Critérios de cobertura de testes gerados a partir de linguagem natural

Área: Engenharia de Software

Aluno: Tomaz de Aquino dos Santos Junior (tasj)

Orientador: Augusto Sampaio (acas), Gustavo Carvalho (ghpc)

Avaliador: Juliano Iyoda (jmi)

Conceitos

Ao longo dos últimos anos, o crescimento de componentes de *software* e *hardware* em *Sistemas Críticos* tem aumentado drasticamente. Um relatório feito pela *NASA* [1] mostra que de 1960 a 2000, a quantidade de *software* embarcado em artefatos militares aumentou de 8% para 80%.

Dado o fato anterior, surgem muitas pesquisas voltadas ao uso de m*étodos formais* de verificação [4, 5] que, em tese, garantem a ausência de erros, contudo necessitam de um grande poder de processamento, tornando-as impraticáveis para sistemas complexos e de grande porte.

Uma alternativa complementar para avaliar o comportamento de sistemas críticos é o *Model-Based Testing* (MBT) [6, 7], que visa ser mais ágil, eficiente e menos custoso [8]. Estes benefícios são obtidos através da geração e execução automática dos casos de teste e dados vindos da especificação dos modelos.

Em meio ao contexto do *MBT*, a qualidade da especificação dos modelos é de extrema importância para um uso efetivo desta alternativa. Se o modelo especificado não captura corretamente o comportamento do sistema, testes incorretos e/ou inconsistentes podem

ser gerados. Portanto é desejável escrever os comportamentos esperados do sistema considerando diferentes níveis de abstração, utilizando notações (semi)formais.

Alguns exemplos destas notações são: *Unied Modeling Language (UML)* [6], *Lustre* [9], *Software Cost Reduction (SCR)* [10], *Communicating Sequential Processes (CSP)* [11] e o *Labelled Transition Systems (LTS)* [12], porém o uso de modelos (semi) formais pode ser um entrave, pela dificuldade do uso, necessitando um especialista na área.

Problemas

O uso do *MBT* necessita um especialista, o que pode ser um entrave em seu uso [2]. Neste contexto, a abordagem *NAT2TEST* [2] foi criada em parceria com a *Embraer*[1]: uma estratégia para geração automática de testes através de requisitos escritos em linguagem natural utilizando *Processamento de Linguagem Natural (PLN)*, à partir de uma *Linguagem Nutural Controlada (LNC)* [13].

Existem algumas versões da ferramenta *NAT2TEST*, utilizando diferentes modelos intermediários com o mesmo objetivo final de gerar casos de teste automáticamente a partir de linguagem natural. Neste trabalho, o foco será na versão que utiliza a álgebra de processos *CSP* juntamente com as ferramentas *FDR*[2] e *Z3*[3] [15].

Atualmente os casos de teste gerados pela ferramenta *NAT2TEST* não possuem critérios de cobertura. Existem potencialmente infinitas possibilidades de testar um determinado sistema, portanto é importante considerar critérios de cobertura que serão utilizados como

[1] Empresa Brasileira de Aeronáutica - http://www.embraer.com/en-us/pages/home.aspx
[2] http://www.cs.ox.ac.uk/projects/concurrency-tools/
[3] http://z3.codeplex.com/

guia na geração dos testes. Ou seja, quando os testes gerados cobrirem determinado critério, o processo de geração é interrompido.

Objetivos

O objetivo deste trabalho é definir e implementar a geração de casos de teste a partir de critérios de cobertura relevantes ao contexto do *NAT2TEST*.

A cobertura será feita utilizando o conhecimento adquirido na concepção da ferramenta *TaRGeT* [16], criada durante a parceria feita com a *Motorola Inc.*, na qual foram utilizados a álgebra de processos *CSP* juntamente com a ferramenta *FDR*.

Dois conceitos importantes no entendimento dos critérios de cobertura no âmbito do *NAT2TEST* são os *traces CSP* e os *ciclos temporais*.

A álgebra de processos CSP tem como base uma estrutura básica chamada *Processo,* entidade autonôma que possui *Interfaces*, que são como portas de entradas ou saídas para interação com o ambiente através de *eventos*. Dois ou mais processos combinados formam outro processo, que terá uma I*nterface* própria [3].

Um T*race* é basicamente o conjunto de eventos que ocorrem durante determinada execução de um *Processo CSP* [3]. O conjunto de todos os *traces* significa o conjunto de todos os "rastros" de eventos para cada possível execução de dado *Processo*.

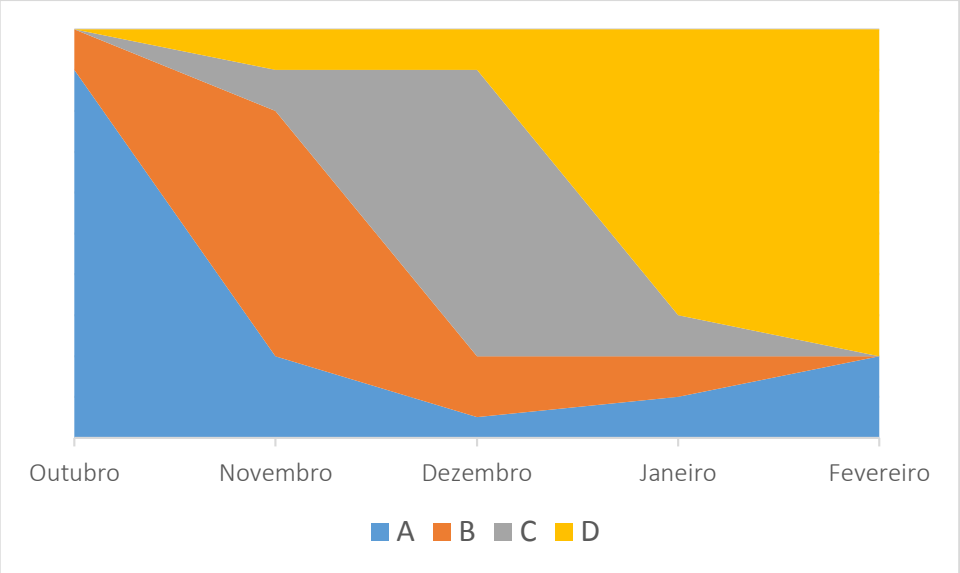
Os *ciclos temporais* estão relacionados as transições *delay* [3] presentes nos traces gerados. Cada *delay* está relacionados ao atraso da sua T*ime Guard* (variável que controla o tempo decorrido) específica [3].

Serão implementados os seguintes critérios de cobertura, além de outros que sejam julgados necessários ao longo do projeto:

1. Quantidade mínima de *ciclos temporais* presentes nos *traces CSP*;
2. Casos de teste que cobrem pelo menos todos os requisitos descritos na *LNC*;
3. *MC/DC* [14] das condições descritas pelos requisitos;
4. Caminhos e estados no modelo operacional (LTS-Labelled Transition System) gerado por FDR a partir da especificação CSP

Cronograma

a. Estudo de bibliografias relativas à ferramenta NAT2TEST e conceitos relacionados;
b. Análise da implementação dos critérios de cobertura implementados no contexto da ferramenta TaRGeT.
c. Implementação dos critérios de cobertura no escopo da NAT2TEST;
d. Elaboração do relatório final e apresentação dos resultados obtidos.



Auguto Cesar Alves Sampaio
Orientador

_____

Referências

[1] A.West, NASA Study on Flight Software Complexity, Tech. rep., NASA (2009).

[2] CARVALHO, Gustavo H. P. ; FALCÃO, Diogo ; BARROS, Flavia ; SAMPAIO, Augusto; MOTA, Alexandre ; MOTTA, Leonardo ; BLACKBURN, Mark . NAT2TEST_SCR: Test Case Generation from Natural Language Requirements based on SCR Specifications. Science of Computer Programming (Print), 2014.

[3] CARVALHO, Gustavo H. P.; SAMPAIO, Augusto; MOTA, Alexandre. A CSP Timed Input-Output Relation and a Strategy for Mechanised Conformance Verification. In: International Conference on Formal Engineering Methods, 2013, Queenstown. Proceedings of the International Conference on Formal Engineering Methods, 2013.

[4] J.-L. Camus, B. Dion, Efficient Development of Airborne Software with Scade Suite, Tech. rep., Esterel Technologies (2003).

[5] A. Mota, J. Jesus, A. Gomes, F. Ferri, E. Watanabe, Evolving a Safe System Design Iteratively, in: Proceedings of the 29th international conference on Computer safety, reliability, and security, SAFECOMP'10, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 361-374.

[6] J. Peleska, A. Honisch, F. Lapschies, H. L oding, H. Schmid, P. Smuda, E. Vorobev, C. Zahlten, A Real-World Benchmark Model for Testing Concurrent Real-Time Systems in the Automotive Domain, in: Proceedings of the 23rd IFIP WG 6.1 international conference on Testing software and systems, ICTSS'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 146-161.

[7] C. Efkemann, J. Peleska, Model-Based Testing for the Second Generation of Integrated Modular Avionics, in: International Conference on Software Testing, Verication and Validation Workshops (ICSTW), 2011, pp. 55-62.

[8] M. Utting, B. Legeard, Practical Model-Based Testing: A Tools Approach, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[9] J.-L. Bergerand, Lustre, un Langage Declaratif pour le Temps Réel , Ph.D. thesis, INPG (jan 1986).

[10] C. Heitmeyer, R. Bharadwaj, Applying the SCR Requirements Method to the Light Control Case Study, Journal of Universal Computer Science 6 (2000) 650 - 678.

[11] S. Schneider, Concurrent and Real Time Systems: the CSP Approach, John Wiley, 1999.

[12] J. Tretmans, Model Based Testing with Labelled Transition Systems, in: R. Hierons, J. Bowen, M. Harman (Eds.), Formal Methods and Testing, Vol. 4949 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2008, pp. 1-38.

[13] J. Allen, Natural Language Understanding, Benjamin/Cummings, 1995.

[14] K. Hayhurst, D. Veerhusen, J. Chilenski, L. Rierson, A Practical Tutorial on Modified Condition/Decision Coverage, Tech. rep., NASA (2001).

[15] NOGUEIRA, Sidney; SAMPAIO, Augusto; MOTA, Alexandre. Test generation from state based use case models: Test generation from state based use case models. Formal Aspects Of Computing. Recife, p. 9-15. 2012.

[16] Felipe Ferreira; Laís Neves; Michelle Silva; BORBA, Paulo. TaRGeT: a Model Based Product Line Testing Tool. In: Tools session, I Brazilian Conference on Software: Theory and Practice (CBSoft 2010), 2010, Salvador. I Congresso Brasileiro de Software (CBSoft 2010), Sessão de Ferramentas, 2010. p. 1-4.