AUTOMAÇÃO DE EXECUÇÃO DE CASOS DE TESTE PARA SISTEMA DE CONTROLE DE ATITUDE E ÓRBITA DE SATÉLITES

André Rodrigues Simões*, Fabrício Galende Marques de Carvalho†

*Fundação de Ciência, Aplicações e Tecnologias Espaciais
Av. Dr. João Guilhermino, 429
São José dos Campos, São Paulo, Brasil

†Instituto Nacional de Pesquisas Espaciais
Av. dos Astronautas, 1758
São José dos Campos, São Paulo, Brasil

Emails: andre.simoes@inpe.br, fabricio.galende@inpe.br

Abstract— In the development of attitude and orbit control subsystem (AOCS) of artificial satellites periodic testing tasks are required to confirm that all system functionalities meets project requirements. In this context, the planning of creation and execution of tests has similar importance to the product planning step. AOCS development requires time; the final product is complex. Test automation is a mechanism to increase product quality and reduce time spent in such activity. However, particularities not found in comercial software testing must be taken into account for control system products due to its non-deterministic response characteristic and the large amount of input/output data representing signals. Based on these observations, this article states about analysis and application of two automated test execution techniques tipically stated on commercial software in AOCS development context. From the analysis, tailorable approaches suited to our case study were identified and two automatic test execution solutions stimulated by small textual descriptions capable to provide modular refactoring and reuse of test functions were created in order to fasten product test cycles.

Keywords— AOCS, artificial satellite validation and verification, automated test execution, control system test, S/W test, functional decomposition, keyword driven test

Resumo— O desenvolvimento de um subsistema de controle de atitude e orbita (AOCS) de um satélite artificial requer tarefas periódicas de teste para garantir que as funcionalidades atendam requisitos de projeto. Neste contexto, planejar a produção e execução dos testes tem importância similar ao planejamento do produto a ser desenvolvido. Desenvolver o AOCS requer tempo, o produto final é complexo. Automatizar a execução dos testes é uma forma de reduzir o tempo gasto na atividade e aumentar a qualidade do produto. Mas em um sistema de controle, a característica estocástica das respostas bem como o grande volume de dados estocásticos amostrados nos testes exigem cuidados diferenciados não encontrados em testes de produtos de outra natureza. Baseado nestas observações, este artigo analisa a aplicação de duas técnicas de automação de execução de testes no contexto de desenvolvimento de AOCS. A partir da análise, abordagens adaptáveis foram identificadas para o estudo de caso e duas soluções de execução automatica de testes foram geradas para agilizar os ciclos de teste do produto excitadas por pequenas descrições textuais que permitem reuso e atualização de funções de teste.

Palavras-chave— AOCS, verificação e validação de satélites artificias, execução automatica de testes, teste de sistema de controle, teste de S/W, decomposição funcional, teste dirigido por palavras chaves

1 Introdução

O projeto do sistema de controle de atitude e órbita é uma importante etapa relacionada ao desenvolvimento de satélites artificiais. Nesta etapa, tipicamente designa-se por AOCS (*Attitude and Orbit Control Subsystem*) o subsistema responsável por controlar a direção do satélite no espaço ou alterar sua órbita conforme os comandos que partem de solo. O desenvolvimento do AOCS resulta em um equipamento eletronicamente conectado e um software (S/W) embarcado expecífico para controle caracterizado por equações representativas de dinâmica, fenômenos físicos e por uma série de restrições impostas por requisitos funcionais e não funcionais do satélite. Estas restrições e exigências tornam o AOCS altamente complexo e, por este motivo, é comum encontrar várias fases de teste do satélite no cronograma de agências espaciais (Wertz and Larson, 2010) para garantir alto grau de qualidade.

O papel do teste é o de determinar se o produto se comporta de acordo com os requisitos de projeto (Sommerville, 2011). Testar exige tempo em análise das necessidades e criação, adaptação e atualização de código de teste ao longo do desenvolvimento do produto. Tarefa que pode ser facilitada por técnicas de automatização de execução de teste já conhecidas. Em se tratando de S/W comercial, (Song et al., 2010), (Xing et al., 2009), (Nagowah and Doorgah, 2012) e (Fewster and Graham, 1999) mencionam típicos exemplos desta abordagem de execução automatizada de testes. Contudo, devido ao contexto em que foram engendradas, suas métricas de validação tipicamente baseiam-se em dados determinísticos sem a presença de comportamento dinâmico. No S/W de AOCS, por exemplo, um sinal constante como entrada incita respostas distintas nas saídas, que podem se modificar ao longo do tempo, de acordo com o estado em que o satélite se encontra.

Observando estas considerações, este trabalho

analisa a aplicação de duas técnicas de automatização de execução de testes em um sistema de controle de satélite simplificado. No contexto de projeto de AOCS, esta análise tem como objetivo:

- Verificar a aplicabilidade das técnicas;
- Automatizar totalmente ou parcialmente as atividades de teste;
- Facilitar o processo de atualização dos testes;
- Viabilizar o reuso de funções de teste.

Este trabalho esta estruturado da seguinte forma: A seção 2 descreve o sistema de controle de atitude simplificado que foi submetido aos testes (*System Under Test* - SUT) automaticamente executados pelas soluções obtidas. A seção 3 disserta sobre as duas técnicas de teste escolhidas para a análise. A seção 4 detalha a viabilidade de aplicação das técnicas para a geração das ferramentas que aplicam automaticamente os testes. A seção 5 mostra resultados obtidos pelas execuções das ferramentas de teste geradas a partir de cada técnica. A seção 6 traz as conclusões obtidas perante a análise efetuda.

2 Sistema em Teste

A figura 1 descreve um modelo simplificado de controle e dinâmica de um (1) eixo para um satélite que utiliza uma roda de reação contendo saturação mas sem zona morta e atrito para controle de atitude. O tipo de órbita adotado para este satélite é circular.

Neste sistema, o sinal de controle indica à *roda de reação* o valor de torque necessário para que o Satélite assuma a orientação. A roda de reação, respeitando sua limitação de velocidade máxima, acelera ou desacelera para gerar um torque no corpo do satélite. O bloco *Satélite* recebe o torque da roda e modifica sua atitude de forma a se aproximar ao valor de referência no decorrer do tempo. O bloco *sensores* usa giroscópios para medir a velocidade e magnetômetros junto a sensores solares (*coarse sun sensors*) para medir a direção angular do satélite. O bloco *determinação de atitude* é o algoritmo determinístico TRIAD (Shuster and Oh, 1981) que calcula a posição angular através dos resultados de funções de determinação de direção do sol e do campo magnético baseados na leitura dos sensores. O bloco *filtro de atitude* tem a função de um filtro das medições que, logo em seguida, são realimentadas. As configurações físicas deste modelo são descritas na Tabela 1. A descrição do funcionamento dos sensores e da roda de reação é dada por (Wertz, 2002).

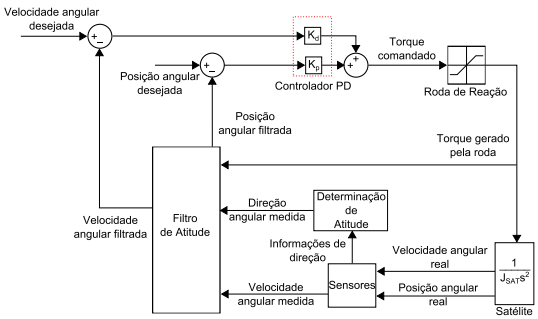

Figura 1: Modelo de sistema de controle simplificado de um satélite

Variável	Descrição física	Valor
J_{SAT}	momento de inércia do satélite	184.1176 $kg.m^2$
J_{RW}	momento de inércia da roda de reação	0.0028 $kg.m^2$
$V_{MAX_{RW}}$	Velocidade máxima da roda de reação	± 5000 rpm
$T_{MAX_{RW}}$	Torque máximo gerado pela roda de reação	± 0.11 N.m

Tabela 1: Configurações usadas nas simulações

3 Técnicas de execução automática de teste

Tipicamente, o AOCS é um sistema desenvolvido através de um processo evolutivo. Por questões financeiras ou de negócio, ele pode sofrer alterações arquiteturais ao longo do projeto. No ambiente em que o produto pode mudar de forma suave ou drástica, o teste de funcionalidades deve ser orientada por parâmetros comuns às diferentes arquiteturas (e.g. precisão de apontamento). Os casos de teste poderiam ser descritos em linguagem de alto nível, poupando detalhes de testes subsistêmicos e deixando que os mesmos possam ser tratados por subrotinas de teste mais específicas. Neste cenário, também se faz necessário o reuso de código de teste mediante a evolução do sistema. Não seria interessante refazer toda a solução de teste se apenas uma parte isolada do produto sofreu alteração/reconfiguração.

As técnicas de decomposição funcional e de teste dirigido por palavras chaves exigem/permitem estruturação modular do código de teste e viabilizam reuso. Nestas abordagens os casos de teste são feitos em pseudocódigo em artefatos distintos ao código de teste. Com estas características, ambas as técnicas são candidatas a auxiliar no desenvolvimento de AOCS, e serão analisadas logo a seguir.

3.1 Decomposição funcional

Na decomposição funcional, as funcionalidades de um produto são decompostas e invocadas em funções e sub-funções que se correlacionam de forma hierarquicamente organizada. Tais funções (ou scripts) são usadas de forma isolada para formar casos de teste de submódulos ou para compor casos de teste de nível sistêmico ao serem chamados pelas funções de teste que atuam neste último nível. Os valores de entrada permanecem em arquivos separados.

Nesta abordagem, do ponto de vista computacional, o reaproveitamento das funções de teste em outras funções ou algoritmos de teste de nível sistêmico mais alto gera uma *pilha de chamada de funções de teste*. O diagrama de atividades simplificado da figura 2 descreve o processo de geração das funções de teste e a correlação entre as funções hierarquicamente vizinhas que geram esta característica.

Em caráter especial, a ação *Gerar função de teste para o requisito específico*, que tem seu comportamento descrito na figura 3, deverá ser chamada mais vezes se existir a necessidade de gerar diferentes funções para atingir os objetivos de teste. O diagrama de atividades da figura 3 também observa que, nem todas as funções de teste de requisitos filhos devem ser utilizadas em funções de teste de requisito de nível mais alto de acordo com o escopo de atividades da função. Adicionalmente, funções utilitárias (e.g. *delays*, geradores de relatórios, etc) também podem ser disponibilizadas para diminuir o trabalho de desenvolvimento dos casos de teste.

3.2 Teste dirigido por palavras chaves

O teste dirigido por palavras chaves (*Keyword Driven Test*) é uma técnica que utiliza palavras chaves, acompanhadas ou não por dados, para descrição em alto nível dos testes. Estas palavras são processadas por um script diretor para executar outros scripts que invocam funcionalidades específicas do produto em teste. A figura 4 descreve um modelo de processo simplificado que gera funções de teste e as palavras chaves correlacionadas. A identificação de subfunções e iteração do código caracterizam reuso e organização/simplificação ao longo do desenvolvimento da solução. Funções de teste subsistêmico podem ser reutilizados por testes sistêmicos. Adicionalmente, funções de teste desenvolvidas por outras técnicas, como a decomposição funcional, podem ser adaptadas e reusadas neste contexto.

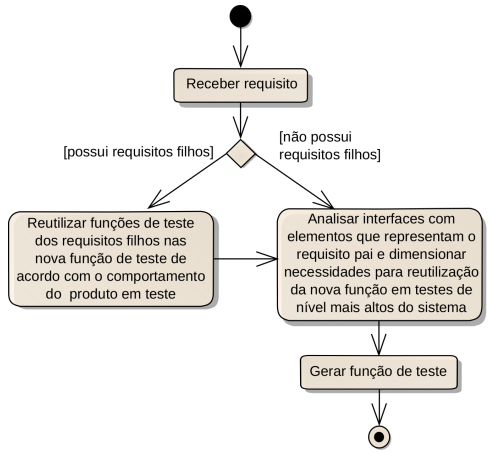

Figura 3: Geração de função de teste para um requisito específico

4 Estudo de caso

4.1 Decomposição funcional

Esta técnica tem maior aplicabilidade em funcionalidades do AOCS que não apresentam comportamento dinâmico devido a organização estrutural formada entre as funções de teste. Ao aplicá-la diretamente aos blocos do sistema em malha fechada

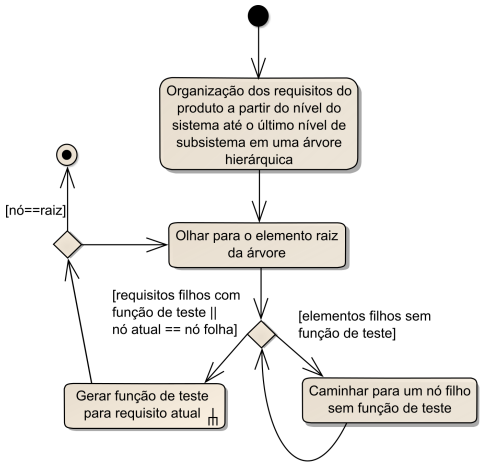

Figura 2: Descrição da mecânica para a geração das funções de teste para o sistema

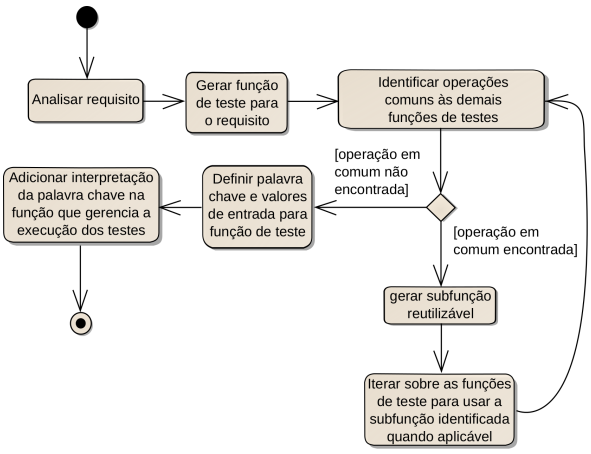

Figura 4: Atividade que processa um requisito para gerar a função de teste

observou-se a necessidade de uma análise de comportamento dinâmico adicional para a geração de valores de entrada às funções de nível subsistêmico quando as mesmas eram chamadas em funções de nível sistêmico (e.g. teste da roda de reação chamada em teste do sistema de controle). Observou-se também a necessidade adicional de substituição das conexões de entrada e saída dos blocos testados pelos resultados das funções de teste subsistêmicas invocadas em testes de nível sistêmicos. Devido a tais observações, preferiu-se utilizar apenas os componentes de determinação de atitude, determinação da direção do Sol e componentes de simulação de sensores solares como SUT na aplicação da decomposição funcional. O determinador do campo magnético juntamente com os dados dos manetômetros foram substituídos por vetores de campo magnético informados diretamente como dados de entrada. Considerou-se como requisitos de projeto a correta implementação das funcionalidades supracitadas.

Feita a análise, um algoritmo orientado à linguagem objeto em C++ foi gerado para testar cada nível das funcionalidades. No lugar de gerar funções de teste, classes foram estruturadas para a execução dos testes e para tratamento específico das informações de entrada/saída e de métrica de teste de cada comportamento. Ao observar a presença de atributos e métodos comuns no código, uma classe pai foi gerada para definir comportamentos mínimos necessários para criação dos testes e facilitar as operações de execução e impressão de resultados obtidos.

Nesta solução, a comunicação das informações entre os vários níveis de teste foi feita através de um método na classe sistêmica que tem como argumento de entrada um ponteiro para o objeto da classe subsitêmica. Inicialmente, atributos que não dependem de atualização são acessados e seu ponteiro é armazenado internamente ao novo teste nesta mesma operação. Após a execução dos testes subsistêmicos, o teste de nível sistêmico acessa novamente o ponteiro para adquirir os resultados (valores atualizados) ao executar o teste.

Os dados de entrada e os valores esperados dos casos de teste foram organizados em arquivos de texto. Funções de leitura foram criadas para a ler estes arquivos e criar um objeto da classe de teste correspondente. Funções de geração de teste por parâmetros de entrada diretamente em código também foram geradas para testes que exigem poucas informações.

A figura 5 exemplifica um código de teste feito para testar a determinação de atitude. Para o algoritmo TRIAD são necessários quatro (4) componentes vetoriais para o cálculo da posição angular. A direção do Sol e do campo magnético observados em Terra e a direção do campo magnético observado via Satélite são informados em um arquivo texto. A direção do Sol computada pelo satélite

```
SunSensorTestCase *sensorTest[ 2 ];
SVDTestCase *sunVectorTest;
AttDetTestCase *attitudeDet;

sensorTest[0] = SunSensorTestCase::CreateTest( "tc01-sunSensor01.txt" );
sensorTest[1] = SunSensorTestCase::CreateTest( "tc01-sunSensor02.txt" );
sunVectorTest = SVDTestCase::CreateTest( sensorTest[0], sensorTest[1],
   "tc01-sunVector.txt" );
attitudeDet = AttDetTestCase::CreateTest( sunVectorTest, "tc01-attitude.txt" );

TestCase *testArray[] = {sensorTest[0],sensorTest[1],sunVectorTest,attitudeDet};

for( unsigned int i = 0; i < 4; ++i )
{
   testArray[i]->RunTest();
   testArray[i]->PrintExpectedVersusReal();
}
```

Figura 5: Código de teste gerado através da solução baseada em decomposição funcional

é informada pelo resultado do caso de teste de determinação de direção do Sol. Este último faz uso dos resultado de dois (2) testes de sensores solares para realizar seu cálculo. Nesta arquitetura, problemas detectados em etapas subsistêmicas são reportadas antes da execução do teste de nível sistêmico. Se os testes são específicos para um componente subsistêmico (teste isolado de sensores solares) este mesmo código diretor (*driver* de teste) deverá ser gerado especificamente para tal.

4.2 Teste dirigido por palavras chaves

Considerando o comportamento dinâmico em malha fechada e em blocos de controle isolados, o método de comparação entre os valores obtidos e esperados e a maneira como os dados dos testes são informados são os principais pontos a serem observados na implementação da técnica. Para torná-la viável, pontos que descrevem o valor inícial e final da referência de entrada foram correlacionados com valores de tempo na descrição dos casos de teste. A partir deles, funções específicas da ferramenta de teste são encarregadas de gerar todos os valores *intermediários* que compõem o sinal ao longo da simulação. Geradores de sinais do tipo *degrau* ou *rampa* foram usados nesta abordagem de acordo com o tipo do caso de teste.

Os valores esperados podem ser informados de duas formas: 1) com um valor constante usado para verificar se a resposta tende à este valor; 2) com valores retornados por uma função externa para descrever o comportamento esperado do sistema. Ambas as formas são acompanhadas de uma segunda variável que define a faixa de tempo da simulação que permite a comparação total dos resultados ou a comparação parcial dos dados da simulação (e.g análise de precisão em regime estacionário ou transiente).

Uma margem de erro também foi adicionada para a execução das comparações. Etapas iniciais do ciclo de desenvolvimento do S/W de AOCS tendem inicialmente à utilizar modelos simplificados ou ideiais à titulo de análise de viabilidade

Palavra-chave	Dados de entrada	Resultados Esperados
SetMinError	0.01	
SetSimTime	80	
rw.ApplyStepTorque	startTime[0 30] inputTorque [0.11 -0.11]	externCall rwBehavior(5000, 0.11, 0.0028, 'rwInputTorque', 80, 0.1)
SetSimTime	7200	
sat.changeDirectionTo	angularPosition [0 360] startTime [0 7200] signalType Ramp	checkTime [200 7180] externCall rampFunction(360, 7200, 0.1, 7200) maxRwSpeed 5000
sat.changeDirectionTo	angularPosition [6] startTime [0] signalType Step	checkTime 40 angularPosition 6

Tabela 2: Exemplo de caso de teste feito em tabela com palavras chaves

com refinamento e melhorias ao longo dos novos ciclos. Uma margem de erro, neste caso, é fatalmente esperada. Além disso, o sistema de controle tipicamente oferece resultados que tendem a um determinado valor e a métrica de comparação utilizada em S/W comerciais puramente determísticos não seria adequada para verificar os resultados dos testes.

Uma outra necessidade que foge aos S/W de outra natureza é o uso de constantes *físicas* (vide Tabela 1). A solução foi designar um artefato específico para informar a ferramenta de teste sobre estes valores e evitar repetição destas informações nos casos de teste.

A Tabela 2 exemplifica uma especificação baseada na técnica de execução automatica de testes dirigido por palavras chaves. Nela, cada linha representa um caso de teste ou uma alteração nas configurações do teste. *SetSimTime* define o tempo total em segundos usado na simulação. *SetMinError* configura o erro entre os valores esperados e obtidos. Nesta solução, testes de componentes determinísticos de S/W não foram considerados.

Casos de testes isolados na roda de reação podem ser invocados pela palavra chave *rw.ApplyStepTorque*; dados em *inputTorque* e *StartTime* indicam o valor de torque e seu período de estimulação. Os valores esperados são gerados por uma função externa definida pelo usuário através de *externCall*; quando o parâmetro é omitido, a ferramenta utiliza uma função interna de geração dos valores esperados.

Casos de teste no sistema em malha fechada são invocados por *sat.changeDirectionTo*; a tripla (*angularPosition*, *startTime*, *signalType*) indica a posição angular que o satélite deve apontar, os instantes em que estes valores são colocados na entrada e o tipo sinal que será gerado entre os instantes, respectivamente. Em *Expected Results*, *checkTime* aliada a *angularPosition* ou *externCall* definem os dados esperados e o intervalor de comparação com os resultados. Adicionalmente, *maxRwSpeed* permite verificar se a roda se mantêm dentro da velocidade especificada ao longo do teste. O apontamento do satélite para o Sol (modo de contingência) é feito com entradas degrau (Sol em posição fixa); apontar o satélite para a Terra exige manutenção da sua velocidade de referência. No caso de uma órbita circular a velocidade tende à um valor constante e uma rampa pode ser utilizada na entrada para o teste.

5 Resultados Obtidos

5.1 Decomposição Funcional

Inicialmente, variações foram aplicadas aos valores dos arquivos lidos pelo driver de teste descrito pela figura 5 para verificar se a ferramenta orientada à decomposição funcional detectava corretamente as alterações de parâmetros. Desconsiderando os algoritmos de determinação, teste de valores limites foram aplicados com relação a detecção de raios solares na borda do campo de visão dos sensores e também com relação à máxima incidência de luz solar.

Para validar o determinador de atitude e direção solar, dois sensores foram configurados em posições que permitiam a computação do Sol. Casos de teste para identificar a correta operação dos algoritmos foram definidos e aplicados. Erros propositais foram adicionados aos códigos para avaliar a detecção dos mesmos. Foram testados os cenários que permitem o cálculo da atitude e cenários de singularidade (paralelismo entre a vetor da direção do Sol e do campo magnético que inviabiliza o uso do TRIAD). Os sensores utilizados eram livres de incertezas e ruídos.

As figuras 6a, 6b, 7a e 7b, exemplificam um cenário de paralelismo testado em um algoritmo determinador de atitude que foi submetido a mutação. Ambos os sensores foram configurados com um campo de visão (*fov*) de 180 graus e fornecimento de corrente máxima de incidência solar (*maxCurrent*) de 15mA. O primeiro sensor se en-

```
minError 0.00001

fov 180
normal [ 0 1 ]
maxCurrent 15
expectedOutput 15
```

(a) tc01-sunSensor01.txt

```
minError 0.00001

fov 180
normal [ 0.03489949 0.99939082 ]
maxCurrent 15
expectedOutput 14.99086
```

(b) tc01-sunSensor02.txt

Figura 6: Fragmento dos arquivos que definem a configuração dos dois sensores usados no teste

```
minError 0.001
refSunDir [ 0 1 ]
refMagDir [ 0 1 ]
bodyMagDir [ 0 1 ]

expectedDegreeRotation -1
```

(a) tc01-attitude.txt

```
minError 0.0001
expectedSunVector [ 0 1 ]
```

(b) tc01-sunVector.txt

Figura 7: Arquivos de entrada dos testes de determinação de atitude e determinação do vetor Sol

contra apontado para cima (*normal*), exatamente a 90 graus do eixo-x. O segundo sensor foi direcionado a 88 graus. A direção dos raios solares que partiram do vetor (0,1) foram informados como entrada dos sensores e como saída esperada (*expectedSunVector*) do algoritmo de determinação da direção do Sol.

Para o determinador de atitude foram informados as direções do Sol e do campo magnético vistas de Terra (*refSunDir* e *refMagDir*) e a direção do campo magnético vista pelo satélite (*bodyMagDir*). Em cenários de não paralelismo dos vetores o algoritmo deve retornar rotações entre 0 e 360 graus. Como o cálculo é impossível, a resposta esperada (*expectedDegreeRotation*) especificada para o algoritmo é o valor -1. A figura 8 mostra a identificação de um erro no tratamento da singularidade pela solução.

5.2 Teste dirigido por palavras chaves

Para comprovar a eficácia dos testes, a ferramenta solução foi configurada com os parâmetros físicos descritos na tabela 1. Testes foram aplicados na configuração original do SUT seguidos dos mesmos testes em versões alteradas do SUT (erro em conexões ou alterações de parâmetros físicos). Para as funções externas permitidas por *extern-*

```
expected sun sensor current = 15
obtained sun sensor current = 15
VEREDICT: PASSED
expected sun sensor current = 14.99086
obtained sun sensor current = 14.990862
VEREDICT: PASSED
Expected sun vector = [ 0 1 ]
Obtained sun vector = [ -3.870875e-015 1 ]
VEREDICT: PASSED
expected angular rotation (degrees) = -1
obtained angular rotation (degrees) = 0
VEREDICT: FAILED
```

Figura 8: Resultados do teste de singularidade no código com mutação

Call geramos saídas em rampa e também descrevemos o comportamento do torque e da velocidade da roda através de integração numérica. Ambas as abordagens demonstraram que a ferramenta foi eficaz na detecção de anomalias e na correta operação do sistema de controle.

Nos resultados que se seguem, juntamente com as especificações descritas na tabela 2, um erro de configuração foi simulado através da substituição do valor real de J_{RW} utilizado pelo SUT por um valor 10 vezes menor do que o especificado inicialmente.

O primeiro teste avaliou a geração de torque pelo atuador bem como seu comportamento em saturação perante uma entrada degrau. As figuras 9-12 são suas saídas gráficas. Elas evidenciam que os valores obtidos são bem dispares dos valores esperados especificados.

O segundo teste avaliou o apontamento para o Sol considerando este em uma posição angular fixa a 30 graus da atitude inicial do satélite. Neste caso o satélite não consegue acompanhar a variação angular exigida em regime transiente e alcança o regime estacionário minutos após o instante previsto no teste (figuras 13 e 14). A figura 15 mostra que a roda de reação não fornece torque necessário e satura logo de início, permanecendo neste estado devido ao erro retornado ao controlador PD que

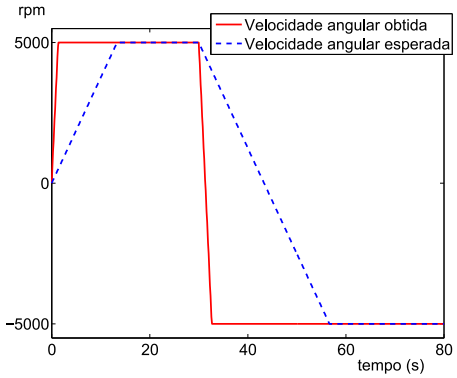

Figura 9: Velocidade esperada e obtida da roda de reação

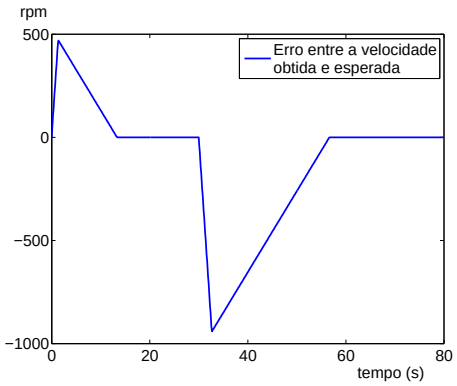

Figura 10: Erro computado na velocidade da roda de reação

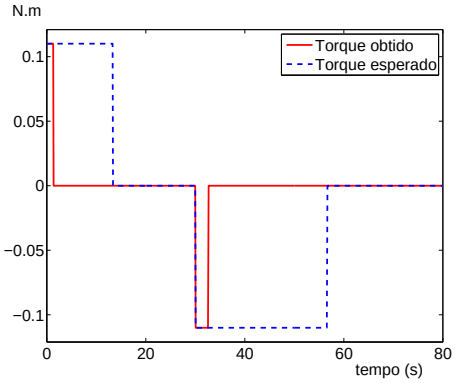

Figura 11: Torque esperado e obtido da roda de reação

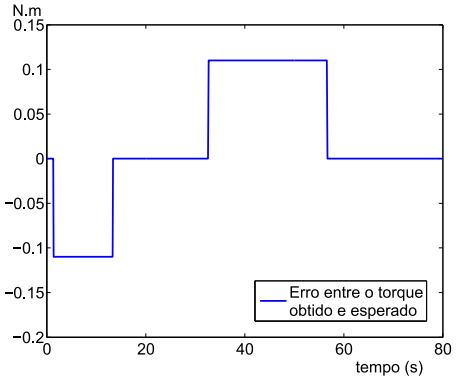

Figura 12: Erro computado no torque de saída da roda de reação

continua a exigir mais torque da roda.

O terceiro teste utilizou a mesma palavra chave do teste de apontamento para a Terra para observar uma manobra de atitude (o tipo de sinal de entrada exigido nos dois casos é uma rampa). Anteriormente ao teste, foi verificado que o SUT original atendia à especificação de velocidade angular mínima de $0.05°/s$ para girar o satélite exemplo descrito em (Wertz and Larson, 2010) completamente em torno de si próprio em aproximadamente 2 horas. Logo em seguida o SUT

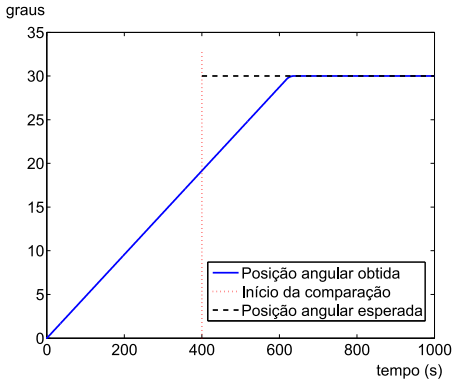

Figura 13: Apontamento para o sol esperado e obtida do satélite

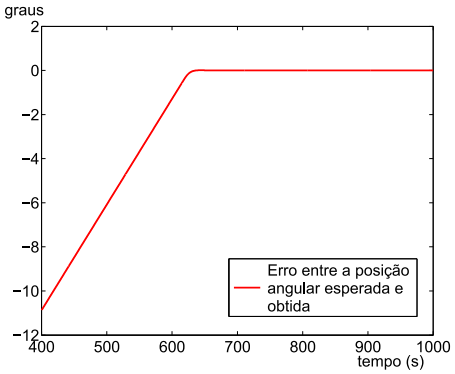

Figura 14: Erro de apontamento para o Sol do satélite

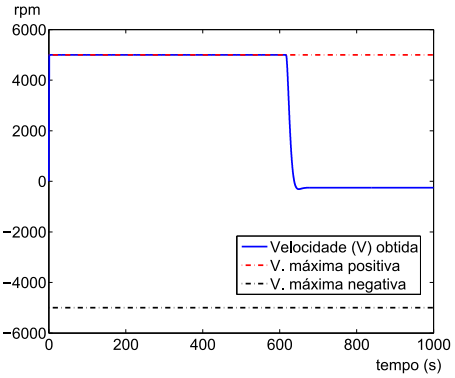

Figura 15: Velocidade gerada pela roda de reação em malha fechada (apontamento para o Sol)

com J_{RW} alterado recebeu o mesmo teste. As figuras 16, 17 evidenciam que o satélite não alcança a velocidade necessária para atender à especificação. A figura 18 mostra que a roda satura logo no início do teste.

6 Conclusões

Este estudo conclui que é possível utilizar a técnica de teste dirigo por palavras chaves para agilizar as atividades de teste do AOCS tanto no contexto de desenvolvimento do sistema de controle bem como

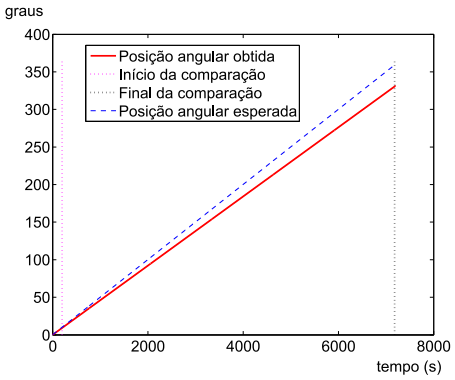

Figura 16: Apontamento (sinal rampa) esperado e obtida do satélite

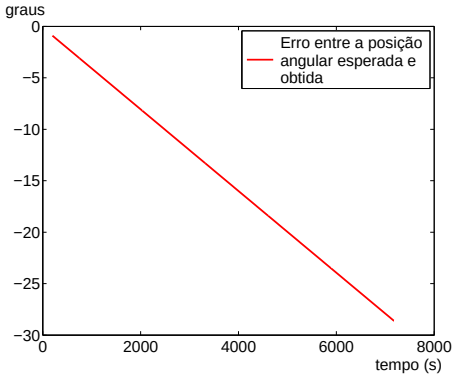

Figura 17: Erro de resposta à rampa de entrada no satélite

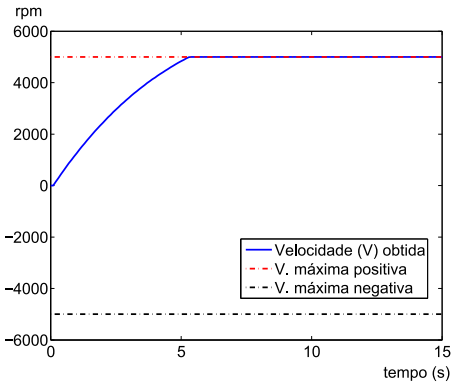

Figura 18: Velocidade gerada pela roda de reação em malha fechada (entrada rampa)

no contexto de desenvolvimento do S/W. A correta modularização das funcionalidades de teste permite a adaptação e reuso de funções de teste específicas. Se um bloco do sistema em teste for atualizado, as funções de teste correlacionadas também poderão ser atualizadas sem a necessidade de alterar a estrutura dos demais casos de teste desde a sua especificação até o código de teste propriamente dito. A especificação dos testes em tabelas permite uma visualização mais clara dos testes e não exige do testador conhecimentos de codificação da ferramenta. Contudo, ainda é necessário o conhecimento de uma pseudo-linguagem para a definição dos casos de teste.

Também ficou claro que a técnica de decomposição funcional permite automatização dos testes do sistema de controle. De acordo com nosso estudo, esta é mais indicada no contexto de S/W de AOCS e configura uma automação parcial da execução dos testes como um todo. Ao longo do desenvolvimento tipicamente incremental e/ou iterativo deste tipo de sistema de controle, as funções de transferência podem se tornar complexas. O custo de analize necessário para reaproveitar testes de nível subsistêmico em testes de nível sistêmico pode afetar a usabilidade da técnica.

Agradecimentos

Os autores agradecem ao Dr. Adenilson Roberto da Silva e ao Conselho Nacional de Pesquisa e Desenvolvimento Tecnológico (CNPq), por parte do apoio dado através do projeto do CNPq Nº 560131/2010-0 , intitulado "CAPACITAÇÃO EM PROJETO DO SISTEMA DE CONTROLE DE ÓRBITA E ATITUDE DE SATÉLITES ESTABILIZADOS EM 3 EIXOS, VIABILIZANDO O ACDH COMPLETO DO SATÉLITE LATTES-1".

Referências

Fewster, M. and Graham, D. (1999). *Software Test Automation*, ACM Press books.

Nagowah, L. and Doorgah, K. (2012). Improving test data management in record and playback testing tools, *Computer Information Science (ICCIS), 2012 International Conference on*, Vol. 2, pp. 931–937.

Shuster, M. D. and Oh, S. D. (1981). Three-axis attitude determination from vector observations, *Journal of Guidance and Control* **4**(1): 70–77.

Sommerville, I. (2011). *Engenharia de Software*, Pearson Education.

Song, M., Yin, G., Wang, H. and Ni, J. (2010). Effect-oriented function analysis and testing method, *Internet Computing for Science and Engineering (ICICSE), 2010 Fifth International Conference on*, pp. 34–38.

Wertz, J. R. (ed.) (2002). *Spacecraft Attitude Determination and Control*, Kluwer Academic Publishers.

Wertz, J. R. and Larson, W. J. (2010). *Space Mission Analysis and Design*, 3rd edn, Microcosm Inc.

Xing, L., Yan, L., Mian, C. and Ying, G. (2009). The testing and evaluating system for the security operating system based on the mechanism of keyword-driven, *Information Assurance and Security, 2009. IAS '09. Fifth International Conference on*, Vol. 2, pp. 471–474.