inteligente, que utiliza regras de inferência para resolução ou diagnóstico de problemas. Máquinas de inferência deste tipo têm aplicação em diversos campos, incluindo-se entre eles o planejamento e

controle de processos. No caso, estipulam-se ações a serem executadas por meio de regras lógicas, que especificam os efeitos a serem aplicados conforme condições são satisfeitas. Isto é realizado segundo o modelo do sistema, construído com base no conhecimento de um ou mais especialistas. Em sistemas especialistas baseados em regras, as regras abstratas que representam o conhecimento são compostas por duas partes: uma de condições e outra de conclusões. Se todas as precondições listadas na parte de condições forem satisfeitas, então se aplicam o efeito dos resultados listados na parte de conclusões (BERNARD, 1988; LIAO, 2005).

A Figura 2.14 ilustra um exemplo de regra, composto por uma condição P1 que, caso seja satisfeita, causa como conclusões a aplicação dos resultados Q1 e Q2 (EICKHOFF, 2009).

Figura 2.14. Exemplo de representação de uma regra.



Fonte: Eickhoff (2009).

Simuladores operacionais em desenvolvimento no INPE utilizam modelos comportamentais baseados em regras (TOMINAGA et al., 2012). A modelagem do comportamento dos subsistemas do satélite é baseada em regras sendo sua especificação dada em tabelas de causa-efeito. A representação de regras, com expressões de precondições e expressões de efeitos na forma tabular é exemplificada na Figura 2.15 (CBERS, 2012; TOMINAGA et al., 2012).

Figura 2.15. Exemplo de representação de regras de comportamento do SIMCBERS.

Preconditions			Effects		
Param A	Param B	Param C	Param X	Param Y	Param Z
a1	b1	c1	x1	y1	z1
a2	b2	> c2	x2	y2	z2
<= a3		-	x3	-	Z+C

Equivalent to:

```
if (A == a1 and B == b1 and C == c1)
   then (X = x1, Y = y1, Z = z1)
if (A == a2 and B == b2 and C > c2)
   then (X = x2, Y = y2, Z = z2)
if (A <= a3 and B == b2)
   then (X = x3, Z = Z+C)
```

Fonte: Tominaga et al. (2012).

Expressões utilizadas para representar precondições e efeitos que compõem regras causais podem ser desmembradas em operandos, operações e funções, conforme ilustrado no lado direito da Figura 2.15.

As falhas embarcadas resultam em mudanças de comportamento do satélite que são descritas por uma ou mais regras que compõem o modelo comportamental de seu simulador.

A busca por novos conjuntos de regras que melhor descrevem as mudanças introduzidas pode ser caracterizada como um problema de otimização e pode ser resolvido por programação genética.

2.4 Programação genética

A programação genética consiste em uma técnica de computação evolutiva que tem como objetivo fazer com que computadores resolvam problemas de forma autônoma, sem a necessidade de um programador instruir explicitamente como estes problemas devam ser resolvidos. A programação genética surgiu como desdobramento da computação evolutiva, que constitui um dos campos de pesquisa da inteligência artificial. A população pai inicial e a geração de filhos são criados de forma aleatória por meio de operações genéticas de mutação e crossover, que são aplicadas para a criação de trechos de código de programa executáveis sintaticamente válidos (KOZA, 1992; POLI et al., 2007).

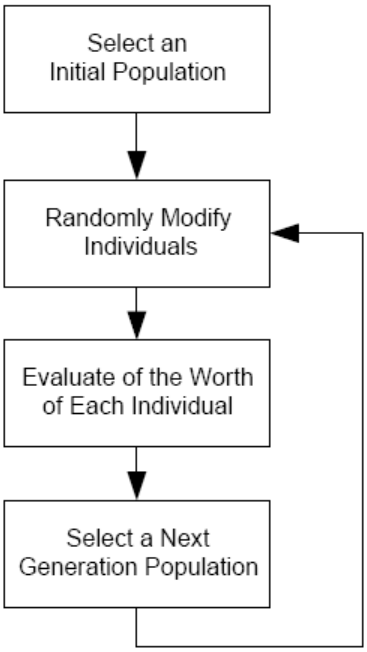
A computação evolutiva pode ser exemplificada de forma clássica por meio de algoritmos evolutivos, cujo objetivo primário consiste na obtenção de soluções para problemas de otimização. Um processo de otimização consiste em minimizar uma função custo, de modo a obter um conjunto de variáveis de projeto que levem a um valor mínimo dentro do espaço de projeto, respeitando um conjunto de restrições de projeto.

Em algoritmos evolutivos, soluções candidatas são levantadas para um problema de otimização. Cada solução candidata é tratada como um indivíduo, cujo conjunto constitui uma população. Gerações sucessivas de populações compostas por novos indivíduos são avaliadas de acordo com critérios de aptidão. Soluções consideradas mais aptas são escolhidas como indivíduos

pais para a geração de indivíduos filhos. Soluções filhas são obtidas a partir de variações aleatórias das soluções pais na forma de mutações e recombinações (FOGEL, 2006).

A Figura 2.16 ilustra o fluxo básico de um algoritmo evolutivo típico, mostrando o processo iterativo. O processo é repetido iterativamente até que seja alcançada uma condição de parada.

Figura 2.16. Fluxo básico de um algoritmo evolutivo.



Fonte: Fogel (2006).

A escolha da população inicial de soluções candidatas pode ser completamente aleatória ou parcialmente direcionada, conforme o nível de conhecimento do problema. Uma vantagem em se conhecer melhor o problema consiste na possibilidade de se restringir mais o espaço de busca da solução e, assim, escolher-se uma população inicial inerentemente mais apta. Isto leva à diminuição do número de gerações necessárias para se chegar a uma solução final, resultando numa convergência mais rápida.

Na modificação aleatória de indivíduos, valores de variáveis de projeto são alterados ao acaso para se gerar novas soluções candidatas, segundo critérios de cruzamento ou mutações. No modelo análogo biológico, durante o

cruzamento ocorre troca de material genético entre genes. De forma similar, informações constituindo variáveis de projeto podem ser cruzadas por meio de permutação de bits. Já o mecanismo de mutação envolve erros aleatórios durante a cópia de material genético no modelo biológico, o que pode ser reproduzido por meio de inversão de bits que compõem variáveis de projeto.

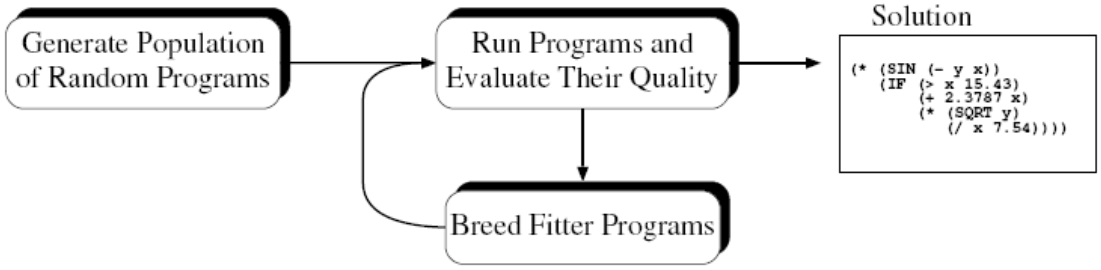
A avaliação do valor de cada indivíduo é realizada por meio da aplicação de uma função de aptidão, baseada na função objetivo a se minimizar.

A escolha dos indivíduos a compor a geração seguinte pode ser realizada por meio de comparação de valores da função adaptabilidade entre os indivíduos da população. Entretanto, técnicas alternativas podem incluir a escolha proposital de uma pequena quantidade de indivíduos com desempenho inferior, porém com material genético bastante diferente dos demais indivíduos, mantendo a diversidade da população. Tipicamente, a seleção se realiza pela escolha dos indivíduos com melhor desempenho global na avaliação de aptidão, porém concedendo a oportunidade a indivíduos menos aptos.

Condições de parada incluem a obtenção de ao menos uma solução cujas características obedeçam a critérios mínimos de aceitação, indicado uma convergência bem-sucedida, ou exceções como, por exemplo, atingimento de um número máximo predefinido de interações ou tempo máximo de execução, ou ausência de melhoria nas últimas iterações, indicando não convergência.

A programação genética consiste em um caso particular de algoritmo evolutivo, onde indivíduos são algoritmos ou programas de computador. A Figura 2.17 ilustra o fluxo básico de um exemplo genérico de programação genética.

Figura 2.17. Loop principal de programação genética.

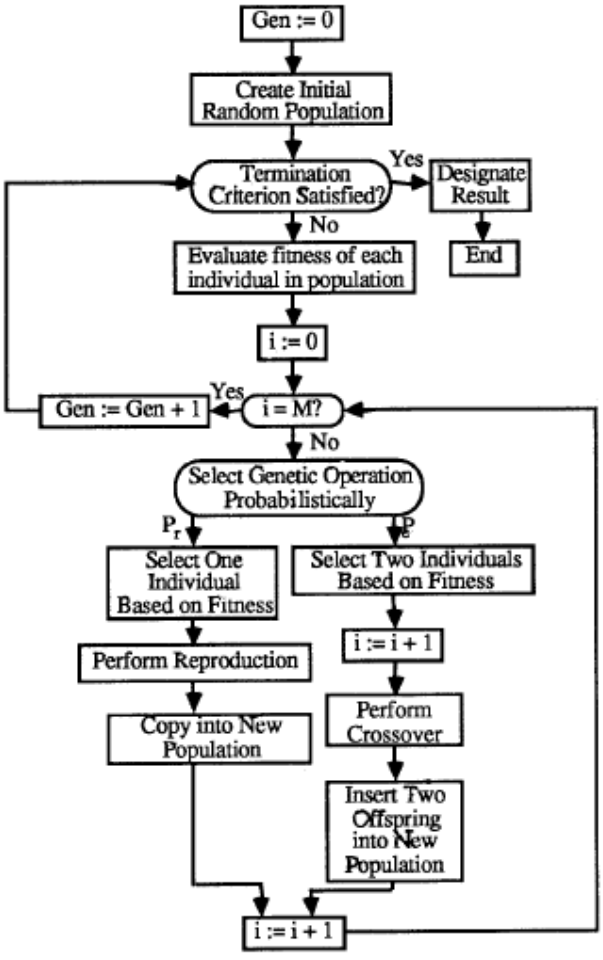


Fonte: Poli et al. (2007).

A geração de uma população inicial em programação genética tipicamente utiliza soluções simples geradas aleatoriamente. Entretanto, dependendo do nível de conhecimento acerca do domínio do problema, a população inicial pode ser gerada de forma direcionada. Entretanto, deve-se garantir um mínimo de diversidade para evitar dominância de soluções filhas que levem a uma convergência prematura (POLI et al., 2007).

A Figura 2.18 mostra o fluxograma de execução da programação genética, que é similar a outros algoritmos evolutivos. Inicialmente a programação genética vislumbrava apenas a aplicação de operações de reprodução e crossover, com troca de parte de código da geração pai (KOZA, 1994). Posteriormente, surgiram variantes (POLI et al., 2007) em que houve introdução da operação de mutação, conforme adotada nesta Tese.

Figura 2.18. Fluxograma de programação genética.



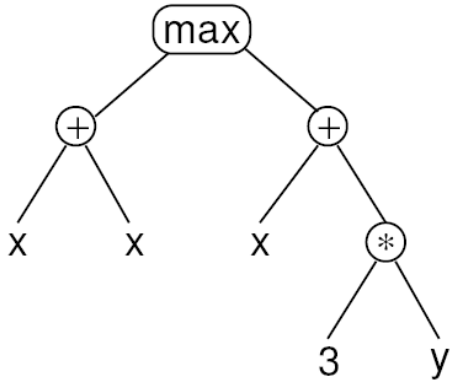
Fonte: Koza (1994).

A execução de soluções candidatas e a avaliação de suas qualidades fornecem o subsídio necessário para a classificação de indivíduos para a formação da próxima geração (POLI et al., 2007).

Para a obtenção da próxima geração, indivíduos pais são submetidos a cruzamento e mutação. Para uma melhor visualização dos processos de modificação aleatória de indivíduos em programação genética, costuma-se utilizar uma árvore sintática.

A árvore sintática, exemplificada na Figura 2.19 é comumente utilizada no escopo da programação genética em substituição a linhas de código. Neste exemplo, variáveis e constantes constituem folhas da árvore, ao passo que funções e operações formam sua raiz ou nós internos. Em programação genética, algoritmos mais complexos podem ser representados na forma de florestas compostas de várias árvores, cujos galhos podem representar estruturas complexas como pacotes, módulos, classes ou sub-rotinas.

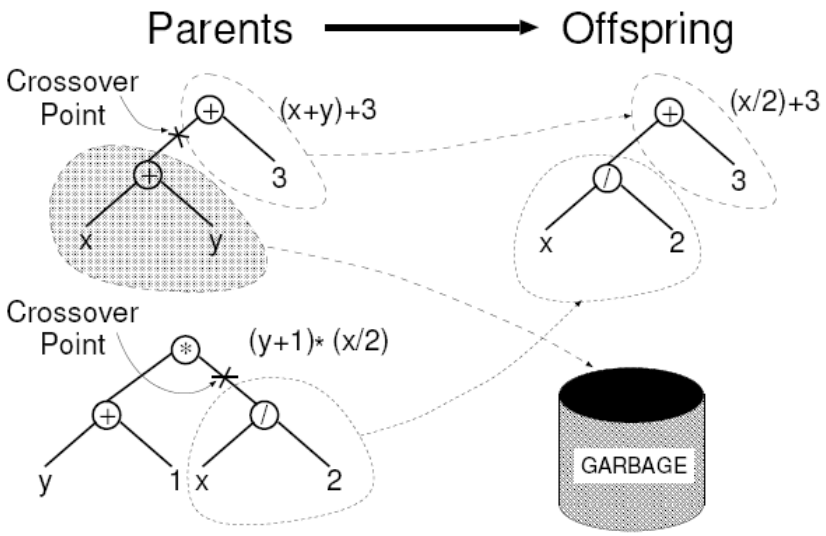
Figura 2.19. Árvore sintática da programação genética representando uma função.



Fonte: Poli et al. (2007).

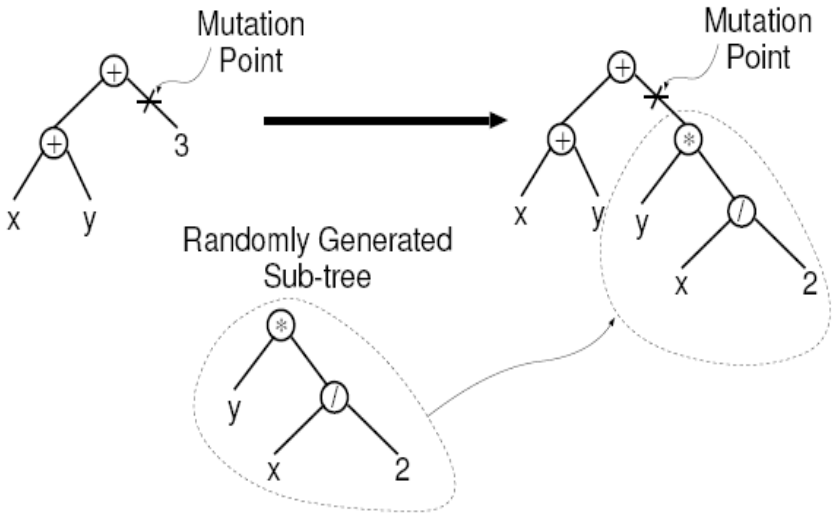
As Figuras 2.20 e 2.21 ilustram, respectivamente, casos de cruzamento e mutação aplicados a indivíduos em programação genética.

Figura 2.20. Exemplo de representação de cruzamento em árvore sintática.



Fonte: Poli et al (2007).

Figura 2.21. Exemplo de representação de mutação em árvore sintática.



Fonte: Poli et al. (2007).

Cuidados devem ser tomados ao se aplicar programação genética para evitar a geração de soluções inchadas, com complexidade desnecessária para atingir um nível razoável de aptidão. Segundo teorias apresentadas por Poli et al. (2007), a programação genética tende a gerar soluções filhas que replicam as funcionalidades das soluções pais. Além disto, expressões inchadas, com

trechos de expressões inúteis, sem efeito na saída, podem apresentar o mesmo nível de aptidão que pais mais enxutos.

A aplicabilidade de programação genética em assertivas condicionais, com a mesma estrutura de regras utilizadas para modelos comportamentais baseados em regras, é explicitamente citada por Angeline (1994). Este artigo explora características de assertivas do tipo “se <condição> então <ação1> caso contrário <ação2>” para a aplicação de programação genética para alcançar inteligência emergente. Na inteligência emergente, o software descobre soluções dinamicamente sem necessariamente informações prévias detalhadas do sistema, de tal forma que o conhecimento “emerge” como subproduto da busca por soluções.

Baseado nestas ideias, a pesquisa segue a busca por soluções para uma ferramenta com modelos autoadaptáveis para modelos comportamentais.

2.5 Adaptação de modelos de simuladores de satélites

O primeiro trabalho encontrado sobre adaptação de modelos foi Guiotto et al (2003), no qual mostrou a adaptação de modelos comportamentais a falhas simples previstas em projeto (ver Seção 2.2.)

Em Catastini et al. (2010) o simulador do tipo *end-to-end* desenvolvido para a missão GOCE da ESA é apresentado. Este projeto da antiga Alenia Spazio (atual Thales Alenia) foi desenvolvido em cooperação com a ESA (European Space Agency) para missão GOCE (Gravity field and steady-state Ocean Circulation Explorer). O simulador GOCE E2E é um simulador de alta fidelidade e pode ser utilizado acoplado com a ferramenta de análise e solução de falhas SMART-FDIR, descrita no parágrafo anterior. Este simulador utiliza modelos precisos para a descrição de comportamento de equipamentos de carga útil e plataforma. Segundo o autor, este comportamento é atualizado constantemente de acordo com a evolução do projeto. A atualização seria derivada de resultados de testes ou análises. O modelo comportamental de um simulador compatível com o sistema mencionado por Guiotto et al. (2003) deve ser atualizado constantemente ao longo das diversas fases do projeto. Esta Tese

propõe uma ferramenta para realizar esta tarefa de forma autônoma na fase em voo, sem a intervenção dos projetistas.

Mais recentemente, o artigo de Maldague e Wissler (2018) vislumbra um framework de simulação de veículo espacial multimissão durante todo o ciclo de vida, desde a concepção até o fim da missão. Trata-se de um trabalho em desenvolvimento na NASA ("*National Aeronautics and Space Administration*") para uma missão interplanetária a Europa, um dos satélites naturais de Júpiter. O artigo cita a ferramenta APGen, utilizada para modelagem e simulação, e menciona dificuldades encontradas para o processo manual de modelagem e incorporação de mudanças. A solução teria sido a elaboração da APGen DSL ("*Domain Specific Language*"), também chamado "linguagem de adaptação" pelos autores, utilizado para especificar atividades e recursos, de solo e de bordo, e suas interações. Uma parte significativa do texto descreve esforços para personalizar uma ferramenta genérica multimissão para as especificidades de uma missão particular e define os elementos a serem modelados por subsistema. O artigo ainda lista características desejáveis de um sistema de modelagem ideal ("*tweakability, evolvability, integrability and verifiability*"), sendo o segundo elemento motivador central desta Tese.

Além disso, Maldague e Wissler (2018) descrevem superficialmente, uma linguagem que permitiria a codificação de modelos comportamentais autoadaptáveis. Esta Tese propõe uma ferramenta autoadaptável para modelos porém não considera o uso de uma linguagem especial para codificação dos modelos.

2.6 Softwares autoadaptativos

Para uma visão mais geral, foram pesquisados alguns estudos de sondagem do estado da arte sobre sistemas de software autoadaptativos.

Macías-Escrivá et al. (2013) lista a computação evolutiva como uma técnica de modelagem e simulação ligada à otimização multiobjetivo. Krupitzer et al. (2015) menciona o uso de algoritmos evolutivos para abordagens de aprendizado, o que segundo o artigo estaria associado a processos de auto-otimização em relação a desempenho ou custo, mas chama a atenção para

incertezas quanto ao tempo de execução. Nenhum deles cita a programação genética pelo nome.

Segundo Macías-Escrivá et al. (2013), técnicas e ferramentas para implementação de autoadaptação em modelagem e simulação podem se resumir a oito áreas: teoria dos jogos (*game theory*), vidro de spin (*spin glasses*), análise de série temporal (*time series analysis*), lógica nebulosa (*fuzzy logic*), otimização multiobjetivo (*multiobjective optimization*), dinâmica de sistemas (*system dynamics*), dinâmica evolucional (*evolutionary dynamics*) e sistemas multiagente (*multi-agent systems*).

Para um modelo comportamental adaptativo, o uso de algoritmos evolutivos, que se enquadra na categoria de otimização multiobjetivo, apresenta características adequadas devido a semelhança entre as características encontradas nos modelos comportamentais de simuladores de satélites operacionais conhecidos e a proposta de aplicação de um tipo de algoritmo evolutivo denominado programação genética, como será visto adiante.

Pesquisas mostram controladores autoadaptativos utilizando lógica nebulosa (CALLAI et al., 2007) e sistemas de inferência baseadas em regras autoadaptativas utilizando redes neurais nebulosas (WANG & LEE, 2002) que, apesar de não estarem relacionadas a modelos comportamentais de simuladores de satélites, poderão ser eventualmente aplicadas ao problema atacado neste trabalho, após as devidas adaptações. Isto, entretanto, não fará parte do escopo deste trabalho.

Já Zhou et al. (2011) propõe uma sondagem específica referente a algoritmos evolucionários multiobjetivos e menciona trabalhos utilizando programação genética nas seguintes áreas: (i) extração de regras associada a mineração de dados, (ii) reconhecimento de padrões e processamento de imagens e (iii) mineração da web (*web mining*).

Analisando o artigo de referência que trataria de extração de regras (ZHANG & ROCKETT, 2011), verificou-se que ele descreve uma metodologia mais voltada ao reconhecimento de padrões. Entretanto, não foi encontrada nenhuma menção específica a técnicas ou ferramentas destinadas à adaptação

autônoma de modelos comportamentais de simuladores, muito menos associada a satélites artificiais.

Chiang (2010) propõe o uso de programação genética para controle autônomo. O artigo mostra uma arquitetura de um sistema de controle inteligente com programação genética, denominada GPICS. Este sistema cria regras causais do tipo "*IF-THEN*", por meio de programação genética, e descreve uma aplicação para o planejamento de rota de um robô. Ele divide a aplicação em duas fases, uma de aprendizado em que a programação genética é utilizada para a criação de regras, e outra de aplicação, onde as regras geradas são utilizadas para controle.

Regras causais do tipo "*IF-THEN*" são utilizadas nos simuladores operacionais de satélites do INPE cuja adaptação autônoma é o tema desta Tese. De forma similar a Chang (2010), esta Tese propõe o aprendizado do novo comportamento descrito pelas telemetrias discrepantes de um satélite para a geração de um novo modelo comportamental baseado em regras utilizando programação genética para sua aplicação em simulação operacional do satélite.

Em Dias et al. (2019), propõe-se uma arquitetura com uso de modelo de objeto adaptativo (AOM, "*Adaptive Object-Model*") para integrar estruturas tradicionais para serviços da Web baseadas em frameworks existentes, de maneira que a alteração da estrutura do tipo de entidade permita a criação de novos serviços e alteração dos existentes. Alterações nos serviços da Web podem ser frequentes e se beneficiariam de uma arquitetura adaptável, com capacidade de criar novos serviços e alterar os existentes.

O sistema AOM utiliza um modelo de objeto explícito que é interpretado em tempo de execução. O AOM define os objetos, seus estados, eventos e condições sob os quais um objeto muda de estado. Objetos possuem estados que alteram em resposta a eventos. O AOM define o modelo de objeto, ou seja, os objetos, seus estados, os eventos e as condições sob o qual um objeto muda de estado, de uma maneira que permite a sua modificação dinâmica. Se o modelo de objeto é alterado, o sistema altera seu comportamento.

(REVAULT & YODER, 2001). Na arquitetura proposta nesta Tese (Capítulo 3), uma ferramenta externa dota um simulador com modelagem não compatível com AOM, mas com capacidade de autoadaptação.

Para Ferreira et al. (2008), no AOM o modelo de objeto abrange a especificação completa de objetos, estados, eventos, condições, restrições e regras de negócios. Eles descrevem métodos para rastrear, versionar e evoluir informações nos vários níveis de abstração que podem existir no AOM. A alteração do modelo resulta imediatamente em um novo modelo de domínio de negócios, diferente do original, em termos de dados e metadados. Metadados seriam, segundo um dicionário (MERRIAM-WEBSTER, 2020), “dados que proveem informação sobre outros dados”. De acordo com Baca (2016), metadados providenciam meios para indexação, acesso, preservação e descoberta de recursos digitais.

Segundo Welicki et al. (2009) o AOM permite que usuários, que podem não ser programadores, possam mudar metadados armazenados externamente sempre que quiserem mudar as definições das entradas de domínio. Toda vez que as definições armazenadas externamente forem modificadas, o sistema pode refletir estas mudanças sem recompilação da aplicação. Welicki et al. (2009) descrevem então uma implementação de um “*AOM Builder*” genérico, que seria capaz de construir diferentes tipos de entidades, dirigido por metadados de entidades e seus processos.

Com relação ao uso de metadados na área espacial, no padrão Simulator Model Portability Standard 2 (SMP2), o formato de dados de metamodelos é especificado em ECSS (2011b) e os componentes do modelo em (ECSS, 2011c).

Referente à capacidade de adaptação de modelos de um simulador de satélite, Sebastião et al. (2008) mostram como o simulador para a missão CryoSat foi usado como base para o desenvolvimento de simuladores para outras missões, evidenciando a capacidade de adaptação do modelo nas fases de desenvolvimento para diferentes usos, conforme Figura 2.2 (Seção 2.1). Enquanto, Barreto et al. (2010) mostram que a adaptação de modelos do

SIMCBERS para compatibilizá-lo com o SMP2 poderia ser realizada com pouco trabalho, estando a dificuldade maior na implementação de interfaces de comunicação entre as partes que compõem os modelos. Azevedo et al. (2012) analisam a aplicabilidade do SMP2 para o desenvolvimento futuro de simuladores do INPE (baseando-se nas plataformas do SIMCBERS e o SISAO), em diferentes cenários de desenvolvimento para estimativa de custos na forma de esforço e retrabalho.

Esta Tese propõe adaptação de modelos baseados em regras genéricas, que foram criadas sem o cuidado de permitir sua adaptação futura, para imbuir a capacidade de autoadaptação a simuladores de satélites criados sem o conceito de AOM ou SMP, mas que utilizam metadados para a configuração de modelos comportamentais baseados em regras durante a sua execução da fase de operação.

2.7 Comparações entre os trabalhos relacionados e esta tese

A Tabela 2.1 sumariza os trabalhos relacionados abordados neste Capítulo e faz uma breve comparação com esta Tese.

Tabela 2.1. Trabalhos relacionados e comparação com esta Tese.

Trabalho relacionado	**Esta Tese**
(CBERS, 2012) (CBERS, 2016c) SIMCBERS é um simulador operacional em software baseado em regras.	Propõe autoadaptação para modelos como deste simulador.
(AMAZONIA, 2017) (AMAZONIA, 2019) SISAO é um simulador "*hardware-in-the-loop*" com modelo parcialmente baseado em regras.	Propõe autoadaptação para modelos de software como deste simulador.
(TOMINAGA, 2010) SATSIM é um simulador em software baseado em regras.	Propõe autoadaptação de modelos como deste simulador.
(KANG et al., 1995) ARTSS é um simulador em software com "*hardware- in-the-loop*"	Propõe autoadaptação para modelos de software como deste simulador.
(MALDAGUE & WISSLER, 2018) APGen DSL é uma linguagem que permite codificação de modelos comportamentais autoadaptáveis	Esta Tese propõe autoadaptação em operação de modelos criados sem preocupação de adaptação.
(GUIOTTO et al., 2003) SMART-FDIR altera o modelo de uma configuração nominal para uma configuração de falha prevista por meio de FDIR.	Propõe autoadaptação de modelos inclusive para falhas não previstas em FDIR via programação genética.
(CATASTINI et al., 2010) Simulador GOCE E2E utiliza SMART-FDIR com modelo de alta fidelidade adaptado por projetistas ao longo das fases de projeto.	Propõe autoadaptação para modelo na fase operacional sem intervenção de projetistas.
Sobre softwares auto adaptativos (ZHOU et al., 2011) (ZHANG & ROCKETT, 2011) (MACÍAS-ESCRIVÁ et al., 2013) (KRUPITZER et al., 2015) não apontam para programação genética em modelos comportamentais de simuladores.	Propõe autoadaptação com programação genética para alteração de expressões de regras em modelos comportamentais baseados em regras.
(CHIANG, 2010) GPICS é um sistema de controle para automação robótica que cria regras causais do tipo "IF-THEN" por meio de programação genética.	Propõe criação de regras causais alternativas do tipo "IF-THEN" para modelo comportamental de simulador por meio de programação genética.

continua

Tabela 2.1. Conclusão.

Trabalho relacionado	Esta Tese
(DIAS et al., 2019) Arquitetura com uso de AOM para serviços da Web baseadas em frameworks existentes, de maneira que a alteração da estrutura do tipo de entidade permita a criação de novos serviços e alterar os existentes.	Propõe autoadaptação de modelos de simulador pré-existentes utilizando parâmetros e regras como base para alterações. Não usa o conceito de serviços
(REVAULT & YODER, 2001) define o modelo de objeto de uma maneira que permite a modificação dinâmica do comportamento usando AOM.	Propõe autoadaptação dinâmica de modelos criados sem AOM.
(FERREIRA et al., 2008) AOM usa dados e metadados para armazenar especificação completa de objetos, estados, eventos, condições, restrições e regras.	Propõe autoadaptação de modelos de simuladores que armazenam informações de dados e metadados em XML.
(WELICKI et al. 2009) AOM permite que usuários não programadores mudem metadados. Pode refletir mudanças sem recompilação.	Propõe autoadaptação de modelos de simulador sem recompilação.
(SEBASTIÃO et al., 2008) SMP2 especifica metadados (ECSS, 2011b) e componentes de modelos (ECSS, 2011c), que podem ser desenvolvidos utilizando a abordagem AOM.	Propõe autoadaptação de modelos de simuladores genéricos sem conceito de AOM. Não se baseia no SMP.

Fonte: Produção do autor.

A Tabela 2.2 compara e evidencia diferenças explicitadas das abordagens e métodos de adaptação entre os trabalhos relacionados e esta Tese.

Tabela 2.2. Comparação de abordagens e métodos de adaptação.

	Adaptação por metadados	**Adaptação em tempo de execução**	**Adaptação em tempo de projeto**	**Adaptação por algoritmos evolutivos**
SIMCBERS (CBERS, 2012) (CBERS, 2016c)	X		X	
SISAO (AMAZONIA, 2017) (AMAZONIA, 2019)	X		X	
SATSIM (TOMINAGA, 2010)	X	X		
ARTSS (KANG et al., 1995)				
APGen DSL (MALDAGUE & WISSLER, 2018)		X	X	
SMART-FDIR (GUIOTTO et al., 2003)		X	X	
GOCE E2E (CATASTINI et al., 2010)		X	X	
(ZHOU et al., 2011) (ZHANG & ROCKETT, 2011) (MACÍAS-ESCRIVÁ et al., 2013) (KRUPITZER et al., 2015)		X		X
GPICS (CHIANG, 2010)		X		X
AOM (DIAS et al., 2019) (REVAULT & YODER, 2001) (FERREIRA et al., 2008) (WELICKI et al. 2009)	X	X	X	
SMP2 (ECSS, 2011b) (ECSS, 2011c) (SEBASTIÃO et al., 2008)	X		X	
Esta Tese	**X**	**X**		**X**

Fonte: Produção do autor.

2.8 Considerações

Neste Capítulo foram analisadas arquiteturas de simuladores operacionais de satélites, falhas embarcadas típicas que podem ocorrer no ambiente espacial, bem como softwares autoadaptativos e algoritmos evolutivos, com ênfase na programação genética visando fazer uma comparação com os conceitos e trabalhos relacionados a esta Tese.

Os simuladores SIMCBERS, SISAO e SATSIM são simuladores de satélites operacionais com modelos em software baseados em regras, com arquiteturas

contendo um núcleo de processamento multimissão e modelos comportamentais específicos por missão com codificação fixa e não autoadaptável. Todos utilizam metadados em formato XML para especificação de parâmetros de simulação e regras de seus modelos comportamentais, o que os torna fáceis de alterar, mas não em tempo de execução. No SIMCBERS, o foco da modelagem estava no reuso do núcleo de processo multimissão e o uso de modelos derivados para missões futuras, como o SISAO. A proposta do SATSIM previa alguma capacidade de adaptação do modelo durante a operação do satélite e de seu simulador, mas executada pelo usuário, manualmente. Já o ARTSS é um simulador híbrido com modelos parcialmente em software e parcialmente em hardware, sendo que a parte em software pode ser caracterizada por meio de regras.

Outros trabalhos que descrevem adaptação de modelos para simuladores de satélites discutem abordagens para criação de modelos adaptáveis. O foco destes trabalhos está mais voltado ao desenvolvimento de simuladores para novas missões, do que para a adaptação de modelos preexistentes para missões em operação como propõe esta Tese.

Estudos de sondagem pesquisados mencionam o uso da abordagem AOM para adaptação de modelos comportamentais em tempo de projeto, inclusive para a aplicação espacial. Esta Tese propõe uma arquitetura e um processo para adaptação autônoma de modelos comportamentais de simuladores baseados em regras desenvolvidos sem a abordagem AOM utilizando uma ferramenta externa para análise e modificação do modelo.

3 PROCESSO E ARQUITETURA

Este Capítulo apresenta o processo para adaptação autônoma de modelos comportamentais de um simulador operacional de satélites artificiais e a arquitetura proposta para um Sistema de Simulação que apoia a execução do processo proposto. Por razões de clareza, primeiramente apresenta-se a arquitetura e seus componentes essenciais para o chamado “Sistema de Simulação” necessários para realização do processo, e em seguida as atividades do processo.

3.1 Arquitetura do sistema de simulação

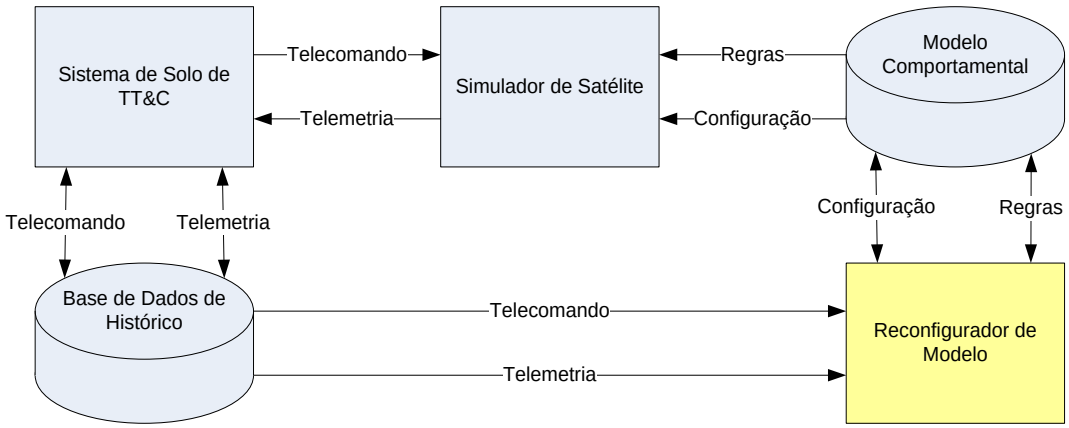
O Sistema de Simulação com adaptação autônoma, proposto nesta Tese, possui um núcleo de processamento multimissão denominado Simulador de Satélite e um modelo específico para cada satélite. Para a aplicação multimissão, a modelagem pode ser simples, com a descrição do comportamento de todo o satélite em uma única estrutura de dados conforme mostrado na Seção 5.1 ou, alternativamente, ela pode ser composta por estruturas de dados parciais em que cada subsistema ou equipamento é modelado separadamente, conforme a estratégia adotada no SIMCBERS (CBERS, 2016c). Os modelos são chamados Modelos Comportamentais, pois representam o comportamento dinâmico do objeto simulado, e são especificados com base em regras conforme descrito no Capítulo 2.

A descrição da arquitetura do Sistema de Simulação com adaptação autônoma mostra o raciocínio sobre as estruturas lógicas organizadas em componentes de software e fluxos de informações que solucionam o problema de adaptação autônoma de modelos comportamentais no contexto apresentado.

A arquitetura proposta, em uma visão de alto nível relacionando todos os elementos envolvidos no uso de um Simulador Operacional como apoio à operação e ao controle de satélites, é composta pelos componentes: Simulador de Satélite, Sistema de Solo de TT&C, Base de Dados de Histórico, Reconfigurador de Modelo, e o banco de dados denominado Modelo Comportamental; e pelos seguintes fluxos de informações: Telemetria, Telecomando, Regras e Configuração, conforme ilustrada na Figura 3.1. Nesta

arquitetura as Regras e os valores para Configuração do Simulador de Satélite são armazenados no banco de dados Modelo Comportamental e acessadas pelo Simulador de Satélite durante a inicialização.

Figura 3.1. Arquitetura do Sistema de Simulação com adaptação autônoma.



Fonte: Produção do autor.

O Simulador de Satélite interage com o Sistema de Solo de TT&C, do qual recebe dados de Telemetria gerados pelo satélite em operação e recebe dados de Telecomando, quando necessário. O Sistema de Solo de TT&C possui uma Base de Dados de Histórico contendo todas as telemetrias recebidas e telecomandos enviados durante as passagens do satélite pela estação terrena.

A característica de adaptação autônoma é implementada pelo componente externo ao Simulador de Satélite, chamado Reconfigurador de Modelo**,** o qual detecta a necessidade das modificações que devem ser aplicadas ao Modelo Comportamental para adaptá-lo as novas realidades que afetam o satélite.

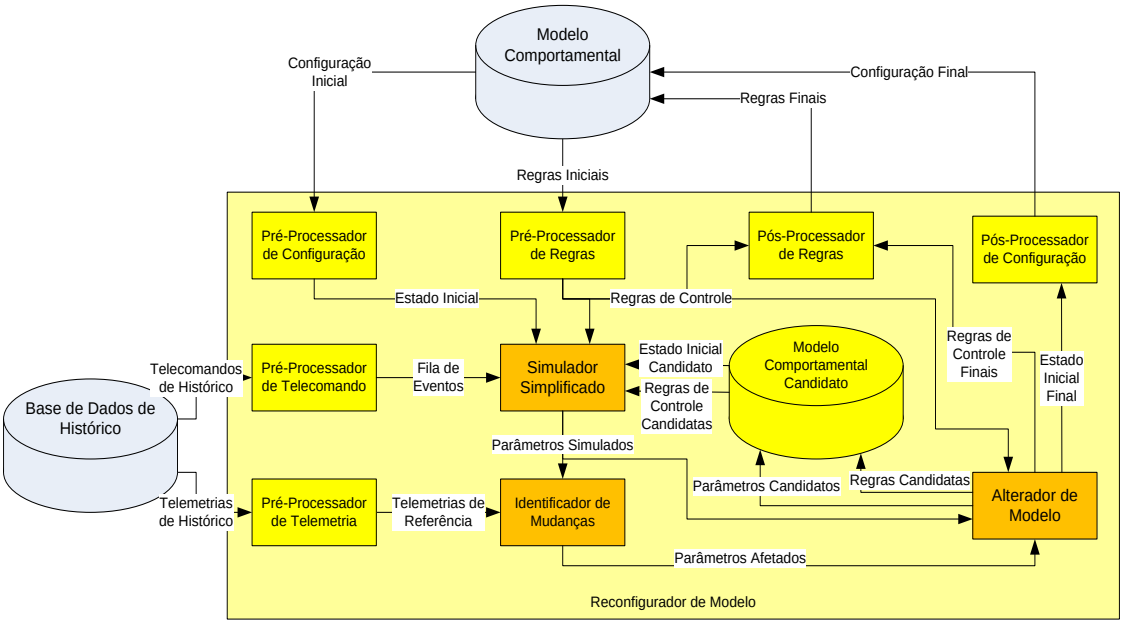
O Reconfigurador de Modelo conecta-se com o banco Modelo Comportamental, associado ao Simulador de Satélite, para receber dados de Configuração com valores de inicialização de parâmetros e Regras com expressões de precondições e efeitos que caracterizam a dinâmica dos parâmetros ao longo da sessão de simulação, e com a Base de Dados de Histórico, associada ao Sistema de Solo de TT&C, para recuperar dados de Telecomando a serem enviados ao Simulador e recuperar valores de Telemetria a serem comparados com os valores das telemetrias produzidas pelo Simulador de Satélite. O Reconfigurador de Modelo identifica alterações

nas telemetrias do satélite para decidir quando atualizar o Modelo Comportamental utilizado pelo Simulador de Satélite. Caso sejam detectadas discrepâncias entre os valores de Telemetria do satélite e os parâmetros simulados, correspondentes às telemetrias geradas pelo Modelo Comportamental, é disparado um processo de atualização do Modelo Comportamental, que busca ajustar os parâmetros de Configuração ou expressões de Regras para compatibilizá-los com os valores de Telemetria recebidos do satélite.

A atualização de um modelo comportamental consta na geração de novos modelos, chamados Modelos Comportamentais Candidatos, através de alterações nas expressões de Regras e de alteração de valores de parâmetros de simulação que constituem a Configuração. A versão final de um modelo comportamental será obtida pela substituição das Regras ou da Configuração que constituem a solução candidata cujas saídas de Telemetria simulada melhor se ajustam aos valores obtidos dos parâmetros de Telemetria vindas da Base de Dados de Histórico, para uma mesma entrada de dados de Telecomando.

O Reconfigurador de Modelo é o componente mais importante da arquitetura geral e por isso tem sua arquitetura definida por um conjunto de componentes, conforme ilustra a Figura 3.2. Os componentes do Reconfigurador de Modelo são quatro pré-processadores, um para cada uma das entradas (Telemetria, Telecomando, Configuração e Regras), dois pós-processadores de saída (o de Regras e o de Configuração), o Simulador Simplificado, o Identificador de Mudanças, o Alterador de Modelos e a banco de dados chamado Modelo Comportamental Candidato.

Figura 3.2. Arquitetura do Reconfigurador de Modelo.



Fonte: Produção do autor.

Os pré-processadores e pós-processadores têm como objetivo implementar interfaces com a Base de Dados de Histórico e o bando de dados Modelo Comportamental. O **Simulador Simplificado** é responsável por gerar valores de Parâmetros Simulados a partir de um Estado Inicial, de uma Fila de Eventos e de um conjunto de Regras de Controle. O **Identificador de Mudanças** compara estes Parâmetros Simulados com as Telemetrias de Referência correspondentes. Se mudanças de comportamento forem identificadas pelo Identificador de Mudanças, uma lista de Parâmetros Afetados é gerada e o **Alterador de Modelo** é disparado. O Alterador de Modelo gera listas de Parâmetros Candidatos e Regras Candidatas. Estados Iniciais Candidatos e conjuntos de Regras de Controle Candidatas são gerados a partir de Parâmetros Candidatos e de Regras Candidatas, respectivamente, que são encaminhadas ao Simulador Simplificado para a geração de novos Parâmetros Simulados para comparação com a Telemetria de Referência. Se uma solução composta por um Estado Inicial Candidato ou um conjunto de Regras Candidatas for considerada compatível com o comportamento indicado pelas Telemetrias de Referência, por meio de um critério de seleção definido por um conjunto vazio de Parâmetros Afetados, o Modelo Comportamental é

atualizado por meio dos pós-processadores para a substituição da Configuração Final ou das Regras Finais no novo modelo comportamental.

A Tabela 3.1 descreve os componentes da arquitetura do Reconfigurador de Modelo e os fluxos de informações, listando-os em ordem alfabética.

Tabela 3.1. Componentes da arquitetura do Reconfigurador de Modelo**.**

Componente	**Descrição**
Alterador de Modelo	Componente que analisa os Parâmetros Afetados e obtém Parâmetros Candidatos e Regras Candidatas.
Base de Dados de Histórico	Base de dados do Sistema de Solo de TT&C que contém informações de Telecomando e de Telemetria de Histórico.
Configuração Final	Arquivo de parâmetros de inicialização do Simulador exportados para o Modelo Comportamental após a adaptação.
Parâmetros Candidatos	Subconjunto de parâmetros que compõem o Estado Inicial Candidato cujos valores são modificados pelo Alterador de Modelo para tentar obter soluções.
Parâmetros Afetados	Subconjunto de parâmetros que compõem os Parâmetros Simulados que foram identificados como discrepantes quando comparados com as Telemetrias de Referência.
Parâmetros Simulados	Saída do Simulador Simplificado que contém os valores de parâmetros do Estado Inicial resultantes da execução das Regras de Controle e da aplicação da Fila de Eventos.
Pós-Processador de Configuração	Componente que obtém a Configuração Final a partir do Estado Inicial Final.
Pós-Processador de Regras	Componente que obtém as Regras Finais a partir do conjunto de Regras de Controle Finais.
Configuração Inicial	Arquivo de parâmetros de inicialização do Simulador importados do Modelo Comportamental antes da adaptação.
Estado Inicial	Arquivo de parâmetros de inicialização do Simulador Simplificado com valores obtidos da Configuração Inicial.
Estado Inicial Candidato	Arquivo de parâmetros de inicialização do Simulador Simplificado com valores obtidos a partir de Parâmetros Candidatos a serem testados.

continua

Tabela 3.1. Continuação.

Componente	**Descrição**
Fila de Eventos	Arquivo de parâmetros de inicialização do Simulador Simplificado com valores obtidos do Telecomando de Histórico.
Identificador de Mudanças	Componente que obtém Parâmetros Afetados a partir da comparação entre Parâmetros Simulados e Telemetrias de Referência.
Modelo Comportamental	Base de dados do Simulador de Satélite que contém informações de Configuração Inicial e Regras Iniciais.
Modelo Comportamental Candidato	Base de dados do Simulador de Satélite Simplificado que contém informações do Estado Inicial Candidato geradas a partir dos Parâmetros Candidatos e das Regras de Controle Candidatas geradas a partir das Regras Candidatas.
Pré-Processador de Configuração	Componente que obtém o Estado Inicial a partir da Configuração Inicial.
Pré-Processador de Regras	Componente que obtém o Estado Inicial a partir da Configuração Inicial.
Pré-Processador de Telecomando	Componente que obtém a Fila de Eventos a partir dos Telecomandos de Histórico.
Pré-Processador de Telemetria	Componente que obtém as Telemetrias de Referência a partir das Telemetrias de Histórico.
Regras Candidatas	Subconjunto de regras que compõem as Regras de Controle Candidatas cujas expressões são modificadas pelo Alterador de Modelo para tentar obter soluções.
Regras de Controle	Arquivo de regas do Simulador Simplificado com expressões obtidas a partir das Regras Iniciais.
Regras de Controle Candidatas	Arquivo de regas do Simulador Simplificado com expressões obtidas a partir das Regras Candidatas a serem testadas.
Regras de Controle Finais	Arquivo de regas do Simulador Simplificado com expressões obtidas a partir das Regras Candidatas testadas que resultaram em solução.
Regras Finais	Arquivo de regras exportados do Modelo Comportamental que descrevem a sua nova dinâmica após a adaptação.
Regras Iniciais	Arquivo de regras importados do Modelo Comportamental que descrevem a sua dinâmica antes da adaptação.
Simulador Simplificado	Componente que obtém Parâmetros Simulados a partir do Estado Inicial, das Regras de Controle e da Fila de Eventos.

continua

Tabela 3.1. Conclusão.

Componente	Descrição
Telecomandos de Histórico	Arquivo de parâmetros de telecomandos da Base de Dados de Histórico utilizado para a geração da Fila de Eventos.
Telemetrias de Histórico	Arquivo de parâmetros de telemetrias da Base de Dados de Histórico utilizado para a geração das Telemetrias de Referência.
Telemetrias de Referência	Arquivo de parâmetros de telemetrias cujos valores servem de referência para comparação com os valores de parâmetros correspondentes dos Parâmetros Simulados.

Fonte: Produção do autor.

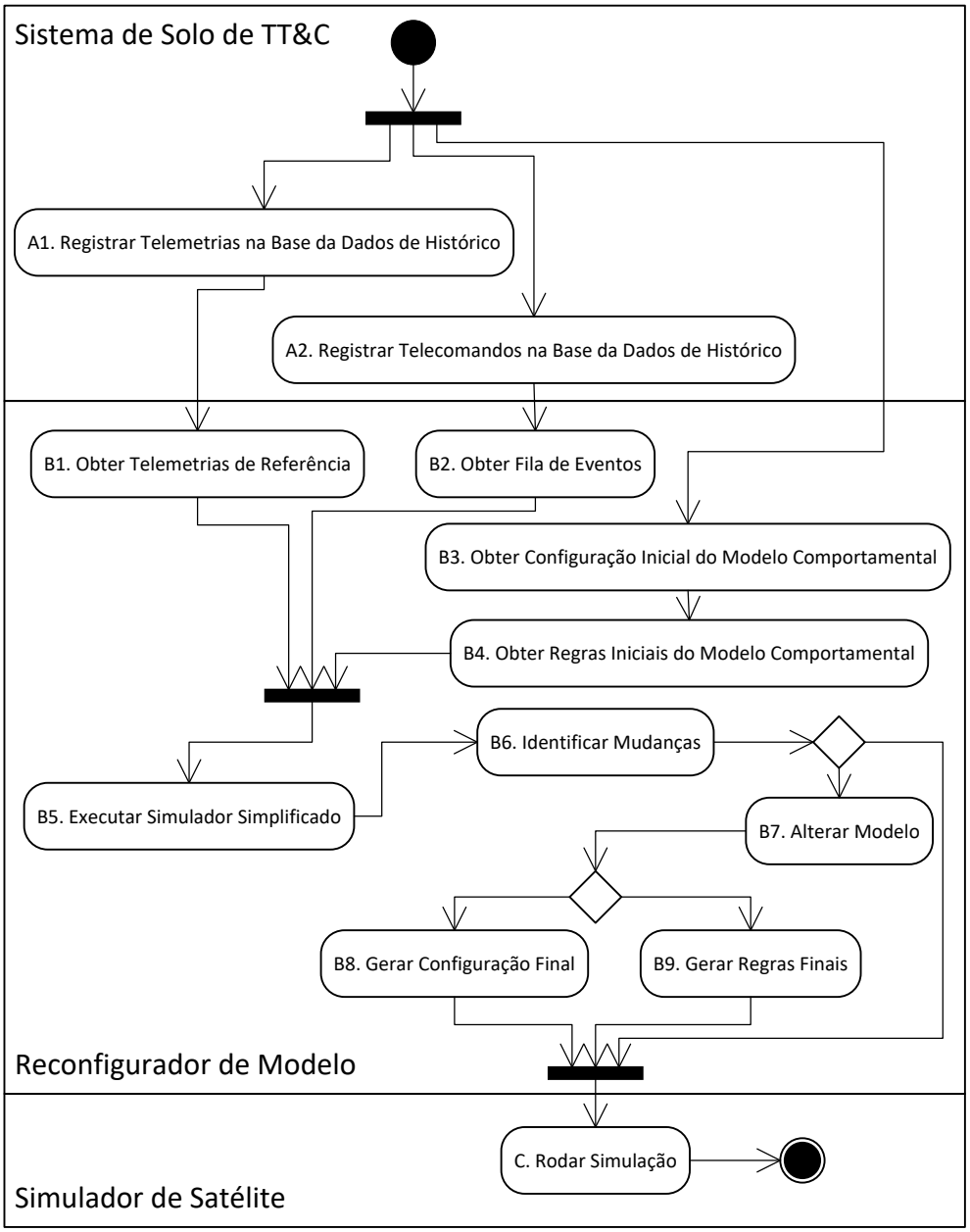
3.2 Processo de adaptação do modelo comportamental

Esta Seção descreve o processo de adaptação autônoma do Modelo Comportamental do Simulador, o qual consta de um conjunto de atividades para se obter um modelo comportamental adaptado para o Simulador de Satélite em conformidade com a arquitetura apresentada na Figura 3.1 da Seção 3.1. O Modelo Comportamental deve ser adaptado para tornar compatíveis os Parâmetros Simulados com as Telemetrias de Referência, por meio de uma nova Configuração Final ou um novo conjunto de Regas Finais.

A arquitetura do Sistema de Simulação com reconfiguração autônoma prevê a inclusão do Reconfigurador de Modelo para autoadaptar o Modelo Comportamental do Simulador de Satélite antes de cada execução. Caso não sejam identificadas diferenças entre a Telemetria de Referência e Parâmetros Simulados, o Modelo Comportamental original é mantido. Caso contrário, ele é modificado para tornar os Parâmetros Simulados compatíveis com a Telemetria de Referência.

A Figura 3.3 mostra o diagrama de atividades do Sistema de Simulação. Nela são mostradas as interações do Reconfigurador de Modelo com o Sistema de Solo de TT&C e com o Simulador de Satélite, por meio das interfaces de Telemetria e Telecomando da Base de Dados de Histórico e das interfaces de Configuração e Regras do Modelo Comportamental, respectivamente.

Figura 3.3. Diagrama de atividades do Sistema de Simulação.



Fonte: Produção do autor.

O processo inclui uma análise dos Parâmetros Afetados para a determinação de correlação entre regras e parâmetros associados, que permitirão a identificação de Parâmetros Candidatos e a redução do espaço de busca na aplicação da programação genética às Regras Candidatas. Com isto visa-se evitar o uso direto de programação genética sobre o conjunto completo das Regras de Controle, que teria um alto custo computacional. As Regras de

Controle Candidatas consistem em um conjunto reduzido de regras que compõem as Regras de Controle obtidas a partir das Regras Iniciais.

O processo do **Sistema de Simulação** para adaptação autônoma do Modelo Comportamental do Simulador de Satélite consiste das seguintes atividades:

A. Executar o Sistema de Solo de TT&C para

   1) Registrar Telemetrias recebidas na Base de Dados de Histórico

   2) Registrar Telecomandos enviados na Base de Dados de Histórico

B. Executar o Reconfigurador de Modelo para

   1) Obter lista de valores de Telemetrias de Histórico da Base de Dados de Histórico do Sistema de Solo de TT&C e sua conversão para Telemetrias de Referência

   2) Obter lista de Telecomandos de Histórico enviados da Base de Dados de Histórico do Sistema de Solo de TT&C e sua conversão para Fila de Eventos

   3) Obter a Configuração Inicial do Modelo Comportamental do Simulador de Satélite e sua conversão para Estado Inicial

   4) Obter o conjunto de Regras Iniciais do Modelo Comportamental do Simulador de Satélite e sua conversão para Regras de Controle

   5) Obter Parâmetros Simulados gerados pela execução do Simulador Simplificado utilizando o Estado Inicial, as Regras de Controle e a Fila de Eventos que representam o Modelo Comportamental corrente

   6) Verificar diferenças entre os valores das Telemetrias de Referência e valores dos Parâmetros de Simulação correspondentes para a geração de Parâmetros Afetados

7) Gerar Modelos Comportamentais Candidatos compostos por um Estado Inicial Candidato, Regras de Controle Candidatas e a Fila de Eventos original para teste de aptidão visando à escolha de uma solução

8) Gerar uma solução que consiste em uma Configuração Final para compor o novo Modelo Comportamental do Simulador de Satélite

   OU

9) Gerar uma solução que consiste em um conjunto de Regras Finais para compor o novo Modelo Comportamental do Simulador de Satélite.

C. Executar o Simulador de Satélite

   1) Rodar Simulação com o Modelo Comportamental adaptado

O Modelo Comportamental adaptado é constituído pelo Modelo Comportamental original, cuja validade se dá desde o início da operação até o instante de identificação da primeira discrepância, e Modelos Comportamentais alternativos gerados de forma incremental como solução da busca por compatibilidade entre Parâmetros Simulados e a Telemetria de Referência para intervalos de tempo durante o qual são válidos, ou seja, até a identificação da próxima discrepância.

As atividades do Reconfigurador de Modelo são descritas a seguir:

1) Processar Telemetrias de Referência - obtém Telemetrias de Referência a partir das Telemetrias de Histórico da Base de Dados de Histórico (Pré-Processador de Telemetria)

2) Processar Fila de Eventos - obtém a Fila de Eventos a partir de Telecomandos de Histórico da Base de Dados de Histórico (Pré-Processador de Telecomandos)

3) Processar Configuração - obtém o Estado Inicial a partir de parâmetros da Configuração Inicial (Pré-Processador de Configuração)

4) Processar Regras - obtém o conjunto de Regras de Controle a partir das Regras Iniciais do Modelo Comportamental do Simulador de Satélite (Pré-Processador de Regras)

5) Executar Simulador Simplificado – gera Parâmetros de Simulação.

6) Identificar Mudanças - verifica mudanças entre Telemetrias de Referência e o Modelo Comportamental original, encontrando discrepâncias entre Parâmetros de Simulação e Telemetrias de Referência e determinando os Parâmetros Afetados

7) Alterar Modelo - executa em loop para obtenção de soluções gerando Modelos Comportamentais Candidatos para teste

    a) Correlacionar Regras e Parâmetros - analisa correlação entre expressões de Regras de Controle e a lista de Parâmetros Afetados

    b) Obter Parâmetros Candidatos - por atribuição de valores alternativos a parâmetros correlacionados com Parâmetros Afetados

    c) Obter Regras Candidatas - por atribuição de expressões alternativas a regras correlacionadas com Parâmetros Afetados

    d) Alterar - Modelos Comportamentais Candidatos por meio de

        1. Parâmetros Candidatos - compostos por Estados Iniciais Candidatos OU

        2. Regras Candidatas - compostos por Regras de Controle Candidatas

e) Executar Simulador Simplificado - para geração de novos Parâmetros de Simulação

    1. Com a Fila de Eventos original, as Regras de Controle original e um Estado Inicial Candidato, OU

    2. Com a Fila de Eventos original, o Estado Inicial original e Regras de Controle Candidatas

f) Identificar Mudanças - para novos Parâmetros de Simulação e a Telemetria de Referência

    1. Para Parâmetros Candidatos, OU

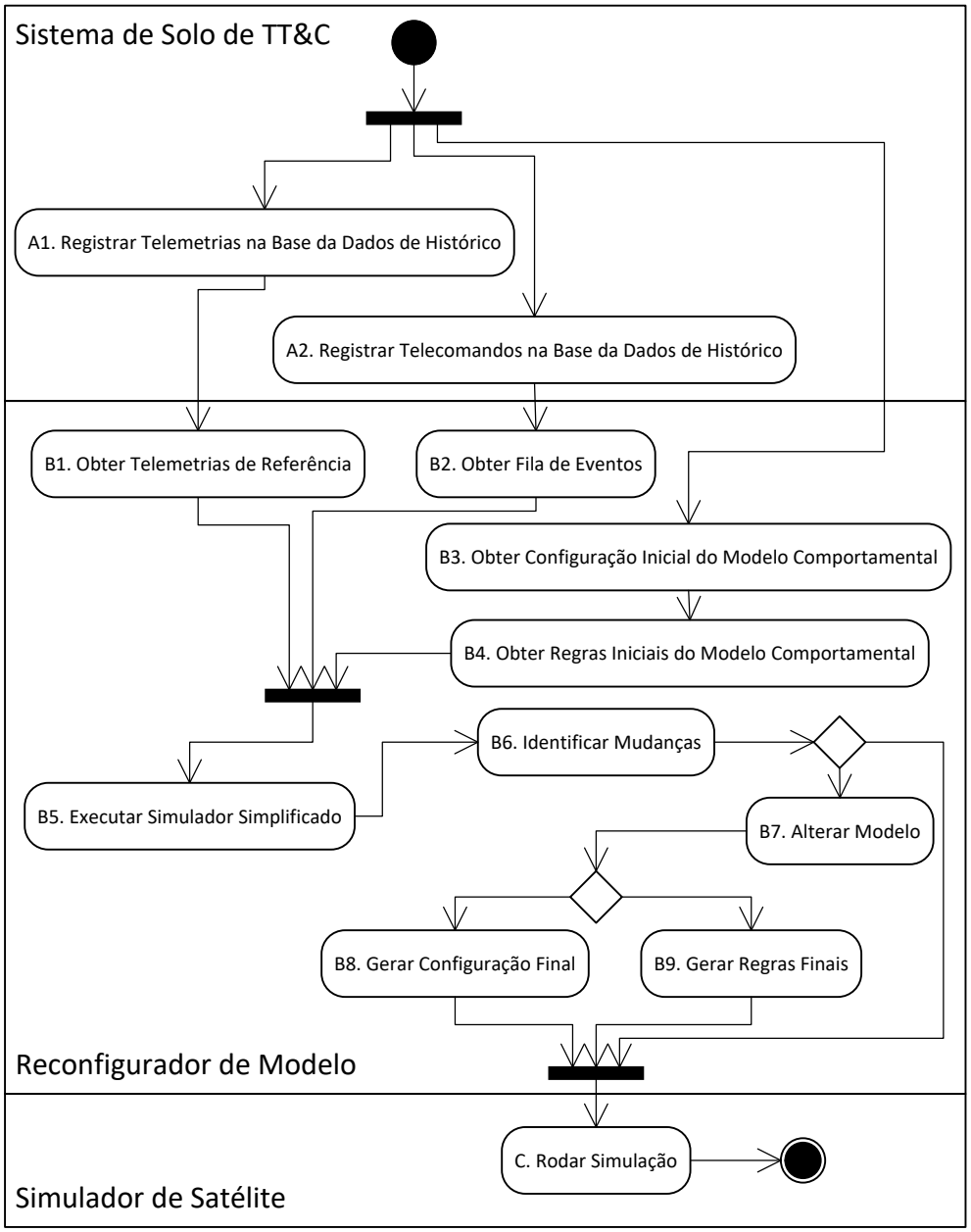
    2. Para Regras Candidatas

8) O Pós-Processador de Configuração - exporta parâmetros do Estado Inicial Final que constitui uma Configuração Final alternativa para o Modelo Comportamental do Simulador de Satélite, ou

9) O Pós-processador de Regras - exporta Regras de Controle Finais que constituem um conjunto de Regras Finais alternativo para o Modelo Comportamental do Simulador de Satélite.

A Figura 3.4 mostra o diagrama de atividades do Reconfigurador de Modelo proposto, que sumariza o comportamento dinâmico descrito.

Figura 3.4. Diagrama de atividades do Reconfigurador de Modelo.



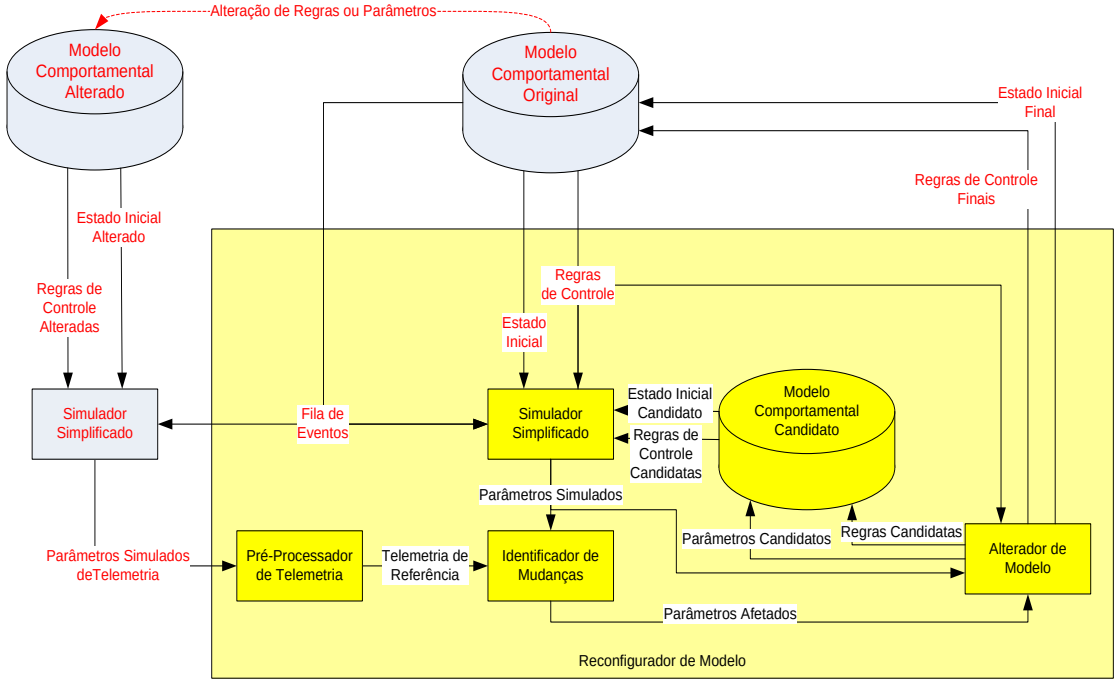
Fonte: Produção do autor.

Na Figura 3.4, a atividade 7c (“Obter Regras Candidatas com Programação Genética”) é onde se aplica a heurística de programação genética apresentada na seção 4.2.2, enquanto que o algoritmo de correlação, descrito na seção 4.3, é executado na atividade 7a (“Correlacionar Regras e Parâmetros”).

3.3 Protótipo

A fim de validar a arquitetura proposta para o Reconfigurador de Modelo, foi criado um protótipo, com funcionalidade reduzida, porém representativa. Sua arquitetura é mostrada na Figura 3.5.

Figura 3.5. Arquitetura do protótipo do Reconfigurador de Modelo.



Fonte: Produção do autor.

O Protótipo do Reconfigurador de Modelo utiliza como entradas um Modelo Comportamental Original configurado para um Simulador Simplificado e Parâmetros Simulados de Telemetria que substituem a Telemetria de Histórico. Parâmetros Simulados de Telemetria são gerados por um Simulador Simplificado executado com o Modelo Comportamental Alterado, criado a partir de alteração de parâmetros de Estado Inicial para gerar o Estado Inicial Alterado e/ou alterações expressões das Regras de Controle do Modelo Comportamental Original para gerar as Regras de Controle Alteradas e uma Fila de Eventos que representa o Telecomando de Histórico. O Protótipo do Reconfigurador de Modelo é capaz de gerar soluções na forma de Estado Inicial Final e/ou Regras Finais para adaptar o Modelo Comportamental Original

com as alterações compatíveis, idealmente idênticas, às introduzidas para a geração do Modelo Comportamental Alterado.

O Protótipo do Reconfigurador de Modelo foi desenvolvido em linguagem Python. A composição do Reconfigurador em termos de módulos, funções e sub-rotinas é descrita no Apêndice A.

Foram usados os softwares SATCS e o SATSIM respectivamente como Sistema de Solo de TT&C e como Simulador de Satélite. As interfaces com o SATCS e com o SATSIM, bem como as entradas e saídas do Reconfigurador de Modelo e o arquivo adicional para permitir o mapeamento de telemetrias e parâmetros de simulação correspondentes são detalhadas no Apêndice B.

3.4 Considerações

Esta Seção apresentou a proposta de arquitetura e um processo para adaptar de forma autônoma o modelo comportamental de um simulador operacional de satélite, descrevendo os principais componentes da arquitetura e a sequência de atividades do processo de adaptação.

Um protótipo simplificado da solução foi desenvolvido demonstrando a capacidade de geração de resultados esperados. O protótipo utilizou um modelo comportamental de teste composto por um conjunto de regras simplificadas.

4 ADAPTAÇÃO AUTÔNOMA

Neste Capítulo são apresentados o funcionamento dos modelos baseados em regras (conforme descrito no Capítulo 2), a heurística da programação genética adotada para adaptação autônoma de modelos comportamentais em simuladores operacionais de satélites artificiais através de um exemplo ilustrativo.

A Seção 4.1 apresenta o funcionamento de modelos baseados em regras, sua caracterização em parâmetros e o mecanismo de alteração das regras quando mudanças de operação são necessárias, bem como, quando falhas típicas de um equipamento acontecem no satélite em voo, descritas na Seção 2.2. O modelo comportamental é ilustrado com um equipamento de satélite com módulos principal e redundante e regras de ligamento de unidade principal, ao qual são aplicadas falhas ilustrativas. Em seguida, são apresentados exemplos de regras comportamentais e alterações no modelo para a obtenção de regras candidatas capazes de gerar resultados de simulação compatíveis com os valores de referência de telemetrias do satélite.

A Seção 4.2 descreve a heurística da programação genética para a geração de regras candidatas para o modelo, atribuindo a característica autoadaptativa ao modelo. A representação das regras de comportamento na forma de árvore sintática, utilizada na programação genética, é apresentada.

Por fim, a Seção 4.3 mostra a abordagem por análise de correlação para a obtenção de novos valores de parâmetros candidatos, compatíveis com telemetrias de referência. Esta abordagem reduz o espaço de busca gerado pelo uso de programação genética.

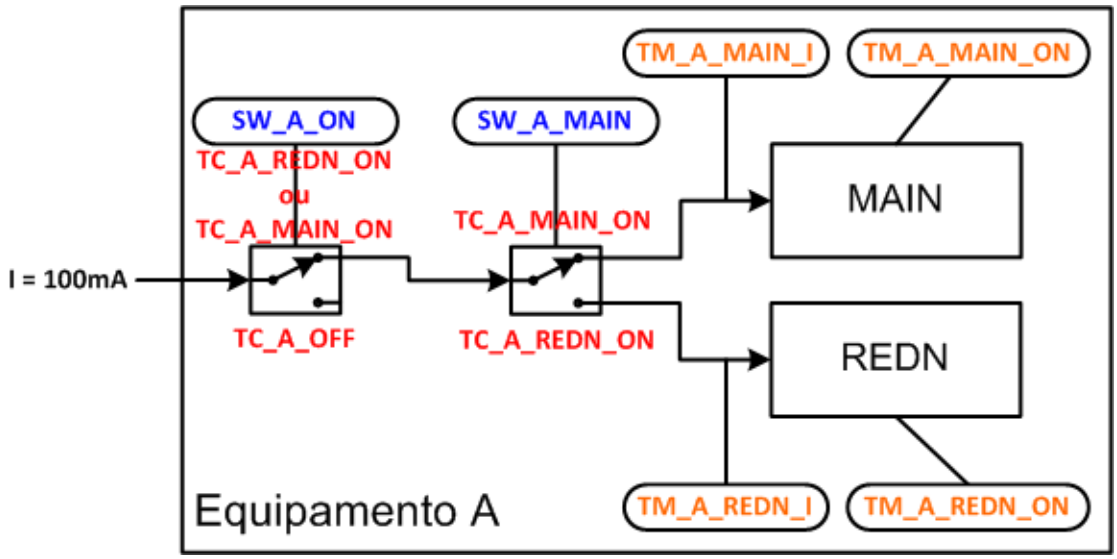
4.1 Modelo comportamental baseado em regras

Esta Seção tem por objetivo explicar como a base de dados Modelo Comportamental de um simulador pode ser representada por meio de regras compostas por expressões de precondições e efeitos. A representação física do equipamento pode ser obtida através de parâmetros bem definidos para as partes, como chaves, pontos de observação de medidas (telemetrias) e controles (telecomandos). No modelo comportamental do equipamento (A) é

composto por uma unidade principal (MAIN) e uma redundante (REDN), alimentada por uma fonte de corrente (I) que supre nominalmente 100 miliampères. Sua operação é realizada por meio de telecomandos (TC_A_MAIN_ON, TC_A_REDN_ON e TC_A_OFF) que atuam sobre duas chaves de controle (SW_A_ON e SW_A_MAIN). Seus parâmetros de monitoração incluem quatro telemetrias, sendo duas lógicas para o estado de funcionamento de cada unidade (TM_A_MAIN_ON e TM_A_REDN_ON) e duas de corrente de alimentação (TM_A_MAIN_I e TM_A_REDN_I).

Um diagrama de blocos do equipamento (A) ilustrando seus módulos principal (MAIN) e redundante (REDN), suas chaves, telemetrias e telecomandos é mostrado na Figura 4.1. Telemetrias são indicadas em laranja e chaves em azul. Telecomandos são mostrados em vermelho e a posição relativa de seu rótulo à chave associada indica a posição comandada da chave. Um telecomando pode simultaneamente mudar a posição de mais de uma chave. Mais de um telecomando pode mudar a posição de uma chave para a mesma posição. A notação usada para representar os elementos do equipamento (A) é semelhante à solução adotada para as especificações de modelos comportamentais para o simulador operacional SIMCBERS. (CBERS, 2012)

Figura 4.1. Diagrama de blocos do equipamento A.



Fonte: Produção do autor

4.1.1 Descrição das regras de comportamento

O modelo comportamental do sistema representado pelo diagrama de blocos da Figura 4.1 é composto por seis regras, identificadas de R1 até R6 (Tabela 4.1).

A Tabela 4.1 mostra as regras de um modelo comportamental simples que descrevem a operação do equipamento (A), em termos de telemetrias e parâmetros internos representando posição de chaves, em resposta à recepção e à execução de telecomandos. A primeira coluna da tabela indica o nome da regra (R1, R2, R3, etc.), a segunda coluna mostra a expressão com a pré-condição e a terceira, os efeitos decorrentes da ativação da pré-condição.

Tabela 4.1. Regras de um modelo comportamental simples.

Regra	Precondição	Efeitos
R1	TC_A_MAIN_ON == 1	SW_A_ON = 1
		SW_A_MAIN = 1
R2	SW_A_ON == 1 AND SW_A_MAIN == 1	TM_A_MAIN_ON = 1
		TM_A_REDN_ON = 0
		TM_A_MAIN_I = 100
		TM_A_REDN_I = 0
R3	TC_A_REDN_ON == 1	SW_A_ON = 1
		SW_A_MAIN = 0
R4	SW_A_ON == 1 AND SW_A_MAIN == 0	TM_A_MAIN_ON = 0
		TM_A_REDN_ON = 1
		TM_A_MAIN_I = 0
		TM_A_REDN_I = 100
R5	TC_A_OFF == 1	SW_A_ON = 0
		SW_A_MAIN = 1
R6	SW_A_ON == 0	TM_A_MAIN_I = 0
		TM_A_REDN_I = 0

Fonte: Produção do autor.

4.1.2 Regras de ligamento da unidade principal

Na Tabela 4.1, o telecomando (TC_A_MAIN_ON) é responsável por ligar a unidade principal (MAIN) do equipamento (A). Um relê controla o chaveamento entre os estados de liga (SW_A_ON == 1) e desliga (SW_A_ON == 0) do

equipamento. Uma chave lógica controla o chaveamento de suas unidades principal (SW_A_MAIN == 1) ou redundante (SW_A_MAIN == 0). Um par de telemetrias de monitoração permite verificar se a unidade principal (TM_A_MAIN_ON) ou redundante (TM_A_REDN_ON) do equipamento está ligada (== 1) ou desligada (== 0). Uma telemetria de monitoração permite verificar o valor das correntes consumidas pelas unidades principal (TM_A_MAIN_I) e redundante (TM_A_REDN_I). Nominalmente, uma unidade ligada consome 100 miliampères quando ligado (TM_A_MAIN_I == 100 ou TM_A_REDN_I == 100) ou 0 miliampères quando desligado (TM_A_MAIN_I == 100 ou TM_A_REDN_I == 0).

A Tabela 4.2 explicita o par de regras (R1 e R2 na Tabela 4.1) responsável por caracterizar a operação de ligamento da unidade principal (MAIN) do equipamento (A).

Tabela 4.2. Regras de ligamento da unidade principal.

Regra	Precondição	Efeitos
R1	TC_A_MAIN_ON == 1	SW_A_ON = 1
		SW_A_MAIN = 1
R2	SW_A_ON == 1 AND SW_A_MAIN == 1	TM_A_MAIN_ON = 1
		TM_A_REDN_ON = 0
		TM_A_MAIN_I = 100
		TM_A_REDN_I = 0

Fonte: Produção do autor.

Na primeira regra (R1), a precondição consiste na recepção do telecomando (TC_A_MAIN_ON == 1) para ligar a unidade principal do equipamento. Sua consequência consiste na aplicação do efeito de atribuir às chaves seletoras de liga-desliga (SW_A_ON) e de principal-redundante (SW_A_MAIN) os estados de ligado (SW_A_ON = 1) e principal (SW_A_MAIN = 1), respectivamente.

A segunda regra (R2) tem como precondição necessária e suficiente a execução bem-sucedida da primeira regra. Sua condição consiste na chave de ligamento o equipamento estar na posição ligada (SW_A_ON == 1) ao mesmo tempo em que a chave de seleção de unidade do equipamento (SW_A_MAIN) estiver na posição principal (SW_A_MAIN == 1). A regra (R2) lista quatro

consequências para esta condição. Primeiramente, a telemetria que indica o status de ligado ou desligado da unidade principal do equipamento (TM_A_MAIN_ON) assume o valor verdadeiro (= 1), indicando unidade principal ligada, e a telemetria correspondente da unidade redundante (TM_A_REDN_ON) assume o valor falso (= 0), indicando unidade redundante desligada. Em seguida, valores de corrente consumida são atribuídas às telemetrias de corrente das unidades principal (TM_A_MAIN_I = 100) e redundante (TM_A_REDN_I = 0).

4.1.3 Regras de ligamento da unidade principal após falha

Uma falha típica em satélites consiste em perda de um canal de dados de telemetria que passa a assumir um valor fixo. Em um cenário deste tipo, a condição de mudança de estado de uma chave monitorada não tem o efeito de alterar o valor da telemetria. Considerando a regra da Tabela 4.2, este tipo de falha pode ser representado por meio de alteração de uma das regras, no caso, a R2, conforme Tabela 4.3.

Tabela 4.3. Regras de ligamento da unidade principal com telemetria em falha.

Regra	Precondição	Efeitos
R1	TC_A_MAIN_ON == 1	SW_A_ON = 1
		SW_A_MAIN = 1
R2X	SW_A_ON == 1 AND SW_A_MAIN == 1	TM_A_MAIN_ON = 0
		TM_A_REDN_ON = 0
		TM_A_MAIN_I = 100
		TM_A_REDN_I = 0

Fonte: Produção do autor.

Na regra alterada (R2X), a telemetria de estado de liga-desliga da unidade principal do equipamento (TM_A_MAIN_ON) encontra-se fixa na posição desligado (== 0). O telecomando (TC_A_MAIN_ON) liga a unidade principal do equipamento e altera os estados das chaves, porém este efeito não é refletido na telemetria.

Uma observação com relação a este cenário é a possibilidade de se diagnosticar a falha a partir da inconsistência entre telemetrias de estado (TM_A_MAIN_ON == 0) e corrente (TM_A_MAIN_I == 100).

A aplicação de uma mudança de regra deste tipo pode ser alcançado a partir de uma mutação na expressão de efeito.

Outro exemplo de falha típica consiste na **chave de relê presa em posição fixa**. Neste cenário, a condição de recepção de um telecomando que controla da posição da chave não tem o efeito de alterar seu estado. Com base no exemplo da Tabela 4.2, um modo de falha deste tipo pode ser representado conforme a Tabela 4.4.

Tabela 4.4. Regras de ligamento da unidade principal com chave presa.

Regra	Precondição	Efeitos
R1Y	TC_A_MAIN_ON == 1	SW_A_ON = 1
		SW_A_MAIN = 0
R2	SW_A_ON == 1 AND SW_A_MAIN == 1	TM_A_MAIN_ON = 1
		TM_A_REDN_ON = 0
		TM_A_MAIN_I = 100
		TM_A_REDN_I = 0

Fonte: Produção do autor.

Na regra alterada (R1Y), a precondição de recepção do telecomando (TC_A_MAIN_ON) para ligar a unidade principal do equipamento tem como efeito comutar a chave de liga-desliga para a posição ligada (SW_A_ON = 1), mas a chave de principal-redundante (SW_A_MAIN) não é comutada para principal (SW_A_MAIN == 0).

Novamente, a aplicação de uma mudança de regra deste tipo pode ser alcançada a partir de uma mutação na segunda expressão de efeito da regra.

Uma observação quanto à modificação de modelos de simuladores é que um mesmo comportamento pode ser expresso de forma diferente, conforme o nível de conhecimento, o foco no detalhamento, o compromisso entre fidelidade e desempenho, ou a abordagem empregada.

4.1.4 Regras alternativas de ligamento do equipamento

Considere-se agora um subconjunto maior das regras de comportamento (R1 a R4 na Tabela 4.1), responsáveis pelo ligamento das unidades principal (MAIN) e redundante (REDN), conforme mostrado a seguir (Tabela 4.5).

Tabela 4.5. Regras de ligamento do equipamento.

Regra	Precondição	Efeitos
R1	TC_A_MAIN_ON == 1	SW_A_ON = 1
		SW_A_MAIN = 1
R2	SW_A_ON == 1 AND SW_A_MAIN == 1	TM_A_MAIN_ON = 1
		TM_A_REDN_ON = 0
		TM_A_MAIN_I = 100
		TM_A_REDN_I = 0
R3	TC_A_REDN_ON == 1	SW_A_ON = 1
		SW_A_MAIN = 0
R4	SW_A_ON == 1 AND SW_A_MAIN == 0	TM_A_MAIN_ON = 0
		TM_A_REDN_ON = 1
		TM_A_MAIN_I = 0
		TM_A_REDN_I = 100

Fonte: Produção do autor.

A terceira regra (R3) ilustra um caso alternativo de telecomando, em que a precondição consiste na recepção do telecomando (TC_A_REDN_ON) para ligar a unidade redundante do equipamento. Sua consequência consiste na aplicação do efeito de atribuir às chaves seletoras de liga-desliga (SW_A_ON) e de principal-redundante (SW_A_MAIN) os estados de ligado (SW_A_ON = 1) e redundante (SW_A_MAIN = 0), respectivamente.

A quarta regra (R4) tem como precondição a correta execução da terceira regra, que consiste na chave de ligamento o equipamento estar na posição ligada (SW_A_ON == 1) ao mesmo tempo em que a chave de seleção de unidade do equipamento (SW_A_MAIN) estiver na posição redundante (SW_A_MAIN == 0). Esta regra lista quatro consequências. A telemetria que indica o status de ligado ou desligado da unidade principal do equipamento (TM_A_MAIN_ON) assume o valor falso (= 0), indicando unidade principal desligada, e a telemetria correspondente da unidade redundante (TM_A_REDN_ON) assume o valor verdadeiro (= 1), indicando unidade redundante ligada. Valores de corrente consumida são atribuídas às telemetrias de corrente das unidades principal (TM_A_MAIN_I = 0) e redundante (TM_A_REDN_I = 100).

Uma maneira mais enxuta de representar o comportamento descrito por estas quatro regras (Tabela 4.5) consiste no exemplo da Tabela 4.6.

Tabela 4.6. Regras alternativas de ligamento do equipamento.

Regra	Precondição	Efeitos
R1B	TC_A_MAIN_ON == 1	TM_A_MAIN_ON = 1
		TM_A_REDN_ON = 0
		TM_A_MAIN_I = 100
		TM_A_REDN_I = 0
R2B	TC_A_REDN_ON == 1	TM_A_MAIN_ON = 0
		TM_A_REDN_ON = 1
		TM_A_MAIN_I = 0
		TM_A_REDN_I = 100

Fonte: Produção do autor.

Neste exemplo, a omissão dos parâmetros intermediários correspondentes a chaves leva a uma simplificação do modelo comportamental, com valores de telemetrias refletindo o efeito de telecomandos de forma equivalente. Entretanto, isto é obtido ao custo de perda de informação relevante para a modelagem de falhas.

Por exemplo, a implementação de uma falha de relê presa em posição fixa (R1Y) requer a alteração de regras conforme mostrado na Tabela 4.7.

Tabela 4.7. Regras alternativas de ligamento do equipamento com falha.

Regra	Precondição	Efeitos
R1BY	TC_A_MAIN_ON == 1	TM_A_MAIN_ON = 0
		TM_A_REDN_ON = 1
		TM_A_MAIN_I = 0
		TM_A_REDN_I = 100
R2B	TC_A_REDN_ON == 1	TM_A_MAIN_ON = 0
		TM_A_REDN_ON = 1
		TM_A_MAIN_I = 0
		TM_A_REDN_I = 100

Fonte: Produção do autor.

Ao passo que a modelagem anterior (R1Y + R2 na Tabela 4.4) requeria alteração de um único parâmetro para incorporar a falha, o modelo simplificado (R1BY, Tabela 4.7) requer o ajuste de quatro parâmetros.

Entretanto, esta mudança de regra pode ser alcançada a partir de uma única operação de cruzamento entre as regras R1B e R2B.

4.2 Obtenção de regras candidatas por programação genética

A característica autoadaptativa do modelo comportamental requer o monitoramento dos valores das telemetrias recebidas do satélite e sua comparação com os valores das telemetrias geradas pelo modelo no simulador, para identificação da necessidade de mudança ou não do modelo. E, quando necessário, a alteração das regras do modelo.

A aplicação de algoritmos evolutivos permite a modificação de expressões de precondição ou efeitos. Por exemplo, na Tabela 4.8 a seguir a regra RB1 indica o funcionamento normal do equipamento (A) principal ligado com corrente 100, enquanto que, uma regra alternativa, R1BY, indica a ocorrência de uma falha que leva o chaveamento automático para o módulo redundante. A regra alternativa poderia ser obtida automaticamente a partir da regra original R1B, através da aplicação nas expressões de **efeitos** de quatro mutações, duas permutações (crossover) ou uma combinação de ambas.

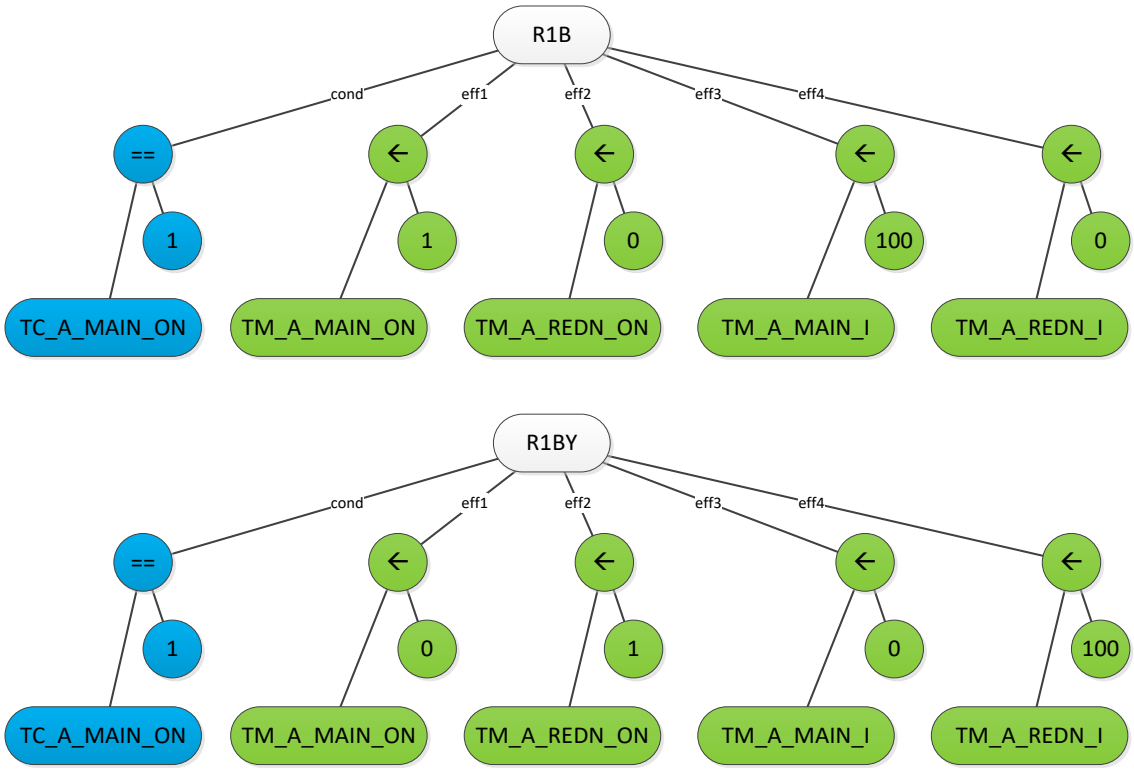
Tabela 4.8. Comparação de regras original e após falha.

Regra	Precondição	Efeitos
R1B	TC_A_MAIN_ON == 1	TM_A_MAIN_ON = 1
		TM_A_REDN_ON = 0
		TM_A_MAIN_I = 100
		TM_A_REDN_I = 0
R1BY	TC_A_MAIN_ON == 1	TM_A_MAIN_ON = 0
		TM_A_REDN_ON = 1
		TM_A_MAIN_I = 0
		TM_A_REDN_I = 100

Fonte: Produção do autor.

A Figura 4.2 permite uma visualização das regras original (R1B) e alternativa (R1BY), a qual passa a vigorar após a ocorrência da falha no módulo principal, apresentadas na forma de árvore sintética.

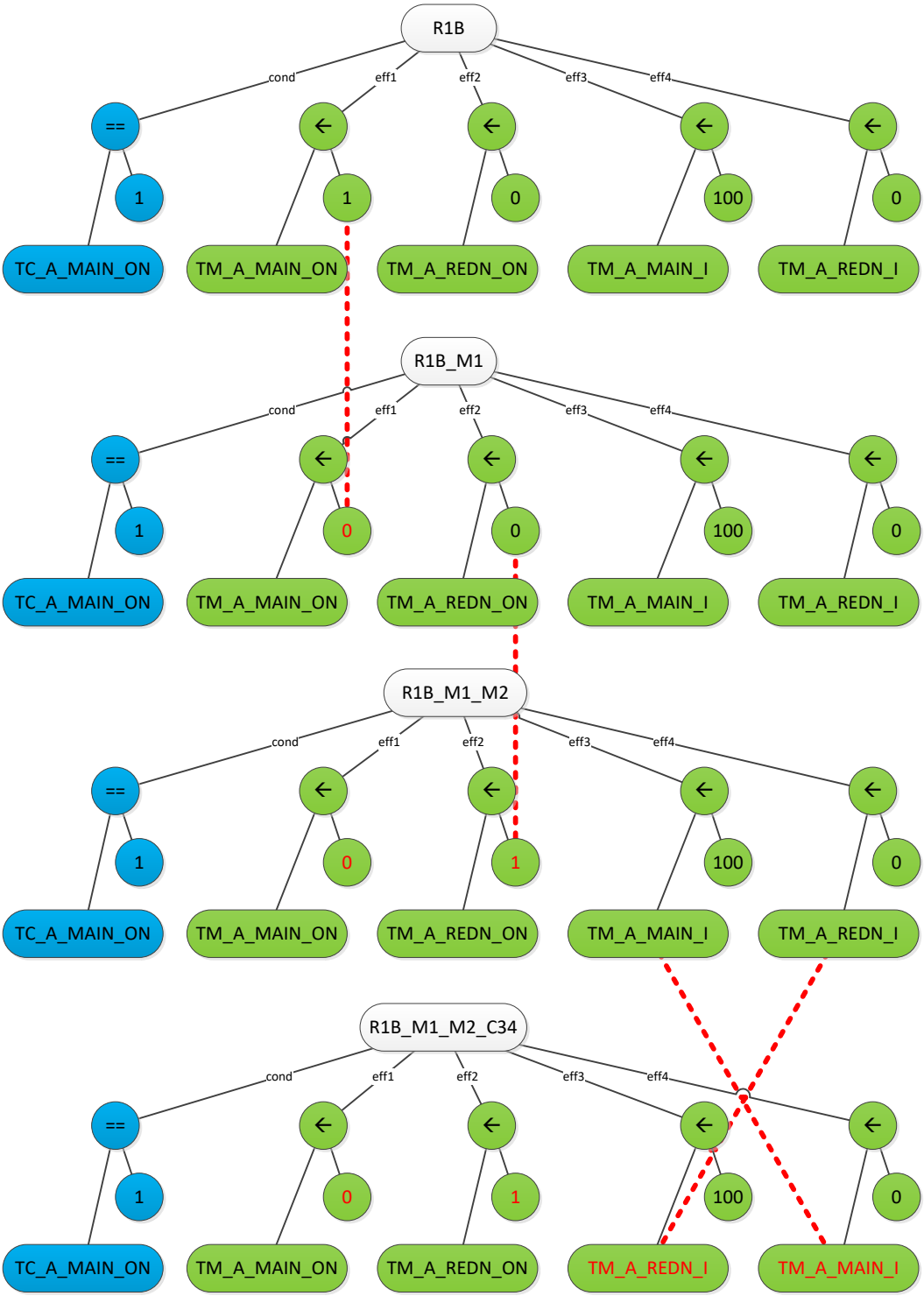
Figura 4.2. Árvores sintáticas de regras original e após falha.



Fonte: Produção do autor.

Uma possibilidade de obtenção de uma regra idêntica à regra após falha (R1BY) a partir da regra original (R1B) através da aplicação do algoritmo de programação genética após quatro gerações é ilustrada na Figura 4.3.

Figura 4.3. Árvores sintáticas da obtenção de regra nova a partir da regra original.



Fonte: Produção do autor

Na Figura 4.3, a regra filha R1B_M1 é obtida por mutação da regra pai R1B. Na próxima geração a regra filha R1B_M1_M2 é obtida por mutação da regra pai R1B_M1. Na geração seguinte a regra filha R1B_M1_M2_C12 é obtida por

crossover da regra pai R1B_M1_M2, para obtenção de uma árvore sintática idêntica à regra R1BY esperada.

4.2.1 Representação das regras de comportamento

Para efeito de implementação da heurística do Reconfigurador de Modelo, uma regra é considerada uma estrutura de dados composta por expressões. Cada regra contém uma expressão de condição e uma ou mais expressões de efeito. Nas Figuras 4.2 e 4.2 as regras (“R”#) indicadas em cinza contêm expressões de condição (“cond”) em verde e listas de efeitos (“eff”#) em azul.

- Regra = {Condição, Efeitos (1..n)}
    - 1 x Condição
    - n x Efeitos

Uma expressão é uma lista de alternada de operandos e operações.

- Expressão = {Operando[, Operador[, Operando, ...]]}
    - Expressão
        - m x Operandos
        - (m-1) x Operadores

No Simulador Simplificado , são consideradas válidas as seguintes operações nativas:

- "NEQ" (“*NOT EQUAL*”): operação lógica binária “≠”
- "GT" (“*GREATER THAN*”): operação lógica binária “>”
- "GE" (“*GREATER THAN OR EQUAL TO*”): operação lógica binária “≥”
- "EQ" (“*EQUAL TO*”): operação lógica binária “=”
- "LE" (“*LESSER THAN OR EQUAL TO*”): operação lógica binária “≤”
- "LT" (“*LESSER THAN*”): operação lógica binária “<”
- "AND": operação lógica binária “E”
- "OR": operação lógica binária “OU”
- "NOT": operação lógica unária “NÃO”
- "+": operação aritmética binária “SOMA”

- "-": operação aritmética binária “SUBTRAÇÃO”
- "*": operação aritmética binária “MULTIPLICAÇÃO”
- "/": operação aritmética binária “DIVISÃO”
- "^": operação aritmética binária “EXPONENCIAÇÃO”
- "%": operação aritmética binária “RESTO”
- "ABS": função unária “MÓDULO”
- "MIN": função binária “MÍNIMO”
- "MAX": função binária “MÁXIMO”
- "LOG": função unária “LOGARITMO NEPERIANO”
- "EXP": função unária “EXPONENCIAL”
- "SIN": função unária “SENO”
- "COS": função unária “COSENO”
- "TAN": função unária “TANGENTE”
- "ASI": função unária “ARCO SENO”
- "ACO": função unária “ARCO COSENO”
- "ATA": função unária “ARCO TANGENTE”

Um operando pode ser um valor constante real, um parâmetro variável que pode assumir um valor real, ou uma expressão.

- Operando
    - Ex: “1” (valor constante real)
    - Ex: “X” (parâmetro variável real)
    - Ex: {“X”, “+”, “1”} (expressão)

Numa expressão, uma operação binária é cercada pelos dois operandos.

- Operação binária
    - Ex: {“X”, “**AND**”, “Y”} = X e Y

    - Ex: {{“A”, “**+**”, “B”}, “**MAX**”, “2”} = máximo ((A + B), 2)

Numa expressão, uma operação unária é cercada pelo mesmo operando.

- Operação unária
    - Ex: {“X”, “**SIN**”, “X”} = seno (X)
    - Ex: {“A”, “OR”, “B”}, “**NOT**” {“A”, “OR”, “B”} = não (A ou B)

Uma função com mais de duas variáveis pode ser implementada como uma função binária especial utilizando como operando uma lista de parâmetros. Neste caso, o primeiro operando seria uma chave tipo “*string*” reservada para indicar esta situação e o segundo operando uma lista de operandos que seriam as suas entradas.

Uma expressão de condição é uma expressão lógica, cujo resultado de sua avaliação deve ser igual a zero ou um segundo a lógica booleana. Entretanto, o Simulador Simplificado utiliza lógica nebulosa para interpretar valores diferentes de zero ou um (TOMINAGA, 2010).

Um efeito é uma lista iniciada por um parâmetro alvo, seguido de uma expressão.

- Efeito: = {Parâmetro, Expressão}
    - Parâmetro
    - Expressão
        - m x Operandos
        - (m-1) x Operadores

O valor numérico resultante da avaliação da expressão é atribuído ao parâmetro alvo se a expressão de condição retornar o valor um. Se expressão de condição retornar o valor zero, o valor do parâmetro alvo é mantido. Condições cujos valores sejam diferentes de zero ou de um podem levar um ajuste no valor atribuído ao parâmetro alvo conforme a proximidade do resultado de zero ou um (TOMINAGA, 2010).

4.2.2 Heurística para adaptação de regras

A heurística para alteração de uma regra por meio de programação genética obedece à sequência de passos a seguir.

1) O conjunto de regras original é replicado para gerar um conjunto de regras alternativas.
2) Conjuntos de regras alternativas são gerados por meio de mutação de uma das regras do conjunto, que pode ocorrer em um operando ou um operador de uma expressão, ou no parâmetro alvo de um efeito.
    a) Na mutação de operação, uma operação é substituída por outra operação escolhida aleatoriamente de uma lista de operações válidas para o simulador.
    b) Na mutação de operando, um operando é substituído por outro operando, que é escolhido aleatoriamente entre um parâmetro que faz parte da lista de parâmetros definidos para o simulador, um valor numérico real, ou uma expressão que é construída através de escolha aleatória de uma operação e seus operandos associados de forma válida.
    c) Na mutação de parâmetro alvo, ele é substituído por outro parâmetro que faz parte da lista de parâmetros definidos para o simulador, escolhido aleatoriamente.
3) Mais conjuntos de regras alternativas são gerados por meio de *crossover* entre duas regras, que pode ocorrer ou entre dois operandos, ou entre dois operadores do mesmo tipo de expressões, ou entre parâmetros alvos.
4) Os conjuntos de regras alternativas são ordenados conforme critério de aptidão.
    a) O Simulador Simplificado é executado para cada conjunto de regras.
    b) Uma função peso é calculada para cada execução com base nas discrepâncias entre telemetrias de referência e valores calculados para parâmetros correspondentes pelo simulador e seus limites de tolerância.

c) Conjuntos de regras com menor discrepância são considerados mais aptos que aqueles cujas discrepâncias são maiores.

5) Os conjuntos de regras alternativas considerados menos aptos são eliminados para compor uma nova geração.

6) Repetem-se os passos anteriores de replicação (1), mutação (2), crossover (3), teste de aptidão (4) e eliminação (5) até a obtenção de pelo menos uma solução válida.

A aplicação da programação genética demanda esforço computacional intenso para um número elevado de regras ou regras complexas com muitos operandos e operações.

O conjunto de regras original a ser utilizado para a geração de regras alternativas é escolhido a partir de **análise de correlação** do modelo comportamental para redução de escopo e diminuição do espaço de busca, conforme descrito na Seção 4.3.

4.3 Abordagem por análise de correlação

Nesta Seção é apresentada a abordagem por análise de correlação que visa reduzir o espaço de busca gerado pela programação genética para obtenção de regras candidatas.

A Seção 4.1 apresentou um conjunto de regras que compõem um modelo comportamental simples de um equipamento hipotético de satélite. Para o Simulador de Satélites, além destas regras caracterizando os modelos deve ser associada uma Configuração Inicial, a qual contém pelo menos um **Estado Inicial** e uma **Fila de Eventos**.

O **Estado Inicial** (ver Tabela 3.1) é constituído por um conjunto de valores atribuídos a cada parâmetro de simulação no instante inicial de uma sessão de simulação. No primeiro passo de uma sessão de simulação, os parâmetros de simulação são inicializados com valores do estado inicial.

A **Fila de Eventos** (ver Tabela 3.1) contém uma lista temporizada de parâmetros de evento que são disparados em seus tempos de execução. Estes

representam tempos de execução de telecomandos enviados pelo sistema de solo de TT&C.

A cada passo de simulação, as regras são aplicadas avaliando-se as expressões de precondição e de efeito. Caso o resultado da avaliação da expressão de precondição retorne um valor verdadeiro, o parâmetro alvo, do lado esquerdo da expressão de efeito, tem seu valor substituído pelo resultado da avaliação da expressão de valor, do lado direito da expressão de efeito. Este processo é executado para cada regra, utilizando-se os valores dos parâmetros atualizados para o cálculo das expressões.

Novos valores são atribuídos a um parâmetro de simulação se e quando uma ou mais regras de atribuição a este parâmetro são executadas. Em outras palavras, um parâmetro precisa ser o alvo de uma expressão de atribuição de uma regra cuja precondição retorne verdadeiro. Caso isto não aconteça, o valor do parâmetro deve se manter igual ao passo anterior.

Esta característica pode ser utilizada para rastrear um conjunto de parâmetros correlacionados e outro de regras correlacionadas a um dado parâmetro. Por exemplo, na Tabela 4.1, o parâmetro TM_A_MAIN_ON tem valor atribuído pelas regras R2 e R4, não sendo utilizado na precondição ou na expressão de valor de efeito de nenhuma regra. As regras R2 e R4, além do parâmetro TM_A_MAIN_ON, utilizam em suas expressões de precondição os parâmetros SW_A_ON e SW_A_MAIN e em suas expressões de efeito os parâmetros TM_A_REDN_ON, TM_A_MAIN_I e TM_A_REDN_I. Nestas condições, se o valor do parâmetro simulado TM_A_MAIN_ON estiver discrepante com o valor da telemetria correspondente recebida do satélite, o motivo pode estar relacionado à regra R2 ou à regra R4. Se outros parâmetros não estiverem discrepantes, no caso TM_A_REDN_ON, TM_A_MAIN_I e TM_A_REDN_I, então o problema estaria relacionado à atribuição de efeito do parâmetro TM_A_MAIN_ON. Caso contrário, a suspeita recairia sobre os parâmetros de precondição SW_A_ON e SW_A_MAIN.

A análise de correlação pode ser caracterizada pelos seguintes passos:

1) Identificação dos parâmetros afetados e não afetados

2) Identificação das regras afetadas, que contêm parâmetros afetados em suas expressões
3) Identificação dos parâmetros candidatos, presentes nas expressões das regras afetadas
4) Identificação de regras candidatas, que contêm parâmetros candidatos em suas expressões
5) Repetição em loop dos passos 3 e 4 até não se encontrarem novas regras candidatas com parâmetros candidatos

Desta forma, a aplicação sucessiva da análise de correlação entre regras e parâmetros permite o fechamento do cerco para a busca de um conjunto mínimo de parâmetros candidatos e regras candidatas a serem alterados. Após isto, a determinação de novos valores para parâmetros candidatos pode ser aplicada diretamente para parâmetros candidatos com telemetrias associadas, ou indiretamente por inferência ou busca no caso de parâmetros candidatos sem telemetrias associadas. Caso não seja possível obter soluções válidas por meio de alterações de valores de parâmetros candidatos, são buscadas expressões alternativas para regras correlacionadas via programação genética.

4.4 Considerações

Neste Capítulo a modelagem de um sistema baseado em regras e como as regras são adaptadas para representar as variações comportamentais de um modelo autoadaptativo são apresentadas através de um exemplo de um equipamento de satélite de um simulador. Regras são compostas por expressões de precondições e efeitos, que podem representar um mesmo comportamento por meio de formas alternativas.

Foram ilustradas: (i) as abordagens para a aplicação de alterações no modelo comportamental, (ii) a heurística para aplicação de algoritmos evolutivos gerar regras alternativas para adaptação autônoma de modelos comportamentais, (iii) o algoritmo de análise de correlação que permite a identificação de parâmetros afetados por meio de discrepâncias com telemetrias de referência identificando parâmetros candidatos e regras candidatas para a aplicação de

alterações minimizando o conjunto de parâmetros e regras a serem alterados ao modelo comportamental.

5 PROVA DE CONCEITO

Este Capítulo mostra a prova de conceito da arquitetura e do processo propostos nesta Tese, considerando os limites da implementação do protótipo do Reconfigurador de Modelo, desenvolvido para demonstrar a efetividade da adaptação autônoma de modelos comportamentais baseados em regras. Para isso foram definidos cenários de comportamentos normais e de falhas característicos e que provocariam alterações em modelos comportamentais de simuladores de satélites.

Os cenários são classificados em simples e complexos.

Os cenários complexos (Seção 5.1) constituem prova de conceito das atividades do processo de adaptação do modelo comportamental por análise de correlação. A simulação de um subsistema simplificado de suprimento de energia de um satélite fictício é a base para a ilustração dos cenários complexos. Os cenários cobrem condições de degradação e falha em equipamento, sendo o cenário (1) degradação por envelhecimento da bateria, e o cenário (2) falha em circuito aberto no subsistema de suprimento de energia. Nestes cenários, parâmetros candidatos em regras associadas são identificados e testadas para adaptação do modelo por alteração de valores de parâmetros sem a necessidade de geração de novas regras.

Os cenários simples (Seção 5.2) foram elaborados para validar a autoadaptação do modelo com uso de programação genética. Cinco cenários cobrem as seguintes situações:

(1) ligamento ou desligamento automático de um equipamento em bordo, implicará em substituição de apenas um parâmetro na regra;

(2) comutação automática de equipamento principal para redundante disparada por um mecanismo de proteção, implicará na troca de dois parâmetros em um conjunto de regras;

(3) falha em um equipamento, implicará em modificação do valor de um único parâmetro;

(4) curto em um destes canais do painel solar que alimentam o satélite, neste caso um termo da soma das correntes parciais de cada canal é zerada implicando em eliminação de uma expressão em uma regra;

(5) perda da capacidade de carga e descarga da bateria levando a dissipação de parte da energia que deveria ser armazenada, implicará em inclusão de uma expressão a regra.

5.1 Processo de adaptação de modelos comportamentais - cenários complexos

Para descrever o processo de identificação e alteração de parâmetros discrepantes em um modelo de simulação baseado em regras analiticamente são apresentados possíveis cenários baseados em um modelo simplificado baseado em regras de um satélite fictício. Eles têm como objetivo explicar, por meio de exemplos, a abordagem de identificação de parâmetros e regras candidatas para atualização automática dos modelos comportamentais de simulação.

5.1.1 Descrição do funcionamento nominal do suprimento de energia de um satélite fictício

Este cenário descreve o funcionamento nominal do suprimento de energia de um satélite fictício no início da sua vida útil, por meio de 13 regras comportamentais e 20 parâmetros de simulação.

O modelo comportamental é caracterizado por meio do conjunto de regras ilustrado na Tabela 5.1.

Tabela 5.1. Regras componentes do cenário nominal.

Regra	Precondição	Efeitos
RNEXT	IF: 1	THEN: SIMTIME = SIMTIME + SIMSTEP
RSTOP	IF: SIMTIME GE SIMEND	THEN: SIMSTOP = 1
R_SUN	IF: EV_SUN	THEN: SUN = 1
R_ECL	IF: EV_ECL	THEN: SUN = 0
R_ISAG_ECL	IF: SUN EQ 0	THEN: ISAG = 0
R_ISAG_SUN	IF: (SUN EQ 1) AND (VBAT LT VBATMAX)	THEN: ISAG = ISAGMAX
R_ISAG_FUL	IF: (SUN EQ 1) AND (VBAT GE VBATMAX)	THEN: ISAG = MIN (ILOAD, ISAGMAX)
R_IBAT	IF: 1	THEN: IBAT = ISAG - ILOAD
R_QBAT	IF: QBAT LE QBATMAX	THEN: QBAT = QBAT + SIMSTEP * IBAT
R_QBAT_MAX	IF: QBAT GT QBATMAX	THEN: QBAT = QBATMAX
R_VBAT	IF: 1	THEN: VBAT = QBAT / CBAT
R_PL_ON	IF: EV_PL_ON	THEN: ILOAD = I_ON
R_PL_OFF	IF: EV_PL_OFF	THEN: ILOAD = I_OFF

Fonte: Produção do autor.

A primeira coluna corresponde ao identificador da regra, a segunda coluna à expressão de precondição e a terceira coluna às expressões de efeito.

A expressão de precondição, à direita do rótulo de precondição (IF:), avalia se o efeito da regra se aplica ou não naquele passo de simulação. Caso o valor resultante da avaliação da expressão de precondição seja VERDADEIRO (1), a expressão de efeito será avaliada.

A expressão de efeito, à direita do rótulo de efeito (THEN:), atribui a um parâmetro alvo de simulação, à esquerda do sinal de atribuição (=), o valor resultante da avaliação da expressão à direita do sinal de atribuição (=).

As primeiras duas regras descrevem o controle de execução do simulador.

- RNEXT determina a taxa de incremento do tempo de simulação (SIMTIME) a cada passo de simulação (SIMSTEP).

- RSTOP verifica a condição de parada da simulação (SIMSTOP), que ocorre quando o tempo de simulação (SIMTIME) atingir um valor maior ou igual ao tempo de parada (SIMEND).

As duas regras seguintes definem o comportamento do estado de iluminação do satélite (SUN) conforme o disparo de eventos de entrada na fase orbital iluminada (EV_SUN) ou em eclipse (EV_ECL).

- R_SUN atribui ao estado de iluminação do satélite (SUN) o valor VERDADEIRO (1) quando houver a entrada do satélite na fase orbital iluminada (EV_SUN).

- R_ECL atribui ao estado de iluminação do satélite (SUN) o valor FALSO (0) quando houver a entrada do satélite na fase orbital em eclipse (EV_ECL).

As três regras seguintes definem o comportamento da corrente do gerador de painel solar (I_SAG) conforme o estado de iluminação do satélite (SUN) e o controle de carga da bateria, executada por meio da medição da tensão de bateria (VBAT).

- R_ISAG_ECL atribui à corrente do gerador de painel solar (I_SAG) o valor zero se o painel solar não estiver iluminado (SUN = 0).

- R_ISAG_SUN atribui à corrente do gerador de painel solar (I_SAG) o valor máximo permitido (ISAGMAX) se o painel solar estiver iluminado (SUN = 1) e a tensão de bateria indicar bateria não totalmente carregada (VBAT < VBATMAX).

- R_ISAG_FUL atribui à corrente do gerador de painel solar (I_SAG) o menor valor entre a corrente consumida pela carga (ILOAD) e a corrente máxima do gerador (ISAGMAX), se o painel solar estiver iluminado (SUN = 1) e a tensão de bateria indicar bateria totalmente carregada (VBAT = VBATMAX).

A regra seguinte define o comportamento da corrente de carga e descarga da bateria (IBAT) conforme os valores de corrente do gerador de painel solar (I_SAG) e de corrente de carga (ILOAD).

R_IBAT atribui à corrente da bateria (IBAT) a diferença entre a corrente gerada pelo painel solar (I_SAG) e a corrente de carga consumida pelo satélite (ILOAD). Uma corrente de bateria positiva (IBAT > 0) indica carga da bateria, ao passo que uma corrente de bateria negativa (IBAT < 0) indica descarga de bateria.

- As regras seguintes definem o comportamento da carga da bateria (QBAT).

- R_QBAT atualiza a carga da bateria (QBAT) de acordo com a corrente da bateria (IBAT).

- R_QBAT_MAX restringe a carga da bateria (QBAT) ao seu valor máximo permitido (QBAT_MAX).

A regra seguinte atualiza o valor da tensão de bateria (VBAT) considerando o valor da carga acumulada (QBAT) recalculada pela regra anterior.

- R_VBAT calcula o valor da tensão de bateria (VBAT) a partir do valor da carga da bateria (QBAT) e da capacitância da bateria (CBAT).

As duas últimas regras definem o valor da corrente de carga consumida pelo satélite (ILOAD)

- R_PL_ON atribui à corrente de carga (ILOAD), em caso de ligamento da carga útil (EV_PL_ON), o valor correspondente ao consumo da plataforma mais a da carga útil (I_ON).

- R_PL_OFF atribui à corrente de carga (ILOAD), em caso de desligamento da carga útil (EV_PL_OFF), o valor correspondente ao consumo da plataforma apenas (I_OFF).

A Tabela 5.2 mostra os valores iniciais dos parâmetros de simulação.

Tabela 5.2. Valores de parâmetros iniciais do cenário nominal.

Parâmetro	Tipo	Valor
SIMTIME	St	0
SIMEND	St	60000
SIMSTEP	St	60
SIMSTOP	Ev	0
SUN	St	1
ISAG	St	20
ILOAD	St	5
IBAT	St	15
VBAT	St	47
QBAT	St	61100
CBAT	St	1300
ISAGMAX	St	20
VBATMAX	St	50
QBATMAX	St	65000
I_ON	St	22
I_OFF	St	5
EV_SUN	Ev	0
EV_ECL	Ev	0
EV_PL_ON	Ev	0
EV_PL_OFF	Ev	0

Fonte: Produção do autor.

A primeira coluna mostra o identificador de parâmetro, a segunda coluna o seu tipo e a terceira coluna, seu valor numérico.

Parâmetros do tipo estado (“st”) são utilizados para armazenar valores numéricos reais que em sua maioria têm algum correspondente físico no modelo comportamental. A dinâmica do Simulador é caracterizada pela mudança de valores parâmetros deste tipo a cada passo, ao longo da sessão de simulação.

Parâmetros do tipo evento (“ev”) são utilizados para armazenar valores lógicos, VERDADEIRO (1) ou FALSO (0), que por default assumem o valor FALSO, mas atingem instantaneamente o valor VERDADEIRO em determinados instantes predefinidos ao longo da sessão de simulação, conforme uma fila de eventos. São utilizados para disparar mudanças comportamentais no simulador a partir de eventos previstos de origem externa ao modelo.

Os quatro primeiros parâmetros correspondem a parâmetros de controle da simulação.

- SIMTIME corresponde ao tempo corrente de simulação, medido em segundos.
- SIMEND corresponde ao tempo de parada de simulação, medido em segundos.
- SIMSTEP representa o intervalo de tempo, em segundos, decorrido a cada passo de simulação.
- SIMSTOP corresponde ao evento de parada de simulação.

Os 12 parâmetros seguintes correspondem a parâmetros de modelo do simulador.

- SUN - estado lógico de iluminação do satélite, controlado pelos eventos de entrada nas fases orbitais de iluminação (EV_SUN) e sombra (EV_ECL).
- ISAG - corrente gerada pelo painel solar, medido em ampères.
- ILOAD - corrente de carga consumida pelo satélite, medido em ampères.
- IBAT - corrente de bateria assumindo valores positivos durante a fase de carga e negativos durante a fase de descarga, medido em ampères.
- VBAT - tensão de bateria, medido em volts.
- QBAT - carga armazenada na bateria, medido em coulombs.
- CBAT - medida de capacitância armazenada na bateria, medido em farads.
- ISAGMAX - corrente máxima que pode ser gerada pelo painel solar, medido em ampères.

- VBATMAX - tensão máxima que a bateria pode atingir antes do corte de corrente pelo controlador de carga, medido em volts.

- QBATMAX - carga máxima que pode ser acumulada pela bateria, medido em coulombs.

- I_ON - corrente de carga correspondente ao consumo da plataforma mais a da carga útil, medido em ampères.

- I_OFF - corrente de carga correspondente ao consumo da plataforma apenas, medido em ampères.

- Os últimos quatro parâmetros correspondem a eventos temporizados na fila de eventos.

- EV_SUN - evento de entrada do satélite na fase orbital em iluminação, calculado externamente pela dinâmica de voo.

- EV_ECL - evento de entrada do satélite na fase orbital em eclipse, calculado externamente pela dinâmica de voo.

- EV_PL_ON - evento de ligamento temporizado da carga útil por telecomando.

- EV_PL_OFF - evento de desligamento temporizado da carga útil por telecomando.

A Tabela 5.3 mostra a **fila de eventos** para disparo temporizado durante a simulação. Na Tabela 5.3, a primeira coluna corresponde ao identificador do parâmetro de evento disparado, a segunda ao tipo de parâmetro (“ev”) e a terceira ao valor correspondente ao tempo de simulação (SIMTIME) de disparo do evento.

Tabela 5.3. Fila de eventos do cenário nominal.

Parâmetro	Tipo	Valor
EV_PL_ON	Ev	1800
EV_PL_OFF	Ev	2400
EV_ECL	Ev	4000
EV_SUN	Ev	6000
EV_PL_ON	Ev	7800
EV_PL_OFF	Ev	8400
EV_ECL	Ev	10000
EV_SUN	Ev	12000
EV_PL_ON	Ev	13800
EV_PL_OFF	Ev	14400
EV_ECL	Ev	16000
EV_SUN	Ev	18000
EV_ECL	Ev	22000
EV_SUN	Ev	24000
EV_PL_ON	Ev	25000
EV_PL_OFF	Ev	25600
EV_ECL	Ev	28000
EV_SUN	Ev	30000
EV_PL_ON	Ev	32000
EV_PL_OFF	Ev	32600
EV_ECL	Ev	34000
EV_SUN	Ev	36000
EV_PL_ON	Ev	38000
EV_PL_OFF	Ev	38600
EV_ECL	Ev	40000
EV_SUN	Ev	42000
EV_PL_ON	Ev	44000
EV_PL_OFF	Ev	44600
EV_ECL	Ev	46000
EV_SUN	Ev	48000
EV_PL_ON	Ev	50000
EV_PL_OFF	Ev	50600
EV_ECL	Ev	52000
EV_SUN	Ev	54000
EV_ECL	Ev	58000
EV_SUN	Ev	60000

Fonte: Produção do autor.

Os valores de EV_PL_ON e EV_PL_OFF correspondem a tempos de início e fim, respectivamente, programados para operação de carga útil.

Os valores de EV_ECL e EV_SUN correspondem aos horários de início e fim de eclipse. Num satélite de baixa órbita como os SCD e CBERS, o período orbital é de aproximadamente 100 minutos ou 6000 segundos.

Os parâmetros de simulação com telemetrias do satélite correspondentes são apresentados na Tabela 5.4.

Tabela 5.4. Parâmetros de simulação e telemetrias correspondentes.

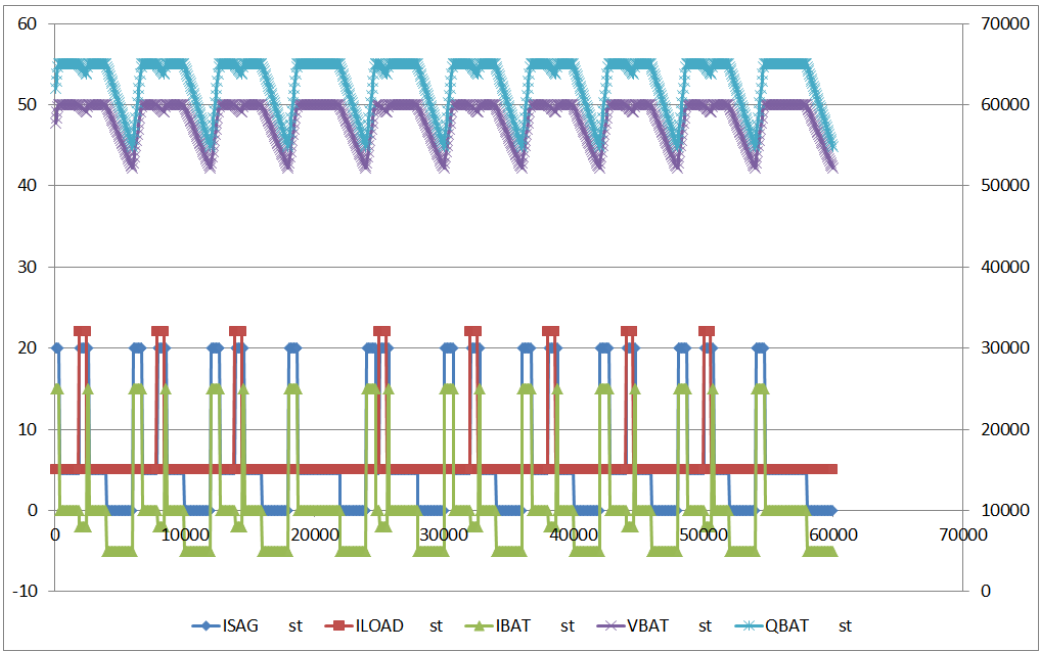
Parâmetro de simulação	**Parâmetro de telemetria**	**Descrição**
SUN	TM_SS	Telemetria de sensor solar
ISAG	TM_ISAG	Telemetria de corrente do painel solar
ILOAD	TM_IBUS	Telemetria de corrente do barramento principal
IBAT	TM_IBAT	Telemetria de corrente de carga/descarga da bateria
VBAT	TM_VBAT	Telemetria de tensão total da bateria

Fonte: Produção do autor.

Na Tabela 5.4 a primeira coluna corresponde ao identificador do parâmetro de simulação, a segunda o identificador da telemetria correspondente, e a terceira uma descrição resumida da telemetria.

A Figura 5.1 mostra a evolução dinâmica de cinco parâmetros do simulador durante uma sessão de simulação, conforme o modelo implantado por meio das regras de controle e o disparo temporizado de eventos da fila.

Figura 5.1. Evolução de parâmetros de simulação no cenário nominal.



Fonte: Produção do autor.

Na Figura 5.1 os parâmetros ISAG, ILOAD, IBAT e VBAT utilizam a escala de valores do eixo à esquerda (-10 a 60), ao passo que o parâmetro QBAT utiliza a escala de valores do eixo à direita (0 a 70000).

Analisando-se o gráfico, pode se perceber que:

- O valor da corrente de carga ILOAD acompanha os eventos de ligamento (EV_PL_ON) e desligamento (EV_PL_ON) da carga útil.

- O valor da corrente gerada pelo painel solar ISAG acompanha os ciclos de sol (EV_SUN, SUN = 1) e eclipse (EV_ECL, SUN = 0), combinado com os eventos de ligamento (EV_PL_ON) e desligamento (EV_PL_ON) da carga útil.

- O valor da corrente de bateria IBAT acompanha os ciclos de carga e descarga da bateria, sendo calculada pelas correntes de geração pelo painel solar (ISAG) e de consumo pela carga (ILOAD).

- Os valores da tensão (VBAT) e da carga de bateria (QBAT) acompanham a integral temporal da corrente de bateria (IBAT), sofrendo

aumento de valor quando há carregamento da bateria (IBAT > 0), manutenção de valor quando a bateria está cheia (IBAT = 0) e diminuição de valor quando há descarregamento da bateria (IBAT < 0).

5.1.2 Cenário complexo 1: degradação por envelhecimento

Este cenário descreve o funcionamento do suprimento de energia de um satélite após a degradação por envelhecimento da bateria, representada por meio das mesmas regras comportamentais e parâmetros de simulação do cenário anterior.

A diferença principal, no Estado Inicial, entre os dois cenários quanto ao modelo de simulação consiste na mudança de valor de um parâmetro, a carga máxima da bateria (QBATMAX), diminuída de 65000 para 63700 coulombs, ou o correspondente a uma queda de 2 %. Consequentemente, os valores iniciais de carga da bateria (QBAT) e tensão da bateria (VBAT) também são ajustados de 61100 para 59800 coulombs e de 47 para 46 volts, respectivamente. A Tabela 5.5 resume estas alterações.

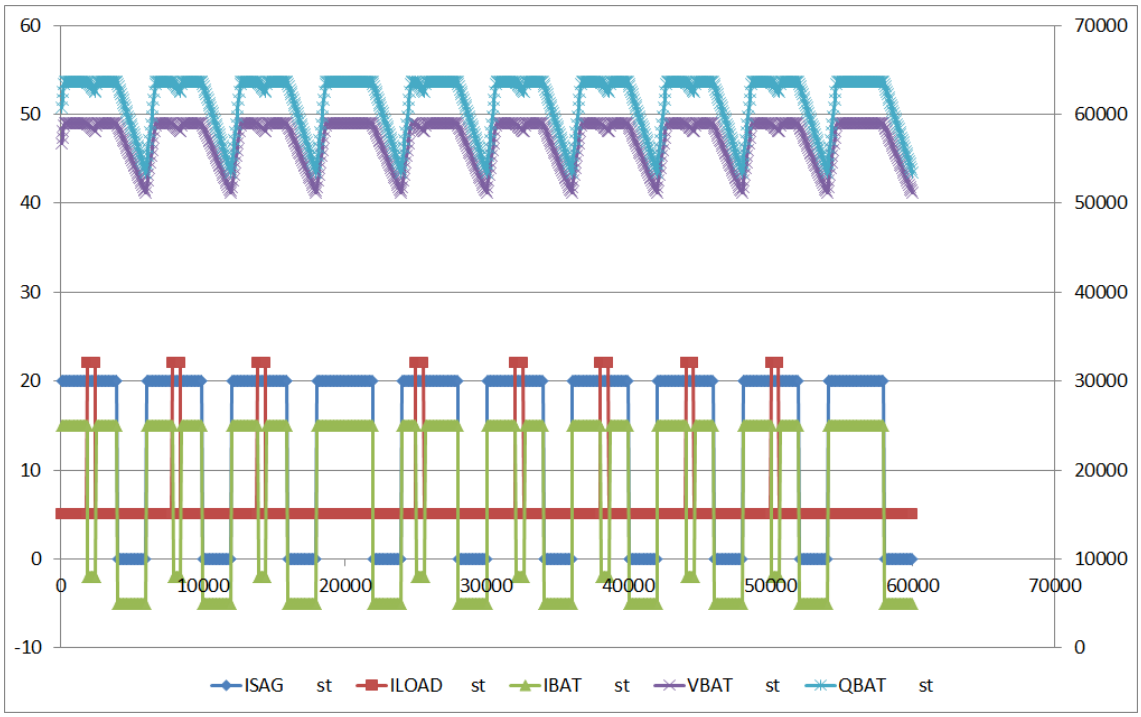
Tabela 5.5. Parâmetros alterados no estado inicial do cenário degradado.

Parâmetro	Tipo	Valor
QBATMAX	st	65000 → 63700
QBAT	st	61100 → 59800
VBAT	st	47 → 46

Fonte: Produção do autor.

O efeito desta pequena alteração pode ser percebido no resultado de saída da simulação, dos quais cinco parâmetros selecionados (ISAG, ILOAD, IBAT, VBAT e QBAT) são mostrados plotados na Figura 5.2.

Figura 5.2. Evolução de parâmetros de simulação no cenário degradado.



Fonte: Produção do autor.

Analisando-se comparativamente as plotagens dos dois cenários, percebe-se que:

- Os valores de QBAT e VBAT são sempre menores no novo cenário do que no cenário anterior.
- O valor de VBAT nunca atinge seu valor máximo VBATMAX.
- O valor de ISAG é mantido num nível alto durante toda a fase iluminada da órbita pelo controle de carga da bateria.
- O valor de IBAT também é mantido positivo durante toda a fase iluminada da órbita em que não há operação de carga útil, pois VBAT nunca atinge VBATMAX. Num cenário real, a manutenção desta situação causaria um eventual superaquecimento da bateria acelerando a sua degradação, portanto uma mudança da tensão máxima VBATMAX seria efetuada por telecomando.

O processo de adaptação do modelo comportamental do primeiro cenário (nominal) para o segundo cenário (degradado) passa pelas seguintes etapas:

- Identificação dos parâmetros afetados e não afetados
- Identificação das regras afetadas
- Identificação dos parâmetros de efeito
- Identificação dos parâmetros condicionantes
- Alteração dos parâmetros condicionantes
- Atualização do modelo comportamental

O processo de identificação de parâmetros afetados e não afetados leva à identificação dos seguintes parâmetros de simulação com telemetrias correspondentes ao longo da simulação.

- Parâmetros afetados: {ISAG, IBAT, VBAT}
- Parâmetros não afetados: {SUN, ILOAD}

A Tabela 5.6 lista os parâmetros de simulação com suas telemetrias correspondentes e mostra se o parâmetro em questão foi afetado ou não pela mudança de cenário.

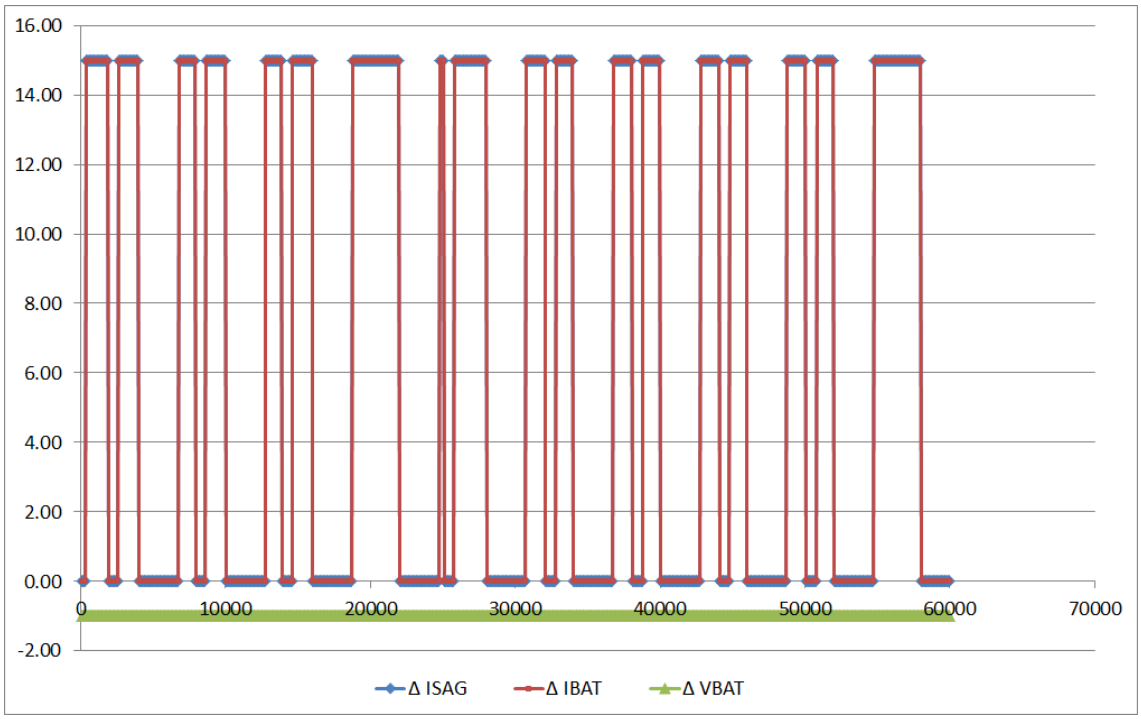
Tabela 5.6. Parâmetros de simulação e telemetrias no cenário degradado.

Parâmetro de simulação	**Parâmetro de telemetria**	**Descrição**	**Estado**
SUN	TM_SS	Telemetria de sensor solar	**Não afetada**
ISAG	TM_ISAG	Telemetria de corrente do painel solar	**Afetada**
ILOAD	TM_IBUS	Telemetria de corrente do barramento principal	**Não afetada**
IBAT	TM_IBAT	Telemetria de corrente de carga/descarga da bateria	**Afetada**
VBAT	TM_VBAT	Telemetria de tensão total da bateria	**Afetada**

Fonte: Produção do autor.

A Figura 5.3 mostra a plotagem temporal da diferença de valores entre parâmetros discrepantes de simulação e de telemetria ao longo de uma sessão de simulação.

Figura 5.3. Evolução de diferença de valores de parâmetros discrepantes.



Fonte: Produção do autor.

O processo de identificação de regras de atribuição a parâmetros afetados leva à identificação das seguintes regras de controle (Tabela 5.7).

Tabela 5.7. Regras de atribuição e parâmetros afetados no cenário degradado.

Regra	**Precondição**	**Efeitos**
R_ISAG_ECL	IF: SUN EQ 0	THEN: **ISAG** = 0
R_ISAG_SUN	IF: (SUN EQ 1) AND (VBAT LT VBATMAX)	THEN: **ISAG** = ISAGMAX
R_ISAG_FUL	IF: (SUN EQ 1) AND (VBAT GE VBATMAX)	THEN: **ISAG** = MIN (ILOAD, ISAGMAX)
R_IBAT	IF: 1	THEN: **IBAT** = ISAG - ILOAD
R_VBAT	IF: 1	THEN: **VBAT** = QBAT / CBAT

Fonte: Produção do autor.

Para cada parâmetro afetado, o conjunto de parâmetros candidatos potencialmente causadores dos novos efeitos é obtido a partir das expressões de precondição e de atribuição de efeito nas regras identificadas anteriormente. Mas destes, os parâmetros de telemetria não afetados devem ser eliminados, bem como parâmetros de controle de simulação (Tabela 5.8).

- ISAG: {VBAT, VBATMAX, ISAGMAX}

- IBAT: {ISAG}
- VBAT: {QBAT, CBAT}

Tabela 5.8. Parâmetros eliminados das regras de atribuição.

Regra	**Precondição**	**Efeitos**
R_ISAG_ECL	IF: ~~**SUN**~~ EQ 0	THEN: ISAG = 0
R_ISAG_SUN	IF: (~~**SUN**~~ EQ 1) AND (**VBAT** LT **VBATMAX**)	THEN: ISAG = **ISAGMAX**
R_ISAG_FUL	IF: (~~**SUN**~~ EQ 1) AND (**VBAT** GE **VBATMAX**)	THEN: ISAG = MIN (~~**ILOAD**~~, **ISAGMAX**)
R_IBAT	IF: 1	THEN: IBAT = **ISAG** - ~~**ILOAD**~~
R_VBAT	IF: 1	THEN: VBAT = **QBAT** / **CBAT**

Fonte: Produção do autor.

A partir de análise dos elementos dos conjuntos de parâmetros candidatos no passo anterior, é possível encontrar uma árvore hierárquica de causalidade dos parâmetros afetados.

- IBAT: {ISAG}
    - ISAG: {VBAT, VBATMAX, ISAGMAX}
        - VBAT: {QBAT, CBAT}

O conjunto dos parâmetros potencialmente afetados pode ser obtido a partir da união dos conjuntos de parâmetros candidatos, identificados no passo anterior para cada parâmetro afetado, de onde são excluídos os parâmetros já sabidamente afetados.

- Parâmetros potencialmente afetados: {VBATMAX, ISAGMAX, QBAT, CBAT}

A iteração seguinte é iniciada com a identificação de regras de atribuição aos parâmetros potencialmente afetados. A atualização deste passo leva à identificação de novas regras de controle conforme Tabela 5.9.

Tabela 5.9. Novas regras de atribuição identificadas no cenário degradado.

Regra	Precondição	Efeitos
R_QBAT	IF: QBAT LE QBATMAX	THEN: **QBAT** = QBAT + SIMSTEP * IBAT
R_QBAT_MAX	IF: QBAT GT QBATMAX	THEN: **QBAT** = QBATMAX

Fonte: Produção do autor.

Para cada parâmetro potencialmente afetado, o conjunto de novos parâmetros candidatos pode ser obtido a partir das expressões de precondição e de atribuição de efeito nas novas regras identificadas. Parâmetros de telemetria desnecessários devem ser eliminados, conforme Tabela 5.10.

VBATMAX: {} ; ISAGMAX: {}; QBAT: {QBAT, QBATMAX, IBAT}; CBAT: {}

Tabela 5.10. Parâmetros eliminados das novas regras de atribuição.

Regra	Precondição	Efeitos
R_QBAT	IF: **QBAT** LE **QBATMAX**	THEN: **QBAT** = **QBAT** + ~~**SIMSTEP**~~ * **IBAT**
R_QBAT_MAX	IF: **QBAT** GT **QBATMAX**	THEN: **QBAT** = **QBATMAX**

Fonte: Produção do autor.

A dependência de QBAT de si próprio e de IBAT constituem graficamente em loops na estrutura hierárquica de causalidade dos parâmetros.

A ausência de novas regras identificadas na iteração seguinte resulta no seguinte conjunto de novos parâmetros candidatos.

- QBATMAX: {}

Desta forma, a atualização da árvore hierárquica de causalidade dos parâmetros afetados resultaria na estrutura a seguir.

- IBAT: {ISAG}
    - ISAG: {VBAT, VBATMAX, ISAGMAX}
        - VBAT: {QBAT, CBAT}
            - QBAT: {QBAT, QBATMAX, IBAT}

        - QBAT (loop)
        - QBATMAX: {}
        - IBAT (loop)
    - CBAT: {}
- VBATMAX: {}
- ISAGMAX: {}

Desta forma, ao fim do processo de identificação dos parâmetros candidatos levaria à identificação dos seguintes parâmetros candidatos independentes.

VBATMAX: {}; ISAGMAX: {}; CBAT: {}; QBATMAX: {}

O próximo passo consiste em alterar os valores dos parâmetros anterior, um por vez, para verificar seu efeito imediato no parâmetro dependente em um ou mais passos de simulação.

As discrepâncias dos parâmetros de simulação com telemetrias correspondentes são verificadas pela primeira vez nos seguintes tempos de simulação.

- ISAG: 5 → 20 (ΔISAG = 15), em SIMTIME = 360
- IBAT: 5 → 20 (ΔIBAT = 15), em SIMTIME = 360
- VBAT: 47 → 46 (ΔVBAT = -1), em SIMTIME = 60

Como a discrepância de VBAT é verificada já no passo inicial da sessão de simulação, então são testados primeiramente os parâmetros dos quais VBAT depende. A regra de atribuição de efeito de VBAT neste passo é apresentada na Tabela 5.11.

Tabela 5.11. Última regra de atribuição de efeito.

Regra	**Precondição**	**Efeitos**
R_VBAT	IF: 1	THEN: VBAT = QBAT / CBAT

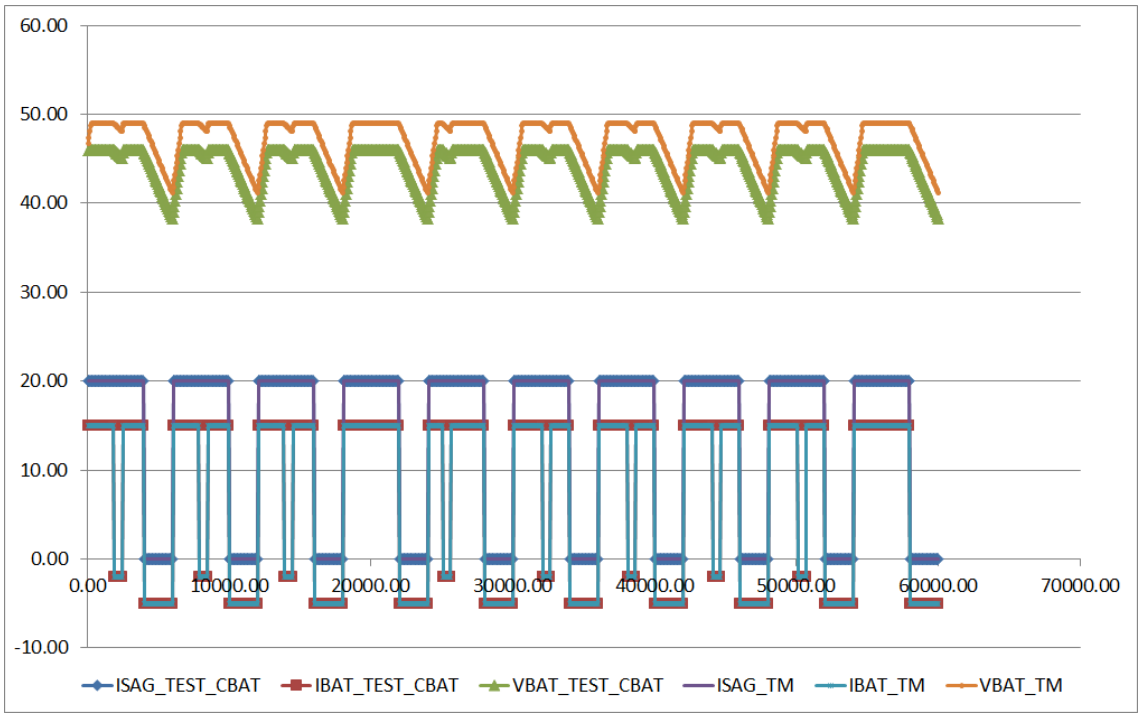
Fonte: Produção do autor.

Pela estrutura hierárquica, o parâmetro independente do qual VBAT depende imediatamente é CBAT.

- VBAT: {QBAT, CBAT}
    - QBAT: {QBAT, QBATMAX, IBAT}
        - QBAT (loop)
        - QBATMAX: {}
        - IBAT: {ISAG}
            - ISAG: {VBAT, VBATMAX, ISAGMAX}
                - VBAT (loop)
                - VBATMAX
                - ISAGMAX
    - CBAT: {}

A primeira tentativa consiste em alterar os valores do parâmetro CBAT para fazer casar os valores de VBAT. O resultado simulado após esta tentativa é mostrada a Figura 5.4.

Figura 5.4. Evolução de parâmetros discrepantes após primeira tentativa de casamento.



Fonte: Produção do autor.

A segunda tentativa consiste em alterar os valores do parâmetro QBAT por meio de QBATMAX para fazer casar os valores de VBAT.

Desta forma, os valores a serem testados para os parâmetros candidatos independentes seriam atribuídos conforme a seguir, com a discrepância sendo identificada.

- ISAGMAX = ISAG (= 20, quando SIMTIME = 420)
- VBATMAX = VBAT (= 46, quando SIMTIME = 0)
- QBATMAX = QBAT (= 60000, quando SIMTIME = 420)

As regras de atribuição que utilizam os parâmetros candidatos independentes como termos de expressões de precondição e/ou efeito são identificados conforme a Tabela 5.12.

Tabela 5.12. Parâmetros candidatos em expressões de regras.

Regra	**Precondição**	**Efeitos**
R_ISAG_SUN	IF: (SUN EQ 1) AND (VBAT LT **VBATMAX**)	THEN: ISAG = **ISAGMAX**
R_ISAG_FUL	IF: (SUN EQ 1) AND (VBAT GE **VBATMAX**)	THEN: ISAG = MIN (ILOAD, **ISAGMAX**)
R_QBAT_MAX	IF: QBAT GT **QBATMAX**	THEN: QBAT = **QBATMAX**

Fonte: Produção do autor.

Os valores iniciais de teste a serem atribuídos aos parâmetros candidatos independentes nas expressões de efeito são obtidos por meio dos valores dos parâmetros de atribuição (Tabela 5.13), no passo em que a discrepância é detectada pela primeira vez para aquele parâmetro de atribuição (Tabela 5.14).

Tabela 5.13. Parâmetros de atribuição a parâmetros candidatos de efeito.

Regra	**Precondição**	**Efeitos**
R_ISAG_SUN	IF: (SUN EQ 1) AND (VBAT LT VBATMAX)	THEN: **ISAG** = **ISAGMAX**
R_ISAG_FUL	IF: (SUN EQ 1) AND (VBAT GE VBATMAX)	THEN: **ISAG** = MIN (ILOAD, **ISAGMAX**)
R_QBAT_MAX	IF: QBAT GT QBATMAX	THEN: **QBAT** = **QBATMAX**

Fonte: Produção do autor.

Tabela 5.14. Atribuições de valores a parâmetros candidatos de efeito.

Regra	**Precondição**	**Efeitos**
R_ISAG_ECL	IF: SUN EQ 0	THEN: **ISAG** = 0
R_ISAG_SUN	IF: (SUN EQ 1) AND (VBAT LT VBATMAX)	THEN: **ISAG** = ISAGMAX
R_ISAG_FUL	IF: (SUN EQ 1) AND (VBAT GE VBATMAX)	THEN: **ISAG** = MIN (ILOAD, ISAGMAX)
R_IBAT	IF: 1	THEN: **IBAT** = ISAG - ILOAD
R_VBAT	IF: 1	THEN: **VBAT** = QBAT / CBAT

Fonte: Produção do autor.

Os valores iniciais de teste a serem atribuídos aos parâmetros candidatos independentes nas expressões de precondição são obtidos por meio dos valores dos parâmetros que compõem os termos da expressão com mesma profundidade de aninhamento, no mesmo passo em que a expressão de condição correspondente é satisfeita (Tabela 5.15).

Tabela 5.15. Atribuições de valores a parâmetros candidatos de precondição.

Regra	Precondição	Efeitos
R_ISAG_SUN	IF: (SUN EQ 1) AND (**VBAT** LT **VBATMAX**)	THEN: ISAG = ISAGMAX
R_ISAG_FUL	IF: (SUN EQ 1) AND (**VBAT** GE **VBATMAX**)	THEN: ISAG = MIN (ILOAD, ISAGMAX)
R_QBAT_MAX	IF: **QBAT** GT **QBATMAX**	THEN: QBAT = QBATMAX

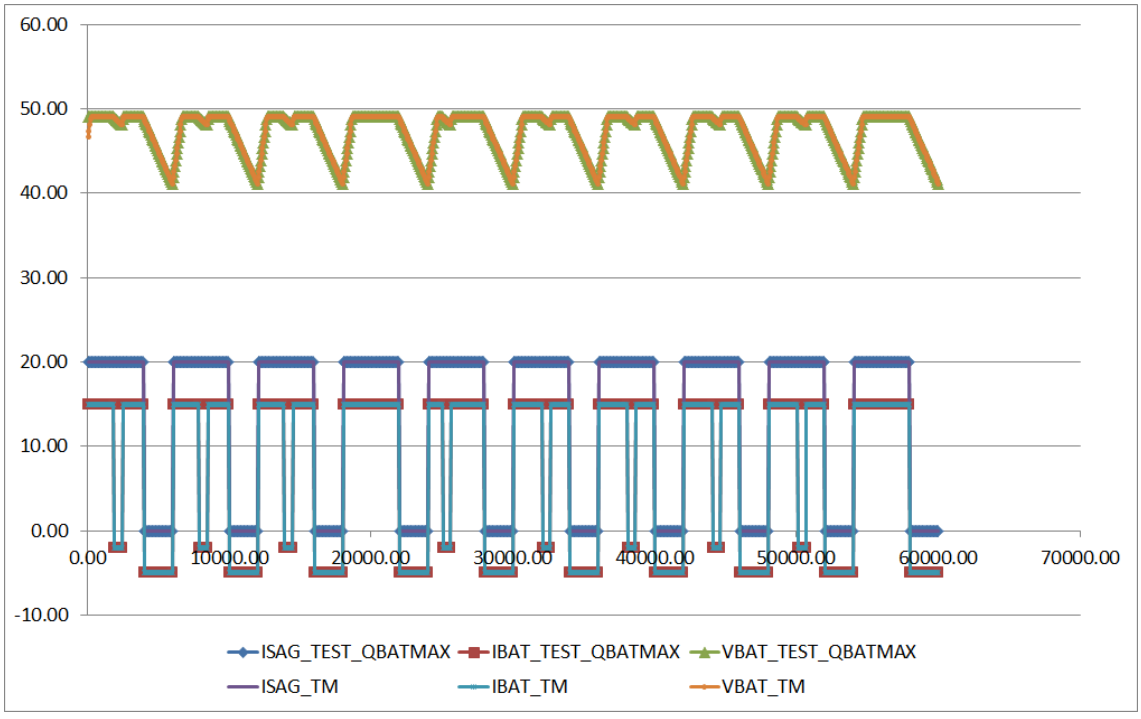
Fonte: Produção do autor.

Desta forma, os valores a serem testados para os parâmetros candidatos independentes seriam atribuídos conforme a seguir, com a discrepância sendo identificada já no primeiro passo neste cenário.

- ISAGMAX = ISAG (= 20, quando SIMTIME = 420)
- VBATMAX = VBAT (= 47, quando SIMTIME = 300)
- QBATMAX = QBAT (= 60000, quando SIMTIME = 420)

O resultado simulado após esta tentativa é mostrado a seguir (Tabela 5.7).

Figura 5.5. Evolução de parâmetros discrepantes após segunda tentativa de casamento.



Fonte: Produção do autor.

Uma vez obtido um conjunto de parâmetros candidatos que leve a um resultado de simulação que leve a uma discrepância, medida por meio de uma função custo, menor do que um erro predefinido, o modelo é atualizado com os parâmetros candidatos e a busca é finalizada.

Caso não seja possível encontrar uma solução deterministicamente por meio de análise de correlação de regras e parâmetros, parte-se para tentativas aleatórias, ou seja, a aplicação da heurística para adaptação de regras com programação genética.

5.1.3 Cenário complexo 2: falha no suprimento de energia

Este cenário descreve o funcionamento anormal do suprimento de energia de um satélite após uma falha, representada por meio do mesmo conjunto de regras comportamentais e parâmetros de simulação do cenário anterior, a menos da substituição de uma regra de controle. Este cenário descreve uma situação hipotética em que a corrente de carga da bateria é reduzida à metade devido a uma falha em circuito aberto na linha de alimentação pelo painel solar.

Esta situação pode ser descrita por meio de uma alteração de valor de um parâmetro (ISAGMAX) após o evento de falha, de forma similar ao cenário anterior, conforme a Tabela 5.16.

Tabela 5.16. Alteração de valor de parâmetro no cenário de falha.

Parâmetro	Tipo	Valor anterior	Valor posterior
ISAGMAX	st	20	10

Fonte: Produção do autor.

Alternativamente, o modelo comportamental pode ser reproduzido por meio da alteração de uma regra (R_ISAG_SUN). Esta alteração não é única, havendo várias variantes igualmente aceitáveis, conforme a Tabela 5.17.

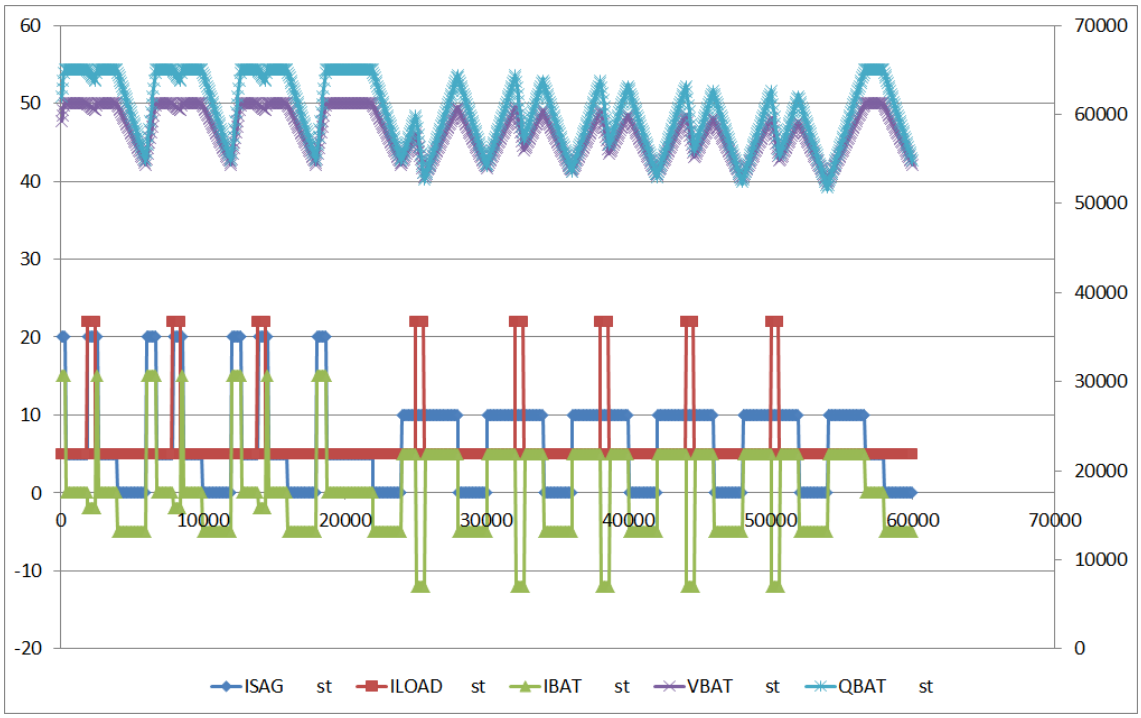
Tabela 5.17. Exemplos de regras alternativas no cenário de falha.

Regra	Precondição	Efeitos
Regra original	IF: (SUN EQ 1) AND (VBAT LT VBATMAX)	THEN: ISAG = ISAGMAX
Regras alternativas	IF: (SUN EQ 1) AND (VBAT LT VBATMAX)	THEN: ISAG = ISAGMAX/2
	IF: (SUN EQ 1) AND (VBAT LT VBATMAX)	THEN: ISAG = ISAGMAX – 10
	IF: (SUN EQ 1) AND (VBAT LT VBATMAX)	THEN: ISAG = 10

Fonte: Produção do autor.

A Figura 5.6 ilustra o resultado da aplicação de qualquer uma das alterações acima na evolução dos parâmetros de simulação de uma sessão de simulação após a introdução da falha no tempo SIMTIME = 20000. Percebe-se uma demora maior até a bateria atingir o fim de carga, que pode até mesmo não ser alcançado em alguns ciclos orbitais, dependendo do padrão de operação de carga útil.

Figura 5.6. Evolução de parâmetros de simulação no cenário de falha.



Fonte: Produção do autor.

O processo de identificação de parâmetros afetados e de regras de atribuição a parâmetros afetados leva à identificação das regras de controle conforme a Seção anterior (5.1.2).

5.2 Alteração de regras com programação genética - cenários simples

Os cenários simples consistem em conjuntos pequenos de regras comportamentais e parâmetros de simulação, que foram concebidos para fornecer um conjunto de dados para execuções em série do simulador para obtenção de um volume de dados estatísticos para a prova de conceito de geração de novas regras com uso de programação genética.

Cinco cenários simples foram especificados e exercitam o módulo de programação genética (**genetic_programming**) descrito na Seção A.4 do Apêndice A.

5.2.1 Descrição do Modelo Comportamental

Os cenários simples são ilustrados tendo como Regras de Controle iniciais o conjunto de quatro regras descrito na Tabela 5.18.

Tabela 5.18. Regras de Controle iniciais componentes dos cenários simples.

Regra	**Precondição**	**Efeitos**
RNEXT	IF: 1	THEN: SIMTIME = SIMTIME + SIMSTEP
RSTOP	IF: SIMTIME GE SIMEND	THEN: SIMSTOP = 1
R1	IF: 1	THEN: X = X + 1
R2	IF: Y	THEN: Y = Y + 2

Fonte: Produção do autor.

As primeiras duas regras da Tabela 5.18 descrevem o controle de execução do simulador através dos parâmetros de sistema do simulador, a saber:

- SIMTIME - tempo de simulação
- SIMSTEP - passo de simulação
- SIMSTOP - condição de parada da simulação
- SIMEND - tempo de parada

- RNEXT - determina a taxa de incremento do tempo de simulação (SIMTIME) a cada passo de simulação (SIMSTEP).

- RSTOP - verifica a condição de parada da simulação (SIMSTOP), que ocorre quando o tempo de simulação (SIMTIME) atingir um valor maior ou igual ao tempo de parada (SIMEND).

As duas regras seguintes definem o comportamento dos parâmetros de simulação:

- R1 sempre atribui ao parâmetro X o valor de X do passo anterior acrescido de um.

- R2 sempre atribui ao parâmetro Y o valor de Y do passo anterior acrescido de dois.

A este conjunto de regras associam-se parâmetros iniciais conforme descrito na Tabela 5.19.

Tabela 5.19. Valores de parâmetros do Estado Inicial dos cenários simples.

Parâmetro	Tipo	Valor
SIMTIME	St	0
SIMEND	St	20
SIMSTEP	St	1
SIMSTOP	Ev	0
X	St	0
Y	St	1

Fonte: Produção do autor.

5.2.2 Cenário simples 1: substituição de um parâmetro de simulação

Em um caso real de funcionamento de um satélite, quando uma falha é detectada em um equipamento, um mecanismo de tolerância a falhas dispara um ligamento ou desligamento automático do equipamento em bordo. Após o evento de chaveamento, um parâmetro de monitoração de um equipamento que operava no módulo principal passa a monitorar o mesmo equipamento no módulo redundante, o qual pode ter características ligeiramente diferentes.

Esta situação levaria uma alteração no modelo do simulador representada pela substituição de um parâmetro de simulação em uma regra.

Baseado nas Regras de Controle iniciais (Tabela 5.18), este cenário consiste na modificação de uma regra (R2), resultando no conjunto de quatro regras conforme descrito na tabela a seguir (Tabela 5.20).

Tabela 5.20. Regras de Controle componentes do cenário simples 1.

Regra	**Precondição**	**Efeitos**
RNEXT	IF: 1	THEN: SIMTIME = SIMTIME + SIMSTEP
RSTOP	IF: SIMTIME GE SIMEND	THEN: SIMSTOP = 1
R1	IF: 1	THEN: X = X + 1
R2	IF: Y	THEN: Y = **X** + 2

Fonte: Produção do autor.

Na regra R2, no lado direito da expressão de efeito, o parâmetro Y foi substituído pelo parâmetro X.

- De: R2 sempre atribui ao parâmetro Y o valor de Y do passo anterior acrescido de dois.

- Para: R2 sempre atribui ao parâmetro Y o valor de X do passo anterior acrescido de dois.

A Tabela 5.21 sumariza o resultado de obtenção de combinações de regras alternativas em substituição a regras originais que resultaram em erro zero entre saídas do conjunto de regras modificado e regras alternativas obtidas após 20 execuções consecutivas do algoritmo de programação genética com parada forçada após três gerações. Nesta tabela, a coluna **Execução** indica o número de vezes em que o processo de busca com programação genética foi reinicializado, a coluna **Iteração** indica um identificador numérico para um conjunto de regras alteradas candidato para uma possível solução, a coluna **Combinação de regras alternativas** apresenta as regras alteradas e a coluna **Soluções** apresenta o número de soluções válidas obtida para a execução.

Tabela 5.21. Resultados de execução do cenário simples 1.

Execução	Iteração	Combinação de regras alternativas	Soluções
0	16	('R_1_C0_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['X']; ['+']; ['1']]])	1
0	16	('R_2_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	1
1	0	('R_1_C0_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['X']; ['+']; ['1']]])	6
1	0	('R_2_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	6
1	33	('R_1_C0_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['X']; ['+']; ['1']]])	6
1	33	('R_2_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	6
1	35	('R_2_M1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	6
1	45	('R_2_C0_1_M1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	6
1	67	('R_1_C0_0_C1_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['X']; ['+']; ['1']]])	6
1	67	('R_2_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	6
1	68	('R_1_C0_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['X']; ['+']; ['1']]])	6
1	68	('R_2_M0_C1_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	6
2	14	('R_2_C0_1_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	1
3	0	('R_2_C0_1_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	5
3	33	('R_2_C0_1_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	5
3	36	('R_2_M1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	5
3	67	('R_2_C0_1_M0_C1_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	5
3	68	('R_1_C1_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['X']; ['+']; ['1']]])	5
3	68	('R_2_C0_1_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	5
4	10	('R_2_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	1
5	0	('R_2_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	5
5	33	('R_2_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	5
5	39	('R_2_C0_1_M1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	5
5	67	('R_2_M0_C1_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	5
5	68	('R_1_C1_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['X']; ['+']; ['1']]])	5
5	68	('R_2_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	5
7	54	('R_1_C0_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['X']; ['+']; ['1']]])	1
7	54	('R_2_C0_1_M1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	1
9	52	('R_1_C0_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['X']; ['+']; ['1']]])	1
9	52	('R_2_C0_1_M1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	1
11	48	('R_1_C0_0_C0_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['X']; ['+']; ['1']]])	1
11	48	('R_2_M1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	1
12	3	('R_2_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	4
12	15	('R_1_C0_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['X']; ['+']; ['1']]])	4
12	15	('R_2_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	4
12	23	('R_1_C0_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['X']; ['+']; ['1']]])	4
12	23	('R_2_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	4
12	24	('R_2_M0_C0_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	4

continua

Tabela 5.21. Conclusão.

Execução	Iteração	Combinação de regras alternativas	Soluções
13	0	('R_2_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	13
13	1	('R_1_C0_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['X']; ['+']; ['1']]])	13
13	1	('R_2_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	13
13	2	('R_1_C0_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['X']; ['+']; ['1']]])	13
13	2	('R_2_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	13
13	3	('R_2_M0_C0_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	13
13	33	('R_2_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	13
13	67	('R_1_C1_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['X']; ['+']; ['1']]])	13
13	67	('R_2_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	13
13	68	('R_2_M0_C1_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	13
13	69	('R_1_C0_0_C1_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['X']; ['+']; ['1']]])	13
13	69	('R_2_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	13
13	70	('R_1_C0_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['X']; ['+']; ['1']]])	13
13	70	('R_2_M0_C1_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	13
13	71	('R_1_C0_0_C1_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['X']; ['+']; ['1']]])	13
13	71	('R_2_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	13
13	72	('R_1_C0_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['X']; ['+']; ['1']]])	13
13	72	('R_2_M0_C1_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	13
13	73	('R_1_C1_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['X']; ['+']; ['1']]])	13
13	73	('R_2_M0_C0_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	13
13	74	('R_2_M0_C0_1_C1_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	13
15	36	('R_2_M0_M1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	1
16	15	('R_2_C0_1_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	1
17	0	('R_2_C0_1_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	5
17	33	('R_2_C0_1_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	5
17	47	('R_1_C0_0_C0_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['X']; ['+']; ['1']]])	5
17	47	('R_2_M1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	5
17	67	('R_2_C0_1_M0_C1_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	5
17	68	('R_1_C1_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['X']; ['+']; ['1']]])	5
17	68	('R_2_C0_1_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	5
18	11	('R_2_M0_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	1
19	0	('R_2_M0_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	6
19	33	('R_2_M0_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	6
19	45	('R_2_C0_1_M1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	6
19	55	('R_2_C0_1_C0_1_M1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	6
19	67	('R_1_C1_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['X']; ['+']; ['1']]])	6
19	67	('R_2_M0_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	6
19	68	('R_2_M0_M0_C1_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	6

Fonte: Produção do autor.

Os resultados acima indicam probabilidade de 16/20 de se encontrar algum resultado aceitável (erro zero) em três gerações.

Em algumas execuções são encontradas múltiplas soluções, como no caso da execução 13 em que são obtidas 13 soluções viáveis. É esperado que o aumento do número máximo permitido de gerações leve à eventual obtenção de algum resultado válido, em troca de aumento do custo computacional.

5.2.3 Cenário simples 2: troca de dois parâmetros de simulação

Este cenário ilustra uma situação real de comutação automática de equipamento em bordo disparada por um mecanismo de proteção. Após o evento, o parâmetro inicialmente associado ao equipamento principal passa a ser aquele associado ao equipamento redundante e vice-versa. Este cenário real é representado no simulador pela troca entre dois parâmetros de simulação em um conjunto de regras.

Este cenário simplificado consiste num conjunto de quatro regras conforme descrito na Tabela 5.22, baseado nas regras iniciais (Tabela 5.18) com a modificação de duas delas.

Tabela 5.22. Regras de Controle componentes do cenário simples 2.

Regra	**Precondição**	**Efeitos**
RNEXT	IF: 1	THEN: SIMTIME = SIMTIME + SIMSTEP
RSTOP	IF: SIMTIME GE SIMEND	THEN: SIMSTOP = 1
R1	IF: 1	THEN: X = **Y** + 1
R2	IF: Y	THEN: Y = **X** + 2

Fonte: Produção do autor.

Na regra R1, no lado direito da expressão de efeito, o parâmetro X foi substituído pelo parâmetro Y.

- De: R1 sempre atribui ao parâmetro X o valor de X do passo anterior acrescido de um.
- Para: R1 sempre atribui ao parâmetro X o valor de Y do passo anterior acrescido de um.

Na regra R2, no lado direito da expressão de efeito, o parâmetro Y foi substituído pelo parâmetro X.

- De: R2 sempre atribui ao parâmetro Y o valor de Y do passo anterior acrescido de dois.

- Para: R2 sempre atribui ao parâmetro Y o valor de X do passo anterior acrescido de dois.

A Tabela 5.23 sumariza o resultado de obtenção de combinações de regras alternativas em substituição a regras originais que resultaram em erro zero entre saídas do conjunto de regras modificado e regras alternativas obtidas após 14 execuções consecutivas do algoritmo de programação genética com parada forçada após três gerações.

Tabela 5.23. Resultados de execução do cenário simples 2.

Execução	Iteração	Combinação de regras alternativas	Soluções
4	11	('R_1_M0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	1
4	11	('R_2_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	1
5	0	('R_1_M0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	4
5	0	('R_2_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	4
5	33	('R_1_M0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	4
5	33	('R_2_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	4
5	67	('R_1_M0_C1_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	4
5	67	('R_2_M0'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	4
5	68	('R_1_M0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	4
5	68	('R_2_M0_C1_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	4

Fonte: Produção do autor.

Em algumas execuções são encontradas múltiplas soluções, como no caso da execução 5 em que são obtidas quatro soluções viáveis.

Os resultados acima indicam probabilidade de 2/14 de se encontrar algum resultado aceitável (erro zero) em três gerações. O aumento do número máximo permitido de gerações leva à eventual obtenção de algum resultado válido em troca de aumento do custo computacional. A Tabela 5.24 mostra parcialmente o sumário do resultado de obtenção de combinações de regras alternativas em substituição a regras originais que resultaram em erro zero

entre saídas do conjunto de regras inicial e regras alternativas obtidas após cinco execuções consecutivas do algoritmo de programação genética com parada forçada após dez gerações.

Tabela 5.24. Parte dos resultados de execução do cenário simples 1 com 10 gerações.

Execução	Iteração	Combinação de regras alternativas	Soluções
0	2866	('R_1_C0_0_C2_0_M4'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	4
0	2866	('R_2_C0_1_M1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	4
0	2882	('R_1_C0_0_M4'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	4
0	2882	('R_2_C0_1_M1_C3_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	4
0	2885	('R_1_C0_0_C3_0_M4'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	4
0	2885	('R_2_C0_1_M1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	4
0	2886	('R_1_M4'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	4
0	2886	('R_2_C0_1_M2_C3_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	4
1	...	...	44
1	17074	('R_1_M4_C5_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	44
1	17074	('R_2_C0_1_M2_C3_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	44
2	...	...	263
2	35311	('R_1_M1_C4_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	263
2	35311	('R_2_C0_1_C2_1_C3_1_C5_1_M6'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	263
2	...	...	263
2	42134	('R_1_C0_0_C2_0_M6'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	263
2	42134	('R_2_C0_1_M1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['X']; ['+']; ['2']]])	263

Fonte: Produção do autor.

5.2.4 Cenário simples 3: modificação de um valor

Num caso real, uma situação de falha em um equipamento seria representada pela modificação do valor de um único parâmetro. Após o evento, o valor de um parâmetro inicialmente associado ao status de funcionamento de um equipamento embarcado passa a ser outro correspondente a um estado anômalo.

Este cenário simplificado consiste num conjunto de quatro regras conforme descrito na Tabela 5.23, baseado nas regras iniciais (Tabela 5.18) com a modificação de duas delas.

Tabela 5.23. Regras de Controle componentes do cenário simples 3.

Regra	Precondição	Efeitos
RNEXT	IF: 1	THEN: SIMTIME = SIMTIME + SIMSTEP
RSTOP	IF: SIMTIME GE SIMEND	THEN: SIMSTOP = 1
R1	IF: 1	THEN: X = X + 2
R2	IF: Y	THEN: Y = Y + 2

Fonte: Produção do autor.

Na regra R1, no lado direito da expressão de efeito, o valor 1 foi substituído pelo valor 2.

- De: R1 sempre atribui ao parâmetro X o valor de X do passo anterior acrescido de um.

- Para: R1 sempre atribui ao parâmetro X o valor de X do passo anterior acrescido de dois.

A Tabela 5.26 sumariza o resultado de obtenção de combinações de regras alternativas em substituição a regras originais que resultaram em erro zero entre saídas do conjunto de regras inicial e regras alternativas obtidas após 20 execuções consecutivas do algoritmo de programação genética com parada forçada após três gerações.

Tabela 5.24. Resultados de execução do cenário simples 3.

Execução	Iteração	Combinação de regras alternativas	Soluções
1	37	('R_1_C0_0_M1'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	2
1	50	('R_1_C0_0_C0_0_M1'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	2
2	3	('R_1_M0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	3
2	23	('R_1_M0_C0_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	3
2	24	('R_1_M0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	3
2	24	('R_2_C0_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['Y']; ['+']; ['2']]])	3
3	0	('R_1_M0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	10
3	1	('R_1_M0_C0_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	10
3	2	('R_1_M0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	10
3	2	('R_2_C0_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['Y']; ['+']; ['2']]])	10
3	33	('R_1_M0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	10
3	67	('R_1_M0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	10
3	67	('R_2_C1_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['Y']; ['+']; ['2']]])	10

continua

Tabela 5.25. Continuação.

Execução	Iteração	Combinação de regras alternativas	Soluções
3	68	('R_1_M0_C1_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	10
3	69	('R_1_M0_C0_0_C1_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	10
3	70	('R_1_M0_C0_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	10
3	70	('R_2_C1_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['Y']; ['+']; ['2']]])	10
3	71	('R_1_M0_C1_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	10
3	71	('R_2_C0_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['Y']; ['+']; ['2']]])	10
3	72	('R_1_M0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	10
3	72	('R_2_C0_1_C1_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['Y']; ['+']; ['2']]])	10
7	51	('R_1_C0_0_M1'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	1
7	51	('R_2_C0_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['Y']; ['+']; ['2']]])	1
9	43	('R_1_M1'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	2
9	43	('R_2_C0_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['Y']; ['+']; ['2']]])	2
9	45	('R_1_C0_0_C0_0_M1'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	2
11	34	('R_1_M1'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	3
11	35	('R_1_M1'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	3
11	38	('R_1_C0_0_M1'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	3
12	9	('R_1_M0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	1
13	0	('R_1_M0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	7
13	33	('R_1_M0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	7
13	41	('R_1_M1'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	7
13	44	('R_1_M1'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	7
13	44	('R_2_C0_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['Y']; ['+']; ['2']]])	7
13	48	('R_1_M1'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	7
13	48	('R_2_C0_1_C0_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['Y']; ['+']; ['2']]])	7
13	67	('R_1_M0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	7
13	67	('R_2_C1_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['Y']; ['+']; ['2']]])	7
13	68	('R_1_M0_C1_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	7
17	45	('R_1_C0_0_M1'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	4
17	45	('R_2_C0_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['Y']; ['+']; ['2']]])	4
17	51	('R_1_C0_0_C0_0_M1'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	4
17	57	('R_1_M0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	4
17	57	('R_2_M0_M1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['Y']; ['+']; ['2']]])	4
17	66	('R_1_M0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	4
17	66	('R_2_M0_M1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['Y']; ['+']; ['2']]])	4
18	9	('R_1_M0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	1
19	0	('R_1_M0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	5
19	33	('R_1_M0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	5
19	50	('R_1_C0_0_M1'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	5

continua

Tabela 5.26. Conclusão.

Execução	Iteração	Combinação de regras alternativas	Soluções
19	50	('R_2_C0_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['Y']; ['+']; ['2']]])	5
19	67	('R_1_M0_C1_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	5
19	68	('R_1_M0'; ['cond'; [['1']]]; ['effect'; [['X']; ['Y']; ['+']; ['1']]])	5
19	68	('R_2_C1_1'; ['cond'; [['1']]]; ['effect'; [['Y']; ['Y']; ['+']; ['2']]])	5

Fonte: Produção do autor

Nos resultados acima, em nenhuma solução encontrada a R1 foi alterada para a expressão de efeito (X ← X + 2) conforme esperado, mas todas as soluções obtidas apontam para a expressão de efeito (X ← Y + 1), cujo resultado é equivalente devido aos valores no estado inicial de X (0) e Y (1), conforme Tabela 10.

- Esperado: R1 sempre atribui ao parâmetro X o valor de X do passo anterior acrescido de dois.

- Obtido: R1 sempre atribui ao parâmetro X o valor de Y do passo anterior acrescido de um.

Para evitar esta solução, um novo conjunto de regras é descrito na Tabela 5.27.

Tabela 5.27. Regras de Controle componentes do cenário simples 3 alternativo.

Regra	Precondição	Efeitos
RNEXT	IF: 1	THEN: SIMTIME = SIMTIME + SIMSTEP
RSTOP	IF: SIMTIME GE SIMEND	THEN: SIMSTOP = 1
R1	IF: 1	THEN: X = X + 3
R2	IF: Y	THEN: Y = Y + 2

Fonte: Produção do autor.

Na regra R1, no lado direito da expressão de efeito, o valor 1, que havia sido substituído pelo valor 2, agora foi substituído pelo valor 3.

- De: R1 sempre atribui ao parâmetro X o valor de X do passo anterior acrescido de um.

- Para: R1 sempre atribui ao parâmetro X o valor de X do passo anterior acrescido de três.

Após 20 execuções consecutivas do algoritmo de programação genética com parada forçada após três gerações, não foi obtido nenhuma solução válida com erro zero entre saídas de simulação usando o novo conjunto de regras modificadas e a ferramenta de programação genética. Este resultado pode ser explicado pela dificuldade, para não se dizer impossibilidade, em se encontrar aleatoriamente um valor numérico real igual a outro em um universo de busca de mais a menos infinito. Para a obtenção de valores numéricos de parâmetros, propõe-se uma abordagem analítica conforme descrito na Seção 0.

5.2.5 Cenário simples 4: eliminação de uma expressão

Num caso real, a corrente gerada pelo painel solar alimenta o satélite por meio de vários canais. Em caso de curto em um destes canais, a corrente deste canal é zerada e a corrente total gerada pelo painel solar, que é dada pela soma das correntes parciais de cada canal, perde um dos termos. Esta situação é representada pela eliminação de uma expressão em uma regra.

Este cenário simplificado consiste em um conjunto de quatro regras conforme descrito na Tabela 5.28, baseado nas regras iniciais (Tabela 5.18) com a modificação de duas delas.

Tabela 5.28. Regras de Controle componentes do cenário simples 4.

Regra	**Precondição**	**Efeitos**
RNEXT	IF: 1	THEN: SIMTIME = SIMTIME + SIMSTEP
RSTOP	IF: SIMTIME GE SIMEND	THEN: SIMSTOP = 1
R1	IF: 1	THEN: X = X + 2
R2	IF: Y	THEN: Y = 2

Fonte: Produção do autor.

Na regra R2, no lado direito da expressão de efeito, o termo (Y +) foi eliminado.

- De: R2 sempre atribui ao parâmetro Y o valor de Y do passo anterior acrescido de dois.
- Para: R2 sempre atribui ao parâmetro Y o valor dois.

A Tabela 5.29 sumariza o resultado de obtenção de combinações de regras alternativas em substituição a regras originais que resultaram em erro zero entre saídas do conjunto de regras inicial e regras alternativas obtidas após 20 execuções consecutivas do algoritmo de programação genética com parada forçada após três gerações.

Tabela 5.29. Resultados de execução do cenário simples 4.

Execução	Iteração	Combinação de regras alternativas	Soluções
15	39	('R_2_C0_1_M1'; ['cond'; [['1']]]; ['effect'; [['Y']; [['Y']; ['MAX']; ['2']]]])	1

Fonte: Produção do autor.

Aqui, o resultado mostra que uma solução válida foi encontrada. Entretanto, percebe-se que não foi encontrada a solução em que regra R1 foi alterada para a expressão de efeito (Y ← 2) conforme esperado, mas a solução obtida aponta para uma nova expressão de efeito (X ← MAX(Y, 2)), cujo resultado é equivalente devido aos valores no estado inicial dos parâmetros X (0) e Y (1), conforme Tabela 5.19.

- Esperado: R2 sempre atribui ao parâmetro Y o valor dois.
- Obtido: R2 sempre atribui ao parâmetro Y o máximo entre o valor de Y no passo anterior e o valor dois.

No contexto das execuções realizadas como prova de conceito, o resultado acima indica probabilidade de 1/20 de se encontrar algum resultado aceitável (erro zero) em três gerações.

Nesta Tese, a programação genética foi implementada utilizando funções aleatórias para a decisão de aplicação da operação de mutação ou crossover, bem como a escolha aleatória do ponto de aplicação desta operação uniformemente com base no número de elementos da expressão.

Koza (1994) escreveu que cada nova geração foi criada da geração anterior aplicando a operação de reprodução para 10% da população e aplicando a operação de cruzamento a 90% da população. Na seleção de pontos de crossover, 90% teriam sido nós internos e 10% eram nós terminais da árvore

sintática. Talvez seja interessante um estudo mais aprofundado para definir pesos mais adequados para a aplicação da operação de mutação ou crossover e seus pontos de aplicação, por meio de abordagem matemática como em Altenberg (1994).

5.2.6 Cenário simples 5: inclusão de uma expressão

Num caso real, de degradação de uma bateria tem-se que o envelhecimento da bateria causa perda da capacidade de carga e descarga da bateria, que poderia ser representada como a adição de uma impedância parasita, que dissipa parte da energia que deveria ser armazenada em função da corrente de carga. Esta situação seria caracterizada pela inclusão de uma expressão a regra.

Este cenário simplificado consiste num conjunto de quatro regras conforme descrito na Tabela 5.30, baseado nas regras iniciais (Tabela 5.18) com a modificação de duas delas.

Tabela 5.30. Regras de Controle componentes do cenário simples 4.

Regra	**Precondição**	**Efeitos**
RNEXT	IF: 1	THEN: SIMTIME = SIMTIME + SIMSTEP
RSTOP	IF: SIMTIME GE SIMEND	THEN: SIMSTOP = 1
R1	IF: 1	THEN: X = X + 1
R2	IF: Y	THEN: Y = Y + 2 + X

Fonte: Produção do autor.

Na regra R2, no lado direito da expressão de efeito, o termo (+ X) foi acrescido.

- De: R2 sempre atribui ao parâmetro Y o valor de Y do passo anterior acrescido de dois.

- Para: R2 sempre atribui ao parâmetro Y o valor de Y do passo anterior acrescido de dois mais o valor o parâmetro X.

A Tabela 5.31 sumariza o resultado de obtenção de combinações de regras alternativas em substituição a regras originais que resultaram em erro zero entre saídas do conjunto de regras inicial e regras alternativas obtidas após 20

execuções consecutivas do algoritmo de programação genética com parada forçada após três gerações.

Tabela 5.31. Resultados de execução do cenário simples 5.

Execução	Iteração	Combinação de regras alternativas	Soluções
1	42	('R_2_M1'; ['cond'; [['1']]]; ['effect'; [['Y']; [['Y']; ['+']; ['X']]; ['+']; ['2']]])	1
5	37	('R_2_C0_1_M1'; ['cond'; [['1']]]; ['effect'; [['Y']; [['Y']; ['+']; ['X']]; ['+']; ['2']]])	1
9	36	('R_1_C0_0'; ['cond'; [['1']]]; ['effect'; [['X']; ['X']; ['+']; ['1']]])	1
9	36	('R_2_M1'; ['cond'; [['1']]]; ['effect'; [['Y']; [['Y']; ['+']; ['X']]; ['+']; ['2']]])	1

Fonte: Produção do autor.

É esperado que o aumento do número máximo permitido de gerações leve à eventual obtenção de algum resultado válido, em troca de aumento do custo computacional. O resultado acima indica probabilidade de 3/20 de se encontrar algum resultado aceitável (erro zero) em três gerações.

5.3 Considerações

Esta Seção apresentou provas de conceito para a proposta de identificação e modificação de parâmetros para geração de novos modelos comportamentais baseados em regras. Uma abordagem analítica de regras para pré-processamento das mesmas restringe o conjunto de regras e o número de parâmetros relevantes, aumentando a chance de obtenção de resultados válidos, conforme descrito nos cenários complexos (5.1).

Cenários simples (5.2), que consistem em conjuntos reduzidos de regras comportamentais e parâmetros, exemplificam o uso de programação genética para modificação de regras preexistentes e geração de novas regras candidatas.

6 CONCLUSÃO

Nesta Tese foram propostos um processo para adaptação autônoma de modelos comportamentais baseados em regras de um simulador operacional de satélites artificiais, bem como uma arquitetura de um sistema de simulação que apoia a execução do processo, dotando assim os modelos comportamentais com características autoadaptativas.

A capacidade de autoadaptação de modelos comportamentais de simuladores de satélites é uma característica extremamente desejável, uma vez que isto elimina a necessidade de ajustes manuais no software. Tradicionalmente, estas adaptações são realizadas de forma manual, possivelmente por meio de edição de bases de dados ou até mesmo a reescrita de códigos-fontes de aplicativos. Isto leva à necessidade de alocação de recursos humanos, financeiros e de tempo para sua execução.

A arquitetura proposta contém um simulador de satélite com um modelo comportamental baseado em regras, cujos resultados na forma de parâmetros simulados são comparados a telemetrias de referência dos dados históricos de missão. Um componente reconfigurador de modelo identifica discrepâncias de valores para o levantamento de parâmetros afetados e analisa a correlação entre parâmetros e regras. Novos valores de parâmetros candidatos e expressões de regras candidatas são gerados e testados de forma iterativa, até identificar-se as alterações apropriadas.

Foi proposta uma heurística para avaliar a técnica de programação genética para atualização das regras que compõem os modelos. A demonstração dos resultados foi apresentada em vários cenários com características distintas de alterações. Para contornar as limitações da programação genética para a obtenção de soluções na determinação de valores alternativos de parâmetros, uma abordagem foi proposta para a análise de correlação de parâmetros e regras. Esta abordagem reduziu o espaço de busca da programação genética e permitiu a obtenção mais rapidamente de valores de parâmetros.

A capacidade de autoadaptação de modelos comportamentais de um simulador de satélites com uso de programação genética e análise de correlação, como

proposto nesta Tese, representa uma inovação. Isto porque a aplicação de técnicas de algoritmos evolutivos como a programação genética mostrou-se inexplorada para a finalidade de adaptação autônoma de modelos comportamentais de simuladores de satélites baseados em regras proposta por este trabalho. O motivo disto pode estar relacionado ao desencorajamento de uso de ferramentas de algoritmos evolutivos a sistemas complexos, conforme textos encontrados na literatura. Outras possibilidades que poderiam estar ligadas à escassez de referências bibliográficas na academia, seriam, por um lado a necessidade de sigilo tecnológico guardado pela indústria de software, responsável pelo desenvolvimento comercial de simuladores e, por outro lado, a complexidade para construção de um simulador de satélite, que requer grande esforço para construção de partes sem inovação. Esta pesquisa contou com o conhecimento prévio do autor em operação de satélites, em especificação de modelos comportamentais de simuladores de satélite e no desenvolvimento de um simulador simplificado.

Dado o princípio de construção dos modelos comportamentais do SIMCBERS e seu reuso em outros simuladores de satélites, podemos afirmar que os resultados desta Tese são diretamente aplicáveis aos futuros simuladores de satélites baseados neste simulador. Os resultados obtidos indicam que um componente Reconfigurador pode ser elaborado a partir da expansão do protótipo desenvolvido nesta Tese para beneficiar outros projetos do INPE, como o simulador operacional para o satélite Amazonia-1 (SISAO) e os futuros simuladores, para dotar modelos comportamentais baseados em regras de capacidade autoadaptativa.

Uma contribuição adicional desta Tese está relacionada à área de inteligência emergente, não necessariamente confinada à área de simulação de satélites. Na inteligência emergente, o software descobre soluções dinamicamente sem necessariamente obter informações prévias detalhadas do sistema, de tal forma que o conhecimento “emerge” como subproduto da busca por soluções.

A pesquisa de trabalhos relacionados indicou que, sob o ponto de vista da engenharia de software, normalmente a adaptação acontece na fase de projeto

pela utilização de AOM como padrão de projeto. O modelo comportamental é concebido prevendo sua adaptação a posteriori, por meio de metaparâmetros criados especificamente para este fim. Na arquitetura original proposta nesta Tese, provê-se capacidade autoadaptativa a um modelo comportamental não elaborado com o conceito de AOM. O modelo comportamental é adaptado dinamicamente à medida que discrepâncias entre a saída do simulador e telemetrias de referência são identificadas por comparação, na fase operacional de execução do simulador à medida que o satélite envelhece. Isto inclusive permite autoadaptação a modos de falha não previstos em FDIR.

Esta Tese utilizou uma abordagem híbrida para a aplicação de programação genética. Soluções para as novas regras podem ser obtidas por meio ou de manipulação da regra como um todo (regras candidatas) ou de alteração apenas de valores de parâmetros (parâmetros candidatos), sendo a programação genética aplicada apenas para a obtenção de regras.

A prova de conceito por meio de cenários simples aponta para categorias de problemas para os quais a programação genética encontra soluções. Para as demais categorias, parâmetros correlacionados são buscados a partir da análise de correlação de regras e parâmetros do modelo comportamental. Esta análise permitiu reduzir o escopo de aplicação da programação genética, poupando esforço computacional.

Falhas intermitentes são tratadas como uma sucessão rápida de novos comportamentos, em que o modelo se adapta após cada identificação de discrepância. No protótipo implementado, o reconfigurador de modelo tenta obter um novo modelo estático a cada caso. Uma questão que se coloca sobre a implementação do reconfigurador é que, se ele for testar a validade de modelos anteriores a cada discrepância identificada, isso levaria à elaboração de um modelo com comportamento intermitente. Neste caso, a escolha do comportamento poderia se dar por meio de variável aleatória cujo valor poderia ser obtido estatisticamente.

Uma limitação desta Tese está na impossibilidade de afirmar que a programação genética é a melhor técnica para a adaptação autônoma de

modelos comportamentais baseados em regras, pois outras técnicas alternativas não foram exploradas. Entretanto, este trabalho provou que ela é uma ferramenta válida para alcançar este objetivo.

O protótipo desenvolvido, atualmente se limita à adaptação de modelos baseados em regras do simulador simplificado, mas ele pode ser adaptado para outros modelos baseados em regras por meio de implementação dos componentes pré-processadores e pós-processadores específicos para outros simuladores.

Um possível trabalho futuro seria a exploração de outras técnicas de inteligência emergente, como por exemplo, na área de pesquisa relacionada ao conceito de "*machine learning*" aplicado em modelos autoadaptativos em Simuladores Operacionais de Satélites.

O processo e a arquitetura propostos, bem como a heurística por programação genética e a abordagem da análise de correlação são soluções específicas para modelos baseados em regras podendo ser adaptadas a expressões causais de regras. Entretanto, eles têm potencial para se tornarem genéricos para linguagens de programação interpretadas imperativas ao considerarmos que um comando de atribuição é uma regra cuja precondição é verdadeira e comandos de controle podem ser caracterizados por meio de relações causais.

Finalmente, para a operacionalização do protótipo implementado nesta Tese, um próximo passo seria a elaboração de uma versão do reconfigurador de modelo utilizando como base o protótipo e implementação dos módulos de pré-processamento e pós-processamento definidos na arquitetura proposta. Desta forma, compatibilizando o reconfigurador com os modelos comportamentais dos simuladores operacionais como o SIMCBERS, da família de satélites CBERS e o SISAO, da família de satélites da Plataforma Multi-missão (PMM), cujo primeiro satélite é o Amazônia-1.

REFERÊNCIAS BIBLIOGRÁFICAS

ALTENBERG, L. The evolution of evolvability in genetic programming. **Advances in Genetic Programming**, v. 3, p. 47-74, 1994.

AMBROSIO, A. M.; CARDOSO, P. E.; ORLANDO, V.; BIANCHI NETO, J. Brazilian satellite simulators previous solutions trade-off and new perspectives for the CBERS program. In: CONFERENCE ON SPACE OPERATIONS (SpaceOps 2006), 8.,2006. **Proceedings…** Rome: IAAA, 2006.

AZEVEDO, D. R.; HOFFMANN, L. T.; AMBROSIO, A. M.; PERONDI, L. F. Analysis of the simulation model platform adoption in the context of INPE simulators. In: WORKSHOP ON SIMULATION FOR EUROPEAN SPACE PROGRAMMES (SESP), 2012, Noordwijk, Netherlands. **Proceedings…** Noordwijk: ESTEC/ESA, 2012.

BACA, M. (Ed.). **Introduction to metadata**. [S.l.]: Getty Publications, 2016.

BARRETO, J.P.; HOFFMANN, L.T.; AMBROSIO, A.M. Using SMP2 standard in operational and analytical simulators. In: CONFERENCE ON SPACE OPERATIONS, 10., 2010, Huntsville, Alabama, USA. **Proceedings…** AIAA, 2010.

BITTNER, B.; BOZZANO, M.; CIMATTI, A.; DE FERLUC, R.; GARIO, M.; GUIOTTO, A.;YUSHTEIN, Y. An integrated process for FDIR design in aerospace. In: ORTMEIER, F.; RAUZY, A. (Ed.). **Model-based safety and assessment**. Berlin: Springer, 2014. p. 82-95.

BRANDT, A.; KOSSEV, I.; FALKE, A.; EICKHOFF, J.; RÖSER, H-P. Preliminary system simulation environment of the university micro-satellite flying laptop. In: SANDAU, R.; ROSER, H. P.; VALENZUELA, A. (Ed.). **Small satellites for Earth observation**. Berlin: Springer, 2008. p. 231-243.

BRAZILIAN MULTI-MISSION PLATFORM (MMP) AMAZONIA-1. **AMAZONIA-1 satellite operation FDIR.** São José dos Campos: [s.n.], 2017. A820000-SPC-017/02.

BRAZILIAN MULTI-MISSION PLATFORM (MMP) AMAZONIA-1. **Satellite control system design definition.** São José dos Campos: [s.n.], 2019. A832000-SPC-001/01.

CALLAI, T. C; COELHO, L. S.; COELHO, A. A. R. Controle nebuloso adaptativo por modelo de referência: projeto e aplicação em sistemas não lineares. **Controle & Automação**, v. 18, n. 4, p. 479-489, 2007.

CATASTINI, G.; CESARE, S.; DUMONTEL, M.; PARISCH, M.; SECHI, G. FLOBERGHAGEN, R. The GOCE end-to-end system simulator. In: ESA LIVING PLANET, 2010. **Proceedings…** Disponível em: https://earth.esa.int/download/goce/livingplanet2010/007-D3-Poster-the-End-to-End-System-Simulator-of-GOCE.pdf. Acesso em: 26 maio 2015.

CHIANG, C. A genetic programming based rule generation approach for intelligent control systems, **2010 International Symposium on Computer, Communication, Control and Automation (3CA)**, Tainan, 2010, pp. 104-107, doi: 10.1109/3CA.2010.5533882.

CHINA-BRAZIL EARTH RESOURCES SATELLITE (CBERS). **Requisitos do software simulador CBERS3**: treinamento de operadores: RT-SRS-1002. São José dos Campos: INPE, 2006.

CHINA-BRAZIL EARTH RESOURCES SATELLITE (CBERS). **CBERS-3&4 simulator models technical specification guidelines:** RT-SRS-1022. São José dos Campos: INPE, 2012.

CHINA-BRAZIL EARTH RESOURCES SATELLITE (CBERS). **CBERS-4 contingency plan**: RB-MOM-4516. Xian: [s.n.], 2014a.

CHINA-BRAZIL EARTH RESOURCES SATELLITE (CBERS). **CBERS 3&4 SEM model technical specification**: RTD-SRS-1017. São José dos Campos: INPE, 2014b.

CHINA-BRAZIL EARTH RESOURCES SATELLITE (CBERS). **Plano para o comissionamento do satélite CBERS-4**: R-MNG-1226. São José dos Campos: INPE, 2015.

CHINA-BRAZIL EARTH RESOURCES SATELLITE (CBERS). **EPSS telemetry**: RB-OAR-1007(F4). São José dos Campos: INPE, 2016a.

CHINA-BRAZIL EARTH RESOURCES SATELLITE (CBERS). **DDR playback switch status**: RB-OAR-1030(F4). São José dos Campos: INPE, 2016b.

CHINA-BRAZIL EARTH RESOURCES SATELLITE (CBERS). **CBERS 3&4 satellite simulator system requirements specification**: RT-SRS-1021/01. São José dos Campos, Brasil: INPE, 2016c.

CHINA-BRAZIL EARTH RESOURCES SATELLITE (CBERS). **Satellite model requirement specification for CBERS-4 satellite simulator**: RT-SRS-1025/00. São José dos Campos, Brasil: INPE, 2016d.

CHINA-BRAZIL EARTH RESOURCES SATELLITE (CBERS). **CBERS-4 DCS model technical specification AS-BUILT**: RTD-SRS-1006. São José dos Campos: INPE, 2016e.

CHINA-BRAZIL EARTH RESOURCES SATELLITE (CBERS). **CBERS 3&4 AOCS model technical specification AS-BUILT**: RTD-SRS-1008. São José dos Campos: INPE, 2016f.

CHINA-BRAZIL EARTH RESOURCES SATELLITE (CBERS). **CBERS 3&4 EPSS model technical specification AS-BUILT**: RTD-SRS-1009. São José dos Campos: INPE, 2016g.

CHINA-BRAZIL EARTH RESOURCES SATELLITE (CBERS). **CBERS-4 TTCS model technical specification AS-BUILT**: RTD-SRS-1010. São José dos Campos: INPE, 2016h.

CHINA-BRAZIL EARTH RESOURCES SATELLITE (CBERS). **CBERS 3&4 TCSS - RBBA model technical specification AS-BUILT**: RTD-SRS-1007. São José dos Campos: INPE, 2017.

CHINA-BRAZIL EARTH RESOURCES SATELLITE (CBERS). **CBERS-4 OBDH model technical specification AS-BUILT**: RTD-SRS-1011. São José dos Campos: INPE, 2017a.

CHINA-BRAZIL EARTH RESOURCES SATELLITE (CBERS). **CBERS-4 SYSC model technical specification AS-BUILT**: RTD-SRS-1012. São José dos Campos: INPE, 2017b.

CHINA-BRAZIL EARTH RESOURCES SATELLITE (CBERS). **CBERS-4 PAN model technical specification AS-BUILT**: RTD-SRS-1013. São José dos Campos: INPE, 2017c.

CHINA-BRAZIL EARTH RESOURCES SATELLITE (CBERS). **CBERS-3&4 IRS model technical specification AS-BUILT**: RTD-SRS-1014. São José dos Campos: INPE, 2017d.

CHINA-BRAZIL EARTH RESOURCES SATELLITE (CBERS). **CBERS-4 PIT model technical specification AS-BUILT**: RTD-SRS-1015. São José dos Campos: INPE, 2017e.

CHINA-BRAZIL EARTH RESOURCES SATELLITE (CBERS). **CBERS-3&4 MWT model technical specification AS-BUILT**: RTD-SRS-1016. São José dos Campos: INPE, 2017f.

CHINA-BRAZIL EARTH RESOURCES SATELLITE (CBERS). **CBERS 3&4 MUX model technical specification AS-BUILT**: RTD-SRS-1018. São José dos Campos: INPE, 2017g.

CHINA-BRAZIL EARTH RESOURCES SATELLITE (CBERS). **CBERS-4 DDR model technical specification AS-BUILT**: RTD-SRS-1019. São José dos Campos: INPE, 2017h.

CHINA-BRAZIL EARTH RESOURCES SATELLITE (CBERS). **CBERS-4 WFI model technical specification AS-BUILT**: RTD-SRS-1020. São José dos Campos: INPE, 2017i.

DIAS, A.; GUERRA, E.; LIMA, P. An architecture for dynamic web services that integrates adaptive object models with existing frameworks. In: BRAZILIAN SYMPOSIUM ON SOFTWARE COMPONENTS, ARCHITECTURES AND REUSE, 13., 2019. **Proceedings…** 2019. p. 13-22.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **ECSS-E-ST-70-11C**:space engineering: space segment operability.Netherlands, July 2008.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **ECSS-Q-ST-30-02C**: space product assurance: Failure Modes, Effects and Criticality Analysis (FMEA/FMECA). Netherlands, Mar. 2009a.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **ECSS-Q-ST-30C**: space product assurance: dependability. Netherlands, Mar. 2009.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **ECSS-E-TM-10-21A**: space engineering: system modelling and simulation. Netherlands, Apr. 2010.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **ECSS-E-TM-40-07 1A**: space engineering: simulation modelling platform - volume 1: principles and requirements. Netherlands, Jan. 2011a.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **ECSS-E-TM-40-07 2A**: space engineering: simulation modelling platform - volume 2: metamodel.. Netherlands, Jan. 2011b.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **ECSS-E-TM-40-07 3A**: space engineering: simulation modelling platform - volume 2: component model.. Netherlands, Jan. 2011.

EICKHOFF, J. **Simulating spacecraft systems**. Berlin: Springer-Verlag, 2009.

EICKHOFF, J.; FALKE, A.; RÖSER, H.-P. Model-based design and verification: state of the art from Galileo constellation down to small university satellites. **Acta Astronautica**, v. 61, n. 1-6, p. 383-390, 2007.

FERREIRA, H. S.; CORREIA, F. F.; WELICKI, L. Patterns for data and metadata evolution in adaptive object-models. In: CONFERENCE ON PATTERN LANGUAGES OF PROGRAMS, 15., 2008. **Proceedings…** 2008. p. 1-9.

FOGEL, D.B. Foundations of evolutionary computation. In: MODELLING AND SIMULATION FOR MILITARY APPLICATIONS, 2006. **Proceedings…** SPIE, 2006. p. 1-13.

FRITZ, M.; FALKE, A.; KUWAHARA, T.; RÖSER, H.-P.; PEARSON, S.; WITTS, A.; EICKHOFF, J. A commercial procedure execution engine completing the command chain of a university satellite simulation infrastructure. **Acta Astronautica**, v. 66, n. 5-6, p. 950-953, 2010.

GANDOMI, A. H.; ALAVI, A. H. Multi-stage genetic programming: a new strategy to nonlinear system modeling. **Information Sciences**, v. 181, p. 5227–5239, 2011.

GUIOTTO, A.; MARTELLI, A.;PACCAGNINI, C. SMART-FDIR: use of artificial intelligence in the implementation of a satellite FDIR. **DASIA**, v.532, p.71, 2003.

HADKA, D; REED, P. Diagnostic assessment of search controls and failure modes in many-objective evolutionary optimization. **Evolutionary Computation**, v. 20, n. 3, p. 423-452, 2012. DOI:10.1162/EVCO_a_00053.

HADKA, D.; REED, P. Borg: an auto-adaptive many-objective evolutionary computing framework. **Evolutionary Computation**, v. 21, n. 2, p. 231-259, 2013. DOI:10.1162/EVCO_a_00075.

HENDRICKS, R.; EICKHOFF, J. The significant role of simulation in satellite development and verification. **Aerospace Science and Technology**, v. 9, n. 3, p. 273-283, 2005.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS. **Proposta de um simulador para o satélite SCD1, OPR-RE-001**. São José dos Campos: INPE, 1990. 81 p. INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (INPE). **Elkep_hHistory.dat**. São José dos Campos: INPE, 2016.

KANG, J.-Y.; KIM, J.-M.; CHUNG, S.-J. Design and development of an advanced real-time satellite simulator. **Etri Journal**, v. 17, n. 3, p. 1-16, 1995.

KOZA, J. R. **Genetic programming**: on the programming of computers by means of natural selection. [S.l.]: MIT Press, 1992.

KOZA, J. R. Genetic programming as a means for programming computers by natural selection. **Statistics and Computing**, v. 4, n. 2, p. 87-112, 1994.

KRUPITZER, C.; ROTH, F. M.; VANSYCKEL, S.; SCHIELE, G.;BECKER, C. A survey on engineering approaches for self-adaptive systems. **Pervasive and Mobile Computing**, v. 17, p. 184-206, 2015.

LEE, S.-U.; KIM, J.-H.; LEE, H.-J.; LEE, S.-P. **Hybrid type satellite simulation system and method thereof**. U.S. Patent Application n. 11/182,133, 11 maio 2006.

LEE, S.-U.; JUNG, O.-C.; JEONG, S.-K.; KIM, J-H. **Satellite simulation system using component-based satellite modeling**. U.S. Patent Application n. 11/437,612, 7 jun. 2007.

LIAO, S.-H. Expert system methodologies and applications - a decade review from 1995 to 2004. **Expert Systems with Applications**, v. 28, p. 93-103, 2005.

MACÍAS-ESCRIVÁ, F. D.; HABER, R.; DEL TORO, R.; HERNANDEZ, V. Self-adaptive systems: a survey of current approaches, research challenges and applications. **Expert Systems with Applications**, v. 40, n. 18, p. 7267-7279, 2013.

MALDAGUE, P. F.; WISSLER, S. S. Towards a cradle-to-grave, mission-wide simulation system. In: SPACEOPS CONFERENCE, 2018. **Proceedings…** Marseille: IAAA, 2018.

MISSÃO ESPACIAL COMPLETA BRASILEIRA (MECB). **SCD2 Operation Handbook:** A-MIN-0085. São José dos Campos: [s.n.], 1993.

MERRIAM-WEBSTER. **Metadata**. Disponível em: https://www.merriam-webster.com/dictionary/metadata. Acesso em: 22 Maio 2020.

POLI, R.; LANGDON, W.B.; MCPHEE, N. F.; KOZA, J. R. **Genetic programming**: an introductory tutorial and a survey of techniques and applications. [S.l.]: CES, 2007.

REVAULT, N.; YODER, J. W. Adaptive object-models and metamodeling techniques. In: EUROPEAN CONFERENCE ON OBJECT-ORIENTED PROGRAMMING, 2001. **Proceedings…** Springer,2001. p. 57-71.

SEBASTIÃO, N.; REGGESTAD, V.; SPADA, M.; WILLIAMS, A.; PECCHIOLI, M.; LINDMAN, N.; FRITZEN, P. A reference architecture for spacecraft simulators. In: SPACEOPS 2008 CONFERENCE, 2008, Heidelberg, Germany. **Proceedings...** [S.l.]: AIAA, 2008.

TOMINAGA, J. **Simulador de satélites para verificação de planos de operações em voo**. 2010. 174 p. Dissertação (Mestrado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2010.

TOMINAGA, J.; CERQUEIRA, C. S.; KONO, J.; AMBROSIO, A. M. Specifying satellite behavior for an operational simulator. In: SIMULATION AND EGSE FACILITIES FOR SPACE PROGRAMMES WORKSHOP (SESP), 2012. **Proceedings…** Noordwijk: ESA, 2012.

TOMINAGA, J.; FERREIRA, M. G. V.; AMBROSIO, A. M. Comparing satellite telemetry against simulation parameters in a simulator model reconfiguration. In: WORKSHOP EM ENGENHARIA E TECNOLOGIAS ESPACIAIS (WETE), 8., 2017. **Anais...** São José dos Campos: INPE, 2017.

TOMINAGA, J.; FERREIRA, M. G. V.; AMBROSIO, A. M. Updating simulation parameters in a simulator model reconfiguration tool. In: WORKSHOP EM ENGENHARIA E TECNOLOGIAS ESPACIAIS (WETE), 9., 2018. **Anais...** São José dos Campos: INPE, Brazil.

WANG, J.; LEE, C. S. G. Self-adaptive neuro-fuzzy inference systems for classification applications. **IEEE Transactions on Fuzzy Systems**, v. 10, n. 6, Dec. 2002.

WELICKI, L; YODER, J. W.; WIRFS-BROCK, R. Adaptive object-model builder. In: CONFERENCE ON PATTERN LANGUAGES OF PROGRAMS, 16., 2009. **Proceedings…** 2009. p. 1-8.

WERTZ, J. R.; EVERETT, D. F.; PUSCHELL J. J. **Space mission engineering**: the new SMAD". [S.l.]: Microcosm, 2011. ISBN 978-1-881883-15-9.

XIAN SATELLITE CONTROL CENTER (XSCC). **CBERS FM4 handbook**. Xian: [s.n.], 2014.

XIAN SATELLITE CONTROL CENTER (XSCC). **Orbit_2016-01-08.doc**. Xian: [s.n.], 2016.

XIAN SATELLITE CONTROL CENTER (XSCC); INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (INPE). **CBERS simulator software design documents**. São José dos Campos: INPE, 2000.

ZHANG, Y; ROCKETT, P. I. A generic optimising feature extraction method using multiobjective genetic programming. **Applied Soft Computing**, v. 11, n. 1, p. 1087-1097, 2011.

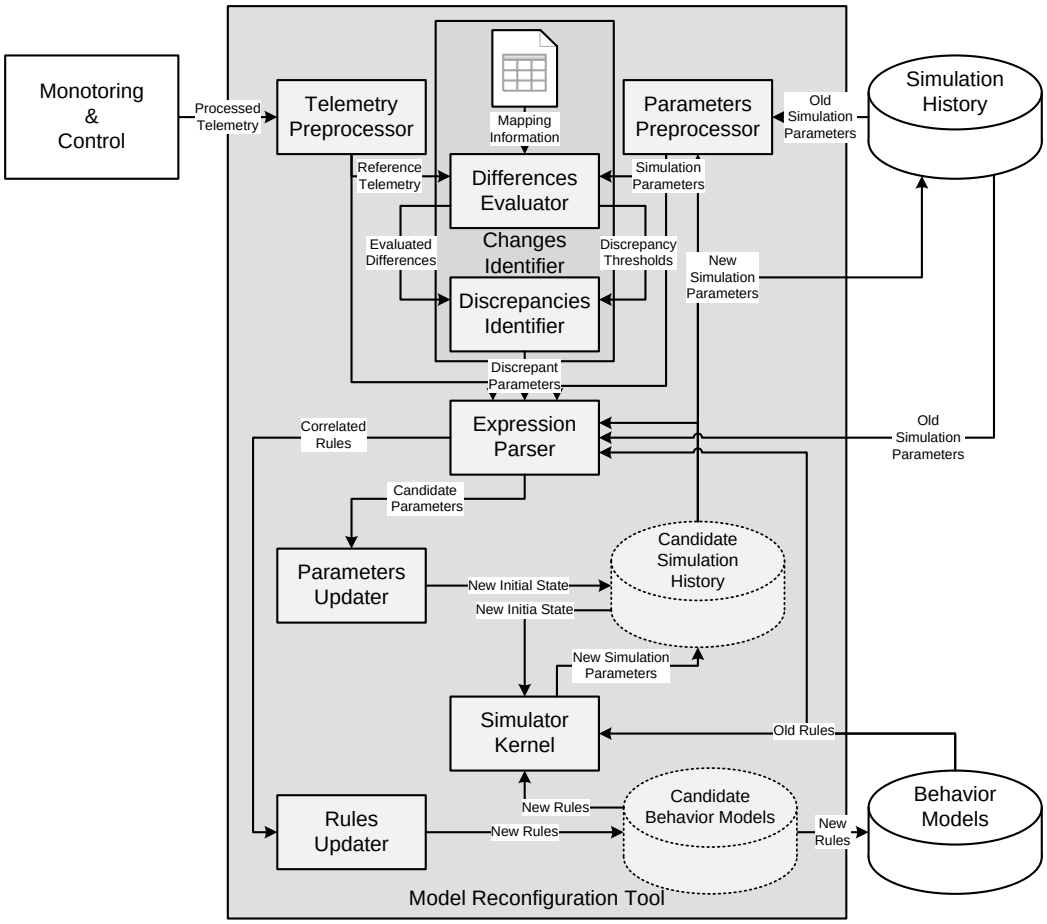
ZHOU, A.; QU, B.; LI, H.; ZHAO, S, Z.; SUGANTHAN, P. N.; ZHANG, Q. Multiobjective evolutionary algorithms: a survey of the state of the art. **Swarm and Evolutionary Computation**, v.1, p. 32–49, 2011.

APÊNDICE A – PROTÓTIPO DO RECONFIGURADOR DE MODELO

A.1 Implementação do Protótipo do Reconfigurador de Modelo

A Figura A.1 ilustra a arquitetura do protótipo do Reconfigurador de Modelo com detalhes de sua implementação, sendo funcionalmente equivalente a arquitetura apresentada no Capítulo 4..

Figura A.1. Arquitetura de implementação do protótipo de Reconfigurador de Modelo.



Fonte: Adaptado de Tominaga et al. (2018).

A.2 Composição do protótipo do Reconfigurador de Modelo

O protótipo do Reconfigurador de Modelo (MODEL_RECONFIG) é composto pelas sub-rotinas e funções a seguir, distribuídas em vários arquivos de código-fonte.

- model_reconfig.py - model_reconfig
- telemetry_preprocessor.py - telemetry_preprocessor
- parameters_preprocessor.py - parameters_preprocessor
- differences_evaluator.py - differences_evaluator
- discrepancies_identifier.py - discrepancies_identifier
- print_telemetry_discrepancies.py - print_telemetry_discrepancies
- expression_parser.py - expression_parser
- find_minimum_interval.py - find_minimum_interval
- find_candidate_parameters.py - find_candidate_parameters
- initial_telemetry_parameters.py - initial_telemetry_parameters
- correlated_rules_parameters.py - correlated_rules_parameters
- search_rules.py - search_rules
- expr_str.py - expr_str
- list_str.py - list_str
- xml_expr - xml_expr
- rules_parameters.py - rules_parameters
- new_candidate_parameters.py - new_candidate_parameters

- filter_telemetry.py - filter_telemetry
- filter_control.py - filter_control
- next_parameters.py - next_parameters
- merge_parameters.py - merge_parameters
- cost_weight.py - cost_weight
- sum_squares.py - sum_squares
- lib_files.py - read_csv, read_map, get_sim_xml, read_xml
- genetic_programming.py
- satsim_subroutine.py
- proc_model_files.py

A.3 Funções do Protótipo do Reconfigurador

As funções e sub-rotinas componentes do protótipo são apresentadas a seguir.

Sub-rotina principal (model_reconfig)

Esta sub-rotina é invocada pelo programa principal e controla o fluxo principal de execução das demais sub-rotinas.

Função de pré-processamento de telemetrias (telemetry_preprocessor)

Esta função é chamada pela sub-rotina **model_reconfig** para adquirir telemetrias processadas exportadas em formato CSV e retorna uma lista de parâmetros de telemetria de referência. Ela chama a função de leitura de valores de arquivo CSV (**read_csv**).

Função de pré-processamento de parâmetros de simulação (parameters_preprocessor)

Esta função é chamada pela sub-rotina **model_reconfig** para adquirir parâmetros de simulação do histórico de simulação gerado em uma execução

anterior do Simulador de Satélite utilizando um modelo comportamental antigo. Estes parâmetros de simulação antiga exportadas pelo Simulador de Satélite em formato CSV são lidos e retornados como uma lista de parâmetros de simulação de saída. Uma lista de parâmetros de simulação com telemetrias correspondentes é obtida com o auxílio de um arquivo CSV de mapeamento de telemetrias, de onde também é obtida uma lista de limites de erro entre parâmetros de simulação e telemetrias correspondentes para caracterização de discrepância.

Ela chama a função de leitura de valores de arquivo CSV (**read_csv**) e a função de leitura de informações de mapeamento entre telemetrias e parâmetros de simulação correspondentes em CSV (**read_map**).

Função de cálculo de diferenças (differences_evaluator)

Esta função é chamada pela sub-rotina **model_reconfig** para calcular as diferenças entre valores da lista de telemetrias de referência e da lista de parâmetros de simulação com telemetrias correspondentes, retornando uma lista de diferenças de telemetria.

Função de identificação de discrepâncias (discrepancies_identifier)

Esta função é chamada pela sub-rotina **model_reconfig** para identificar discrepâncias por meio da comparação de valores da lista de diferenças de telemetria contra a lista de limites de erro, retornando uma lista de discrepâncias.

Sub-rotina de exibição de discrepâncias de telemetrias (print_telemetry_discrepancies)

Esta sub-rotina é invocada pela sub-rotina **model_reconfig** e exibe na tela o resultado da identificação de discrepâncias.

Função de interpretação de expressões (expression_parser)

Esta função é chamada pela sub-rotina **model_reconfig** para decodificar expressões de regras comportamentais do Simulador de Satélite em arquivo XML.

Provisoriamente ela está implementada como sub-rotina, ou função sem retorno, e concentra grande parte das funcionalidades implementadas por meio de funções e sub-rotinas previamente desenvolvidas para o primeiro protótipo XML_EVO após a identificação de discrepâncias, sendo utilizada para testes funcionais e de integração dos componentes reescritos.

Ela chama a função de procura de intervalo mínimo de busca (**find_minimum_interval**), a função de leitura de arquivos XML do simulador (**get_sim_xml**) e a função de procura de parâmetros candidatos (**find_candidate_parameters**), para obter uma lista de parâmetros candidatos.

Atualmente ela chama a função de atribuição de pesos a custos (**cost_weight**) e a função de cálculo de função custo (**sum_squares**) para inicialização do enlace de execução do Simulador de Satélite com arquivos XML de estado inicial com parâmetros candidatos.

Função de procura de intervalo mínimo de busca (find_minimum_interval)

Esta função é chamada pela função **expression_parser** para calcular os tempos de início e fim de simulação dentro do qual foi registrada discrepância entre parâmetros de simulação e telemetrias, conforme a função **minimum_interval**.

O trecho de código responsável pelo erro na função **minimum_interval** no cálculo do tempo final não foi transportada para esta função, mas ainda não foi implementada uma solução, de forma que esta função por enquanto utiliza o tempo final da simulação original.

Função de leitura de arquivos XML do simulador (get_sim_xml)

Esta função é chamada pela função **expression_parser** e realiza a leitura de arquivos de estado inicial (XML), fila de eventos (XML) e regras comportamentais (XML), retornando estruturas de árvores XML correspondentes.

Função de leitura de árvore XML (read_xml)

Esta função é chamada pela função **get_sim_xml** e realiza a leitura de um arquivo XML, retornando a estrutura de árvore XML correspondente.

Função de procura de parâmetros candidatos (find_candidate_parameters)

Esta função é chamada pela função **expression_parser** para buscar uma lista de parâmetros candidatos a partir das listas de parâmetros de simulação com telemetrias correspondentes, das regras de controle, do mapeamento de telemetrias e das discrepâncias encontradas, inicializando a lista de parâmetros candidatos por meio da função **initial_telemetry_parameters** e executando as funções **correlated_rules_parameters**, **new_candidate_parameters** e **merge_parameters** iterativamente.

Função de inicialização de parâmetros de telemetria (initial_telemetry_parameters)

Esta função é chamada pela função **find_candidate_parameters** para inicializar a lista de parâmetros candidatos a partir da lista de discrepâncias de telemetrias.

Função de parâmetros de regras correlacionadas (correlated_rules_parameters)

Esta função é chamada pela função **find_candidate_parameters** para obter uma a lista de parâmetros candidatos a partir das listas de parâmetros de telemetrias, das regras de controle e de parâmetros candidatos previamente identificados.

Função de novos parâmetros candidatos (new_candidate_parameters)

Esta função é chamada pela função **find_candidate_parameters** para obter uma a lista de parâmetros candidatos a partir das listas de parâmetros de regras correlacionadas, do mapeamento de telemetrias e de parâmetros candidatos previamente identificados.

Função de filtragem de parâmetros de telemetria (filter_telemetry)

Esta função é chamada pela função **find_candidate_parameters** para filtrar parâmetros de telemetrias não discrepantes da lista de parâmetros candidatos.

Função de filtragem de parâmetros de controle (filter_control)

Esta função é chamada pela função **find_candidate_parameters** para filtrar parâmetros de controle de simulação da lista de parâmetros candidatos.

Função de avanço de busca de parâmetros candidatos (next_parameters)

Esta função é chamada pela função **find_candidate_parameters** para criar uma nova lista de parâmetros candidatos e avançar o passo do enlace de busca de parâmetros candidatos.

Função de mescla de parâmetros (merge_parameters)

Esta função é chamada pelas funções **find_candidate_parameters** e **correlated_rules_parameters** para unir a lista de parâmetros candidatos encontrado no passo de busca anterior com a uma lista de novos parâmetros candidatos encontrado no novo passo de busca.

Função de busca de regras por parâmetro (search_rules)

Esta função é chamada pela função **correlated_rules_parameters** para obter uma lista de regras de controle que contenham em suas expressões de precondição ou de efeito o identificador de um parâmetro.

Função de conversão de XML para expressão (xml_expr)

Esta função é chamada pela função **search_rules** para obter uma lista representativa de expressões a partir de uma estrutura de árvore XML.

Uma expressão pode ser representada por uma lista cujos elementos podem ser parâmetros, valores, operandos ou listas representando expressões aninhadas.

Esta função não está codificada no padrão de atribuição de nomes proposto para o MODEL_RECONFIG.

Função de conversão de expressão para string (expr_str)

Esta função é chamada pelas funções **search_rules** e **rules_parameters** para gerar uma string para exibição de uma regra a partir de suas expressões de precondição e efeito. Esta função não está codificada no padrão de atribuição de nomes proposto para o MODEL_RECONFIG.

Função de conversão de lista para string (list_str)

Esta função é chamada pela função **search_rules** para gerar uma string para exibição de uma lista de elementos de expressão.

Esta função não está codificada no padrão de atribuição de nomes proposto para o MODEL_RECONFIG.

Função de listagem de parâmetros de uma regra (rules_parameters)

Esta função é chamada pela função **correlated_rules_parameters** para obter uma lista de parâmetros presentes expressões de precondição ou de efeito de uma regra de controle.

Função de atribuição de pesos a custos (cost_weight)

Esta função é chamada provisoriamente pela função **expression_parser** para atribuir pesos a parâmetros no processo de cálculo da função custo.

Função de cálculo de soma de quadrados (sum_squares)

Esta função é chamada provisoriamente pela função **expression_parser** para calcular a função custo a ser minimizada durante a iteração de execução do Simulador de Satélite com novos arquivos XML candidatos.

A.4 Módulo de programação genética

Este módulo é constituído pelas funções a seguir.

- programação genética (**genetic_programming**)
- listas de operações (**operations_lists**)
- inicialização de pais (**init_parents**)
- mutação (**mutate**)
- mutação de regra filha (**mutate_child**)
- mutação de elemento (**mutate_element**)
- expressão para elementos (**expression_elements**)
- travessia de expressão (**traverse_expression**)

- elemento aleatório (**random_element**)
- valor de ponto flutuante aleatório (**random_float_value**)
- comparação de expressões (**compare_expressions**)
- substituição de elementos (**elements_replace**)
- árvore de elementos (**elements_tree**)
- elementos excluídos (**deleted_elements**)
- mudar elemento (**change_element**)
- elementos para expressão (**elements_expression**)
- elementos para subexpressão (**elements_subexpression**)
- crossover (**crossover**)
- inicialização de crossover (**init_cross**)
- busca de operações em expressão (**oper_in_expr**)
- avaliação de aptidão (**eval_fit**)
- escrita de regra (**write_rules**)
- soma dos quadrados (**sum_squares**)
- registro de regras (**log_rules**)

Função de programação genética (genetic_programming)

Esta função é chamada pela sub-rotina **model_reconfig** para aplicar o algoritmo de programação genética sobre as listas de regras correlacionadas e de parâmetros de simulação. Ela aplica as funções de listas de operações (**operations_lists**), inicialização de pais (**init_parents**) e um loop de mutação (**mutate**), crossover (**crossover**) e avaliação de aptidão (**eval_fit**) até encontrar a condição de parada.

Função de listas de operações (operations_lists)

Esta função é chamada pela função de programação genética (**genetic_programming**) e categoriza as operações do Simulador de Satélite

conforme o tipo binário ou unário para permitir a aplicação válida de crossover em expressões de regras.

Função de inicialização de pais (init_parents)

Esta função é chamada pela função de programação genética (**genetic_programming**) para a inicialização da primeira geração da população de regras para a aplicação de programação genética, por meio da combinação do conjunto de regras original, um conjunto de regras mutantes (**mutate**) e um conjunto de regras cruzadas (**crossover**).

Função de mutação (mutate)

Esta função é chamada pelas funções de inicialização de pais (**init_parents**) e de programação genética (**genetic_programming**) para a geração aleatória de conjuntos de regras, cada uma contendo uma regra mutante.

Função de mutação de regra filha (mutate_child)

Esta função é chamada pela função de mutação (**mutate**) para a geração de regras filhas, por meio de substituição aleatória de parte de uma de suas regras.

Função de mutação de elemento (mutate_element)

Esta função é chamada pela função de mutação de regra filha (**mutate_child**) para executar a mutação de um elemento de expressão da regra mutante.

Função de expressão para elementos (expression_elements)

Esta função é chamada pela função de mutação de elemento (**mutate_element**) e pela função de busca de operações em expressão (**oper_in_expr**) para a geração de uma lista de elementos que compõem uma expressão.

Função de travessia de expressão (traverse_expression)

Esta função é chamada pela função conversora de expressão para elementos (**expression_elements**) e recursivamente por ela mesma para a geração de uma estrutura composta pelo índice de contagem, uma lista de elementos encontrados e a profundidade de busca da iteração.

Função de elemento aleatório (random_element)

Esta função é chamada pela função de mutação de elemento (**mutate_element**) para a substituição aleatória de um elemento que compõem uma lista de elementos de uma expressão utilizando como referência conjuntos de parâmetros do modelo de simulação, e também é chamada por ela própria recursivamente.

Função de valor de ponto flutuante aleatório (random_float_value)

Esta função é chamada pela função de elemento aleatório (**random_element**) para geração de um valor numérico real aleatório.

Função de comparação de expressões (compare_expressions)

Esta função é chamada pela função de elemento aleatório (**random_element**) para comparar expressões. É utilizada para evitar a eventual troca aleatória de um elemento por ele mesmo.

Função de substituição de elementos (elements_replace)

Esta função é chamada pela função de mutação de elemento (**mutate_element**) para substituir um elemento de uma lista por uma nova expressão de forma recursiva.

Função árvore de elementos (elements_tree)

Esta função é chamada pela função de substituição de elementos (**elements_replace**) para converter uma lista de elementos para uma estrutura em árvore.

Função de elementos excluídos (deleted_elements)

Esta função é chamada pela função de substituição de elementos (**elements_replace**) recursivamente para a geração de lista de elementos excluídos que fazem parte de uma subexpressão excluída.

Função de mudança de elemento (change_element)

Esta função é chamada pela função de substituição de elementos (**elements_replace**) e recursivamente por ela própria para mudar um elemento

alvo em uma lista de elementos utilizando informação de sua árvore correspondente.

Função de elementos para expressão (elements_expression)

Esta função é chamada pela função de mutação de elemento (**mutate_element**) para a geração de uma expressão a partir da lista de elementos que a compõem.

Função de elementos para subexpressão (elements_subexpression)

Esta função é chamada pela função de elementos para expressão (**elements_expression**) e recursivamente por ela própria para a geração de uma subexpressão a partir da lista de elementos que a compõem.

Função de crossover (crossover)

Esta função é chamada pelas funções de inicialização de pais (**init_parents**) e de programação genética (**genetic_programming**) para a geração aleatória de conjuntos de regras, cada uma contendo um par de regras cruzadas. O cruzamento é realizado entre elementos de tipo intercambiáveis, ou seja, operando ou operação unária por operando ou operação unária, ou operação binária por operação binária.

Função de inicialização de crossover (init_cross)

Esta função é chamada pela função de crossover (**crossover**) para criação de estruturas de dados para novas regras a serem cruzadas.

Função de busca de operações em expressão (oper_in_expr)

Esta função é chamada pela função de crossover (**crossover**) para a verificação de operações que fazem parte de uma lista numa expressão e a geração de duas listas de elementos, uma contendo operações que fazem parte da lista de operações e outra contendo elementos que não fazem parte da lista de operações. Esta função é invocada separadamente para verificação de operações binárias e operações unárias.

Função de avaliação de aptidão (eval_fit)

Esta função é chamada pela função de programação genética (**genetic_programming**) para a avaliação de aptidão. Conjuntos de regras pais e filhas são exportados em formato XML para execução do Simulador de Satélite. As saídas CSV são comparadas com a telemetria de referência e uma função custo é calculada por meio da soma dos quadrados das diferenças em cada passo de simulação. São considerados mais aptos os conjuntos de regras com menor função custo, que são escolhidos para compor a próxima geração de regras.

Função de escrita de regra (write_rules)

Esta função é chamada pela função de avaliação de aptidão (**eval_fit**) para a exportação de regras para um formato XML compatível com o Simulador de Satélite.

Função de soma dos quadrados (sum_squares)

Esta função é chamada pela função de avaliação de aptidão (**eval_fit**) para o cálculo da função custo. Ela retorna um valor real obtido por meio da soma dos quadrados das diferenças entre a telemetria de referência e os parâmetros de simulação correspondentes em cada passo de simulação.

Função de registro de regras (log_rules)

Esta função é chamada pela função de avaliação de aptidão (**eval_fit**) para o registro de informações em arquivo texto referentes a regras encontradas durante as iterações de busca do conjunto de regras mais apto, ou seja, de menor função custo.

A.5 Módulo de simulador de satélite: satsim_subroutine

Este módulo é derivado do programa Simulador de Satélite (SATSIM), cuja versão original em Python era constituída por classes multi-threading, em programação orientada a objeto.

Este módulo é constituído pelas funções e sub-rotinas a seguir.

- simulador (**simulator**)

- simulador com regra alternativa (**sim_alt_rule**)
- inicialização (**init**)
- inicialização de configuração (**init_config**)
- inicialização de estado inicial (**init_stat**)
- inicialização da fila de eventos (**init_queue**)
- inicialização das regras de controle (**init_rules**)
- inicialização dos parâmetros de saída (**init_out**)
- inicialização de curvas de função (**init_curv**)
- inicialização de estrutura XML (**init_xml**)
- sessão de simulação (**session**)
- verificação de eventos (**check_event**)
- avaliação de regras (**eval_rules**)
- armazenamento de histórico (**store_hist**)
- exportação de CSV (**export**)
- leitura de XML (**read_xml**)
- formação de expressão (**form_expr**)
- avaliação de expressão (**eval_expr**)
- avaliação de curva (**eval_curv**)

Sub-rotina de simulação (simulator)

Esta sub-rotina executa o Simulador de Satélite de acordo com os arquivos de entrada e saída conforme lidos do arquivo de configuração.

Sub-rotina de simulação com regras alternativas (sim_alt_rule)

Esta sub-rotina executa o Simulador de Satélite parcialmente de acordo com os arquivos de entrada e saída conforme lidos do arquivo de configuração, porém

utiliza arquivos de regras XML e saída CSV especificadas à parte. É invocada pela função de avaliação de aptidão (**eval_fit**) do módulo de programação genética (**genetic_programming**) para a execução de simulação com regras alternativas.

Função de inicialização (init)

Esta função inicializa a sessão de simulação por meio da obtenção de dados dos arquivos de entrada e saída XML conforme lidos do arquivo de configuração. É invocada pela sub-rotina de simulação (**simulator**).

Função de inicialização de configuração (init_config)

Esta função lê e retorna os nomes dos arquivos de entrada e saída XML do arquivo de configuração. É invocada pela função de inicialização (**init**).

Função de inicialização de estado inicial (init_stat)

Esta função lê e retorna dados obtidos do arquivo de entrada XML de estado inicial cujo nome é fornecido no arquivo de configuração. É invocada pela função de inicialização (**init**).

Função de inicialização da fila de eventos (init_queue)

Esta função lê e retorna dados obtidos do arquivo de entrada XML de fila de eventos cujo nome é fornecido no arquivo de configuração. É invocada pela função de inicialização (**init**).

Função de inicialização das regras de controle (init_rules)

Esta função lê e retorna dados obtidos do arquivo de entrada XML de regras de controle cujo nome é fornecido no arquivo de configuração. É invocada pela função de inicialização (**init**).

Função de inicialização dos parâmetros de saída (init_out)

Esta função escreve dados de cabeçalho no arquivo de saída CSV de parâmetros de saída cujo nome é fornecido no arquivo de configuração. É invocada pela função de inicialização (**init**).

Função de inicialização de curvas de função (init_curv)

Esta função lê e retorna dados obtidos do arquivo de entrada XML de curvas de função cujo nome é fornecido no arquivo de configuração. É invocada pela função de inicialização (**init**).

Função de inicialização de estrutura XML (init_xml)

Esta função lê e retorna dados obtidos de um arquivo de entrada XML. É invocada pela função de inicialização de estado inicial (**init_stat**).

Sub-rotina de sessão de simulação (session)

Esta sub-rotina executa a sessão de simulação conforme os dados de entrada e saída obtidos da função de inicialização (**init**). É invocada pela sub-rotina de simulação (**simulator**) e pela sub-rotina de simulação com regras alternativas (**sim_alt_rule**).

Sub-rotina de verificação de eventos (check_event)

Esta sub-rotina executa a verificação de eventos conforme a fila de eventos dos dados de entrada inicializados.É invocada pela sub-rotina de sessão de simulação (**session**) a cada passo.

Função de avaliação de regras (eval_rules)

Esta função executa a avaliação de cada regra contida no conjunto de regras de controle nos dados de entrada inicializados. Atribui valores a parâmetros alvos de efeito conforme a avaliação da expressão de efeito se a expressão de condição resultar em valor verdadeiro, ou pode aplicar valores calculados parcialmente com base no valor corrente e o novo valor em caso de resultado não booleano por meio de lógica nebulosa. Retorna valor verdadeiro se a condição de parada de simulação for satisfeita, ou falso caso contrário.

É invocada pela sub-rotina de sessão de simulação (**session**) a cada passo.

Sub-rotina de armazenamento de histórico (store_hist)

Esta sub-rotina atualiza o histórico de sessão contendo os valores de todos os parâmetros de simulação. É invocada pela sub-rotina de sessão de simulação (**session**) a cada passo.

Sub-rotina de exportação de CSV (export)

Esta sub-rotina salva o histórico de sessão contendo os valores de todos os parâmetros de simulação a cada passo no arquivo de saída CSV.

É invocada pela sub-rotina de sessão de simulação (**session**) ao fim da sessão.

Sub-rotina de leitura de XML (read_xml)

Esta sub-rotina lê elementos componentes de expressões e seus níveis de aninhamento a partir de uma estrutura XML.

É invocada pela função de avaliação de regras (**eval_rules**) e recursivamente por ela mesma.

Função de formação de expressão (form_expr)

Esta função forma a estrutura de expressões que compõem as condições e efeitos das regras de controle. É invocada pela função de avaliação de regras (**eval_rules**).

Função de avaliação de expressão (eval_expr)

Esta função avalia o resultado de uma expressão a partir dos valores de parâmetros que compõem sua estrutura. É invocada pela função de avaliação de regras (**eval_rules**) e recursivamente por ela mesma.

Sub-rotina de avaliação de operação (eval_oper)

Esta sub-rotina avalia o resultado de uma operação de uma expressão a partir dos valores de parâmetros e constantes associados a ela. É invocada pela função de avaliação de expressão (**eval_expr**).

Função de avaliação de curva (eval_curv)

Esta função avalia o resultado de uma curva de função a partir do valor de sua entrada. É invocada pela sub-rotina de avaliação de operação (**eval_oper**).

A.6 Módulo de interface entre simulador de satélite e programação genética - sim_gp

Este módulo é constituído pelas funções a seguir.

- processamento de arquivos de modelo (**proc_model_files**)
- obtenção de regras (**get_rules**)

Função de processamento de arquivos de modelo (proc_model_files)

Esta função executa o pré-processamento dos arquivos de telemetrias de referência (**telemetry_preprocessor**) e de parâmetros de simulação (**parameters_preprocessor**), o cálculo de diferenças (**differences_evaluator**) e a identificação de discrepâncias (**discrepancies_identifier**). É invocada pela função de avaliação de aptidão (**eval_fit**) do módulo de programação genética (**genetic_programming**) para a inicialização de parâmetros antes da execução da sub-rotina simulador com regras alternativas (**sim_alt_rule**).

Função de obtenção de regras (get_rules)

Esta função gera uma lista de regras compatível com a função de programação genética (**genetic_programming**) a partir de uma estrutura XML exportada pela função de leitura de arquivos XML do simulador (**get_sim_xml**).

APÊNDICE B – INTERFACES DO RECONFIGURADOR DE MODELO

B.1 Interfaces com o Sistema de Solo de TT&C (SATCS)

O SATCS é o software de Sistema de Solo de TT&C desenvolvido pelo INPE para a operação de satélites artificiais sob sua responsabilidade.

Entre outras funcionalidades, o SATCS permite o cadastramento de monitoramento de falhas, conforme documentação de FDIR, em função de outros parâmetros, em adição a parâmetros mais básicos como os de telemetrias e telecomandos. (XSCC, 2014) (CBERS, 2014a)

O SATCS oferece a funcionalidade necessária para o cadastramento e o processamento de modelos comportamentais em forma de regras. Porém, nesta Tese esta funcionalidade não será explorada.

O seu aplicativo de monitoração e controle (MCS) possui um modo de reprodução de telemetrias de histórico, a partir do qual é possível a recuperação de telemetrias armazenadas na base de dados histórica (HISDB) e exportar parâmetros em unidades de engenharia tabeladas em formato CSV de acordo com regras de processamento cadastradas em sua base de dados operacional (OPDB).

O log histórico de envio de telecomandos pode ser acessado por meio do módulo de recuperação de histórico (A&R) e salvo também em formato CSV com telecomandos indicados por meio de identificadores. Estas características simplificam a elaboração de um protótipo de ferramenta para adaptação autônoma de modelo de simulador para leitura de arquivos tipo texto com dados separados por vírgulas (CSV), dispensando a necessidade de acesso às bases de dados HISDB para recuperação de dados brutos e OPDB para processamento.

B.2 Telemetrias processadas

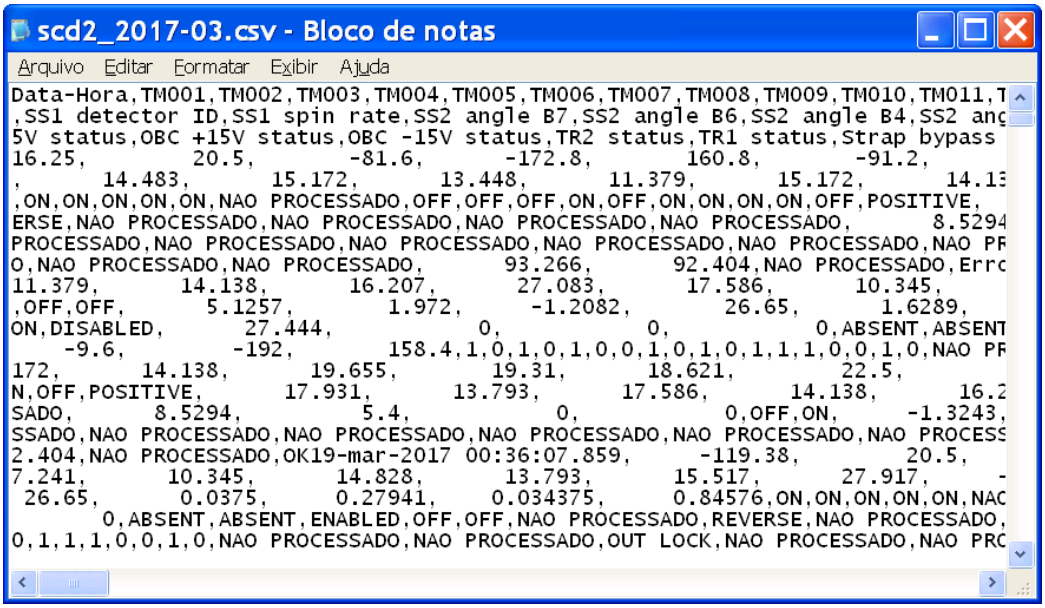
Telemetrias processadas do SATCS podem ser exportadas na forma de arquivo texto em formato CSV, em que parâmetros de telemetrias são arranjados em colunas e valores amostrados são organizados em linhas.

A primeira coluna contém o parâmetro de tempo de bordo (OBTIME), representando a data-hora de amostragem da telemetria em bordo, cujo tempo é exibido em Tempo Universal Coordenado (UTC). As demais colunas contêm parâmetros de telemetrias do satélite cujos valores são exibidos em linhas e atualizados conforme a taxa de amostragem.

As primeiras três linhas contêm informações de cabeçalho. A primeira linha contêm códigos de identificação, a segunda linha um texto descritivo e a terceira linha a unidade associada ao valor, se aplicável, para cada parâmetro de telemetria. As demais linhas contêm valores associados ao tempo de bordo na primeira coluna, podendo conter valores atualizados naquele instante ou valores repetidos da linha anterior, caso não tenha havido atualização de valor naquele tempo.

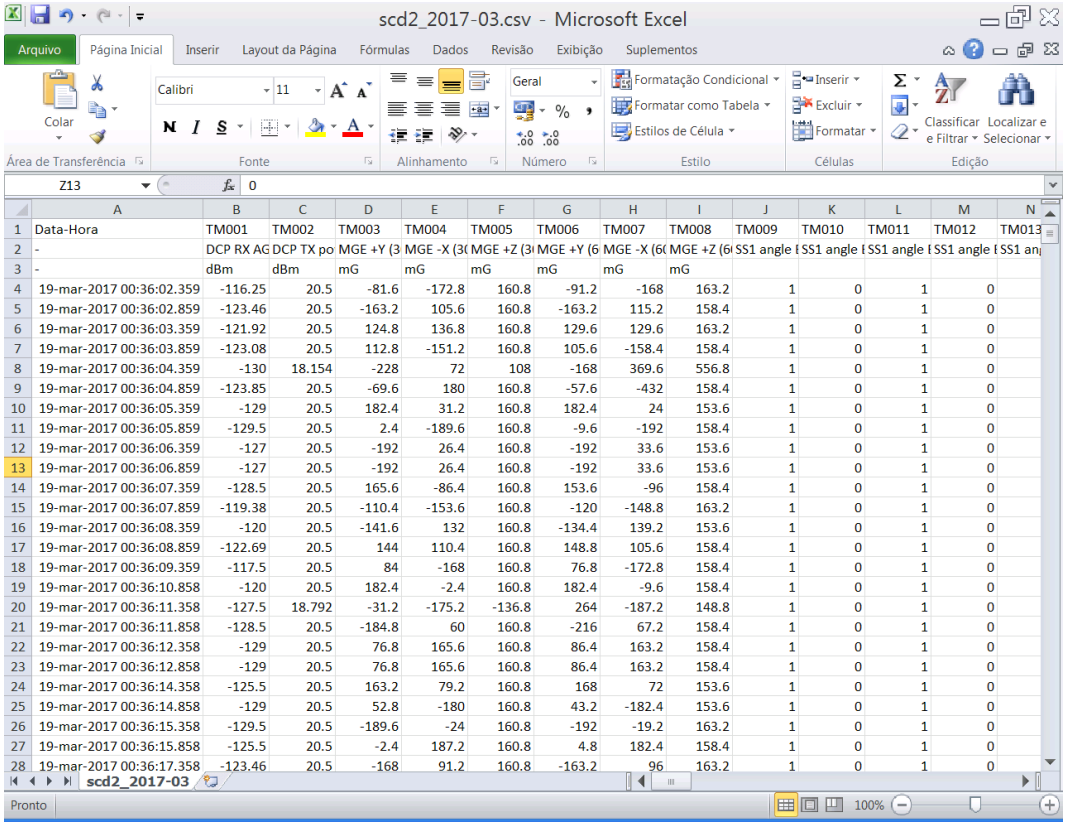
Um exemplo de arquivo CSV de telemetrias processadas é apresentado nas figuras a seguir (Figura B.1 e Figura B.2).

Figura B.1. Arquivo CSV de telemetrias aberto como arquivo texto.



Fonte: Tominaga et al. (2017).

Figura B.2. Arquivo CSV de telemetrias aberto como planilha.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Data-Hora	TM001	TM002	TM003	TM004	TM005	TM006	TM007	TM008	TM009	TM010	TM011	TM012	TM013
2	-	DCP RX AG	DCP TX po	MGE +Y (3	MGE -X (3(	MGE +Z (3(	MGE +Y (6	MGE -X (6(	MGE +Z (6(	SS1 angle	SS1 angle	SS1 angle	SS1 angle	SS1 an
3	-	dBm	dBm	mG	mG	mG	mG	mG	mG					
4	19-mar-2017 00:36:02.359	-116.25	20.5	-81.6	-172.8	160.8	-91.2	-168	163.2	1	0	1	0	
5	19-mar-2017 00:36:02.859	-123.46	20.5	-163.2	105.6	160.8	-163.2	115.2	158.4	1	0	1	0	
6	19-mar-2017 00:36:03.359	-121.92	20.5	124.8	136.8	160.8	129.6	129.6	163.2	1	0	1	0	
7	19-mar-2017 00:36:03.859	-123.08	20.5	112.8	-151.2	160.8	105.6	-158.4	158.4	1	0	1	0	
8	19-mar-2017 00:36:04.359	-130	18.154	-228	72	108	-168	369.6	556.8	1	0	1	0	
9	19-mar-2017 00:36:04.859	-123.85	20.5	-69.6	180	160.8	-57.6	-432	158.4	1	0	1	0	
10	19-mar-2017 00:36:05.359	-129	20.5	182.4	31.2	160.8	182.4	24	153.6	1	0	1	0	
11	19-mar-2017 00:36:05.859	-129.5	20.5	2.4	-189.6	160.8	-9.6	-192	158.4	1	0	1	0	
12	19-mar-2017 00:36:06.359	-127	20.5	-192	26.4	160.8	-192	33.6	153.6	1	0	1	0	
13	19-mar-2017 00:36:06.859	-127	20.5	-192	26.4	160.8	-192	33.6	153.6	1	0	1	0	
14	19-mar-2017 00:36:07.359	-128.5	20.5	165.6	-86.4	160.8	153.6	-96	158.4	1	0	1	0	
15	19-mar-2017 00:36:07.859	-119.38	20.5	-110.4	-153.6	160.8	-120	-148.8	163.2	1	0	1	0	
16	19-mar-2017 00:36:08.359	-120	20.5	-141.6	132	160.8	-134.4	139.2	153.6	1	0	1	0	
17	19-mar-2017 00:36:08.859	-122.69	20.5	144	110.4	160.8	148.8	105.6	158.4	1	0	1	0	
18	19-mar-2017 00:36:09.359	-117.5	20.5	84	-168	160.8	76.8	-172.8	158.4	1	0	1	0	
19	19-mar-2017 00:36:10.858	-120	20.5	182.4	-2.4	160.8	182.4	-9.6	158.4	1	0	1	0	
20	19-mar-2017 00:36:11.358	-127.5	18.792	-31.2	-175.2	-136.8	264	-187.2	148.8	1	0	1	0	
21	19-mar-2017 00:36:11.858	-128.5	20.5	-184.8	60	160.8	-216	67.2	158.4	1	0	1	0	
22	19-mar-2017 00:36:12.358	-129	20.5	76.8	165.6	160.8	86.4	163.2	158.4	1	0	1	0	
23	19-mar-2017 00:36:12.858	-129	20.5	76.8	165.6	160.8	86.4	163.2	158.4	1	0	1	0	
24	19-mar-2017 00:36:14.358	-125.5	20.5	163.2	79.2	160.8	168	72	153.6	1	0	1	0	
25	19-mar-2017 00:36:14.858	-129	20.5	52.8	-180	160.8	43.2	-182.4	153.6	1	0	1	0	
26	19-mar-2017 00:36:15.358	-129.5	20.5	-189.6	-24	160.8	-192	-19.2	163.2	1	0	1	0	
27	19-mar-2017 00:36:15.858	-125.5	20.5	-2.4	187.2	160.8	4.8	182.4	158.4	1	0	1	0	
28	19-mar-2017 00:36:17.358	-123.46	20.5	-168	91.2	160.8	-163.2	96	163.2	1	0	1	0	

Fonte: Tominaga et al. (2017).

B.3 Telecomandos enviados

Telecomandos enviados pelo SATCS podem ser exportados na forma de arquivo texto em formato CSV, em que parâmetros referentes ao envio de telecomandos são arranjados em colunas e tempos de envio organizados em linhas.

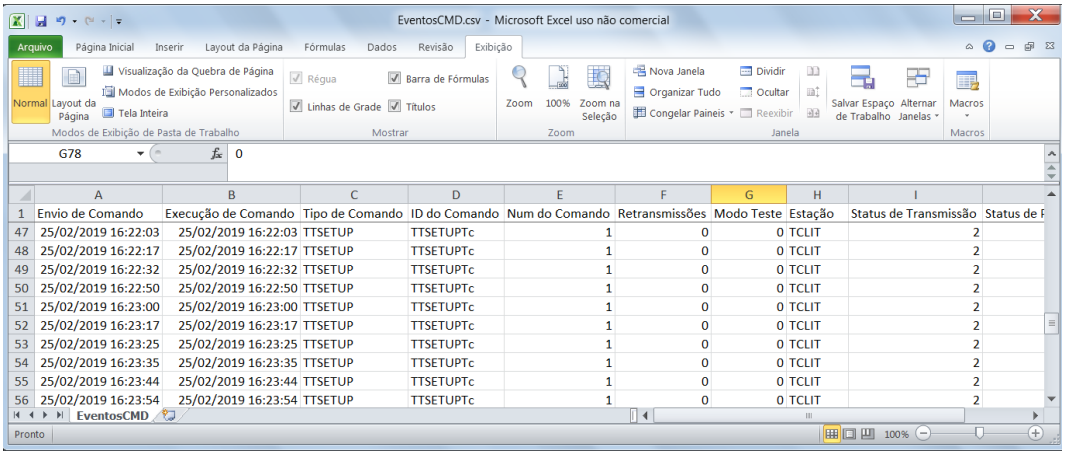
O SATCS permite selecionar um ou mais dos seguintes parâmetros a serem exportados no arquivo CSV.

- Envio de Comando
- Execução de Comando
- Tipo de Comando
- ID do Comando
- Número do Comando
- Retransmissões
- Modo Teste
- Estação
- Status de Transmissão
- Status de Pré-Verificação

- Status de Execução
- Parâmetro do Comando

Um exemplo de arquivo CSV de envio de telecomandos é apresentado na figura a seguir (Figura B.3).

Figura B.3. Arquivo CSV de telecomandos aberto como planilha.



	A	B	C	D	E	F	G	H	I
1	Envio de Comando	Execução de Comando	Tipo de Comando	ID do Comando	Num do Comando	Retransmissões	Modo Teste	Estação	Status de Transmissão
47	25/02/2019 16:22:03	25/02/2019 16:22:03	TTSETUP	TTSETUPTc	1	0	0	TCLIT	2
48	25/02/2019 16:22:17	25/02/2019 16:22:17	TTSETUP	TTSETUPTc	1	0	0	TCLIT	2
49	25/02/2019 16:22:32	25/02/2019 16:22:32	TTSETUP	TTSETUPTc	1	0	0	TCLIT	2
50	25/02/2019 16:22:50	25/02/2019 16:22:50	TTSETUP	TTSETUPTc	1	0	0	TCLIT	2
51	25/02/2019 16:23:00	25/02/2019 16:23:00	TTSETUP	TTSETUPTc	1	0	0	TCLIT	2
52	25/02/2019 16:23:17	25/02/2019 16:23:17	TTSETUP	TTSETUPTc	1	0	0	TCLIT	2
53	25/02/2019 16:23:25	25/02/2019 16:23:25	TTSETUP	TTSETUPTc	1	0	0	TCLIT	2
54	25/02/2019 16:23:35	25/02/2019 16:23:35	TTSETUP	TTSETUPTc	1	0	0	TCLIT	2
55	25/02/2019 16:23:44	25/02/2019 16:23:44	TTSETUP	TTSETUPTc	1	0	0	TCLIT	2
56	25/02/2019 16:23:54	25/02/2019 16:23:54	TTSETUP	TTSETUPTc	1	0	0	TCLIT	2

Fonte: Produção do autor.

B.4 Interfaces com o Simulador de Satélite (SATSIM)

O Simulador de Satélite SATSIM foi escolhido devido à sua simplicidade e familiarização do autor com sua estrutura de dados. O SATSIM permite a criação de modelos tão simples quanto uma única regra de usuário acrescida de um par de regras de controle de simulação, composto por uma regra de atualização de passos e outra de parada de simulação. Parâmetros de simulação e seus valores iniciais são obtidos a partir de um arquivo de estado inicial. Eventos cujos instantes de ocorrência são calculados externamente são registrados em um arquivo de fila de eventos temporizados. Os valores de todos os parâmetros de simulação, recalculados por meio da avaliação de todas as regras de controle que compõem o modelo comportamental a cada passo de simulação, são registrados num arquivo de histórico de simulação.

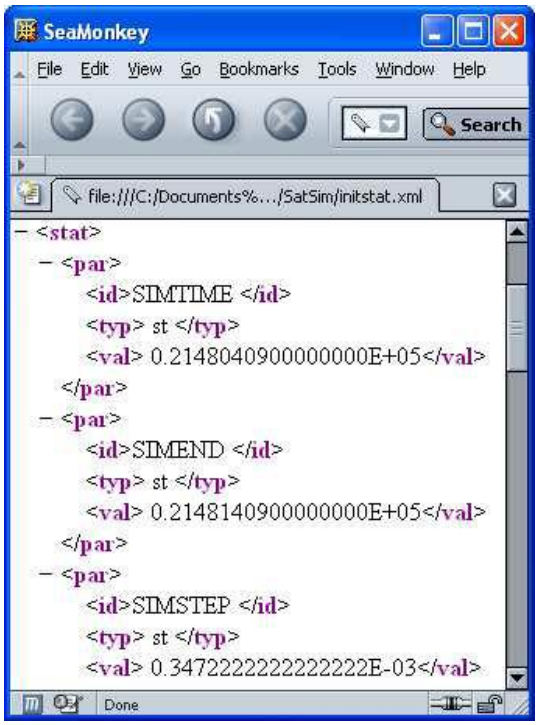
O SATSIM utiliza arquivos texto em formato XML (Extended Markup Language) para armazenar informações de parâmetros de simulação e regras comportamentais de modelos de satélite. O histórico de simulação contendo a evolução temporal de valores dos parâmetros de simulação é exportado na forma de arquivo texto em formato CSV.

B.5 Estado inicial

O estado inicial do sistema simulado corresponde a um conjunto de parâmetros de simulação que definem o estado interno do sistema no passo de simulação inicial da sessão de simulação. Além de estabelecer valores que caracterizam o sistema no início da sessão, o estado inicial é responsável também por definir o conjunto de parâmetros existentes no modelo. Desta forma, todos os parâmetros que fazem parte do modelo devem ser declarados neste arquivo, uma única vez (TOMINAGA, 2010).

O estado inicial é constituído por um arquivo XML contendo uma lista de estruturas de parâmetros identificada pela chave **<stat>**. Cada parâmetro é indicado pela chave **<par>**. A figura a seguir (Figura B.4) mostra um exemplo de arquivo de estado inicial. Parâmetros são referenciados por regras através de seu identificador (**<id>**), possuem um indicador de tipo (**<typ>**) que indicam se o parâmetro é do tipo estado (**"st"**) ou do tipo evento (**"ev"**). O estado interno é definido através de valores (**<val>**) de ponto flutuante assumidos por cada parâmetro.

Figura B.4. Exemplo de arquivo de estado inicial de simulação.



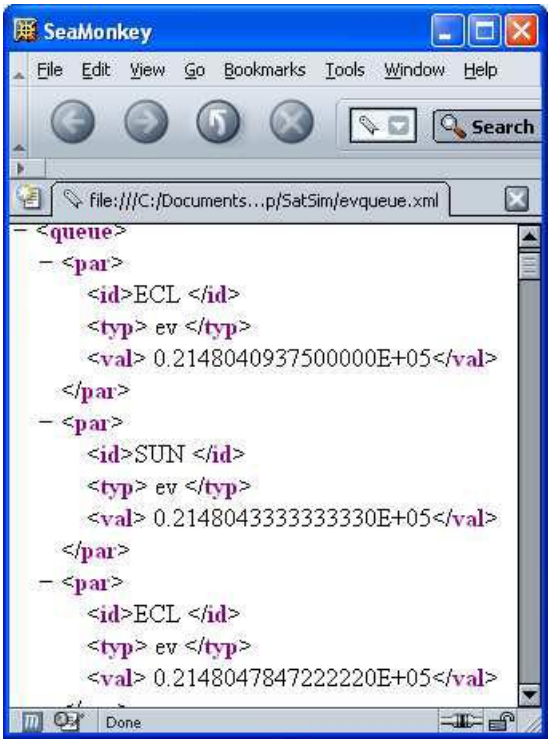
Fonte: Tominaga (2010).

B.6 Fila de eventos

A fila de eventos contém uma compilação de previsões de eventos e atividades representados como uma lista temporizada de instâncias de parâmetros de simulação do tipo evento.

A fila de eventos é constituída por um arquivo XML contendo uma lista de estruturas de parâmetros identificada pela chave **<queue>**. A figura a seguir (Figura B.5) mostra um exemplo de arquivo de fila de eventos.

Figura B.5. Exemplo de arquivo de fila de eventos de simulação.



Fonte: Tominaga (2010).

B.7 Regras de controle

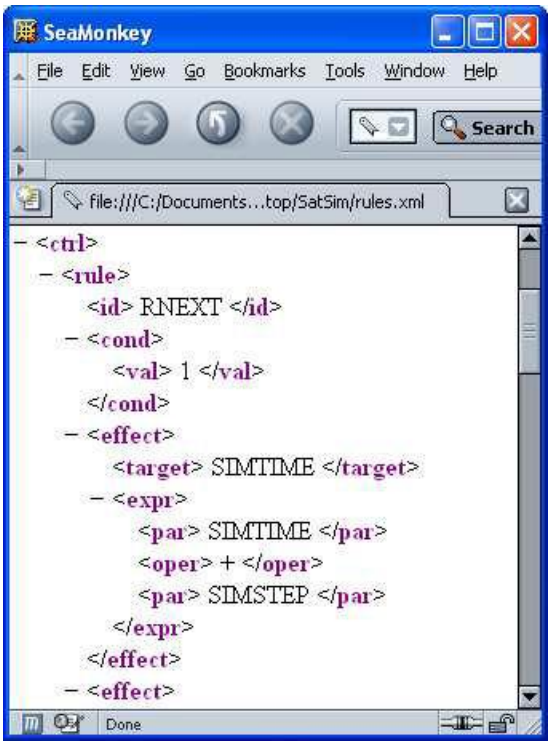
As regras de controle explicitam como parâmetros de simulação contidos no estado interno devem ser alterados em cada passo de simulação, ao longo da sessão de simulação (TOMINAGA, 2010).

As regras de controle são registradas em um arquivo XML contendo uma lista de estruturas de regras identificadas pela chave **<ctrl>**. Cada regra é indicada pela chave **<rule>** e contém uma expressão de precondição (**<cond>**) e um ou

mais expressões de efeito (**<effect>**) cujo valor resultante da avaliação é atribuído a um parâmetro alvo (**<target>**). Expressões são compostas por operandos, operações e expressões. Operandos podem incluir parâmetros (**<par>**), valores (**<val>**) e curvas (**<curv>**). Operações podem incluir operações elementares (**<oper>**) e funções (**<func>**) Expressões podem ser explicitadas por meio da chave **<expr>**.

A figura a seguir (Figura B.6) mostra um exemplo de arquivo de regras de controle.

Figura B.6. Exemplo de arquivo de regras de controle de simulação.



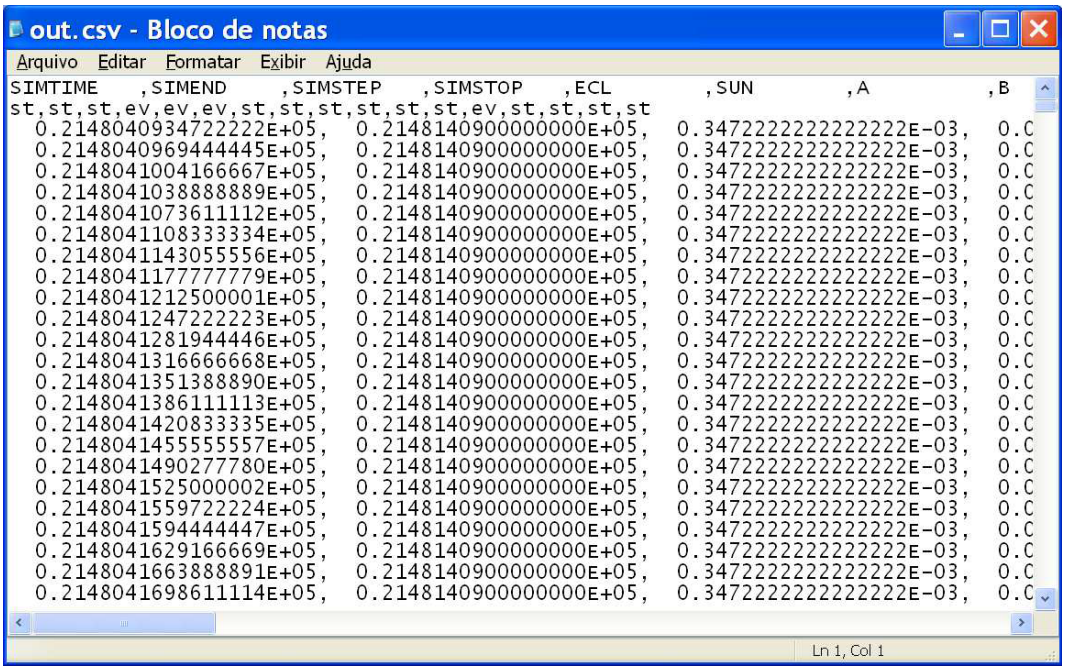
Fonte: Tominaga (2010).

B.8 Histórico de simulação

Os parâmetros do histórico de estados são exportados em forma de tabela, com colunas representando a evolução temporal de valores de um parâmetro de estado, e linhas representando o estado interno num determinado passo de simulação. O formato deste arquivo é CSV. (TOMINAGA, 2010)

A figura a seguir (Figura B.7) mostra um exemplo de arquivo de histórico de simulação.

Figura B.7. Exemplo de arquivo de regras de controle de simulação.



Fonte: Tominaga (2010).

B.9 Interfaces internas do Reconfigurador de Modelo

Para desempenhar a finalidade proposta, o protótipo do Reconfigurador de Modelo deve ser capaz de desempenhar as seguintes funções referentes a interfaces com demais componentes do sistema cuja arquitetura foi descrita no Capítulo 3.

- Ler o histórico de simulação do Simulador de Satélite
- Ler as telemetrias processadas do Sistema de Solo de TT&C convertidas para um formato compatível com o histórico de simulação do Simulador de Satélite
- Ler as regras comportamentais do Simulador de Satélite
- Executar o Simulador Simplificado
- Exportar um novo estado inicial do Simulador de Satélite
- Exportar novas regras comportamentais do Simulador de Satélite

À lista anterior foi acrescida uma função de aquisição de informações de mapeamento entre telemetrias e parâmetros de simulação correspondentes e valores de erro definidos por telemetria para iniciar e finalizar a busca por modelos comportamentais atualizados.

- Ler informações de mapeamento entre telemetrias e parâmetros de simulação correspondentes e valores de erro

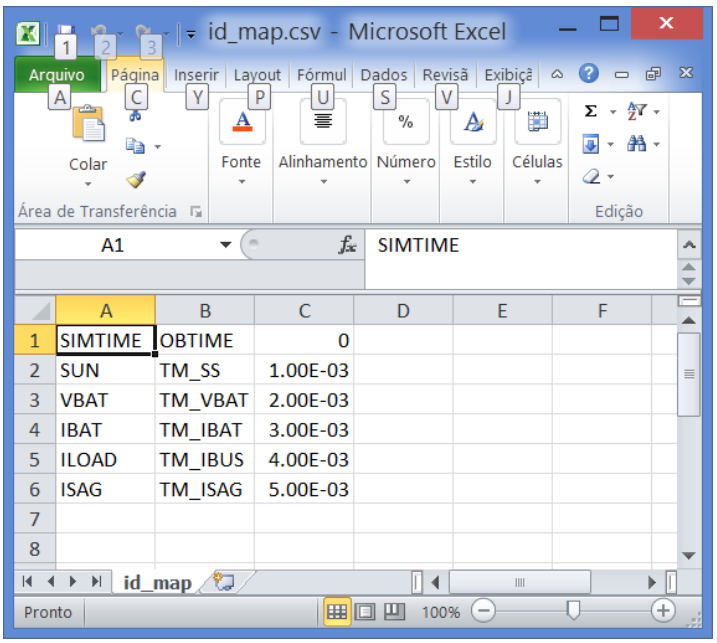
B.10 Mapa de telemetrias e parâmetros de simulação

O mapa de telemetrias e parâmetros de simulação correspondentes é um arquivo texto em formato CSV, onde os identificadores de parâmetros de simulação são agrupados na primeira coluna e as telemetrias correspondentes na mesma linha, na segunda coluna.

Foi adicionada a este arquivo uma terceira coluna para o registro de valores de limites de erro absoluto por parâmetro para disparar o processo de busca de novos arquivos XML para o Simulador de Satélite.

A figura a seguir (Figura B.8) exemplifica um arquivo de mapeamento de telemetrias e parâmetros de simulação.

Figura B.8. Exemplo de mapa de telemetrias e parâmetros de simulação.



Fonte: Produção do autor.