CONEM 2010
VI Congresso Nacional de Engenharia Mecânica

ABCM COMPROMETIDA COM A PROMOÇÃO DO DESENVOLVIMENTO DA ENGENHARIA E DAS CIÊNCIAS MECÂNICAS
ASSOCIAÇÃO BRASILEIRA DE ENGENHARIA E CIÊNCIAS MECÂNICAS

VI CONGRESSO NACIONAL DE ENGENHARIA MECÂNICA
VI NATIONAL CONGRESS OF MECHANICAL ENGINEERING
18 a 21 de agosto de 2010 – Campina Grande – Paraíba - Brasil
August 18 – 21, 2010 – Campina Grande – Paraíba – Brazil

MODELAGEM E VERIFICAÇÃO FORMAL DE SISTEMAS DE TEMPO-REAL EMBARCADOS PARA APLICAÇÕES AEROESPACIAIS

Rhenzo Losso, rhenzo@gmail.com[1]
Emilia Villani, evillani@ita.br[1]
Osamu Saotome, osaotome@ita.br[1]
Luis Carlos Sandorval Góes, goes@ita.br[1]

[1] Instituto Tecnológico de Aeronáutica, R. Mal do Ar Eduardo Gomes, n° 50, Vila das Acacias , 12.228-900, São José dos Campos – SP - Brasil

Resumo: A necessidade de sistemas confiáveis pela indústria aeroespacial faz com que o estudo de sistemas críticos seja amplamente realizado. Um sistema crítico deve ser capaz de ser previsível e livre de falhas que possam causar grandes catástrofes. A aplicação de uma modelagem formal através do método de model checking pode realizar a inspeção de certas propriedades de requisitos estabelecidas. Neste trabalho, um estudo de caso de uma mesa aero-suspensa que simula um satélite com um grau de liberdade é o cenário da modelagem. Tal sistema será modelado embasado nos serviços estabelecidos pela norma da ECSS (European Cooperation for Space Standardization).

Palavras-chave: *Formal Verification, Real-Time, Embedded System, UPPAAL, Aerospace Application;*

1. INTRODUÇÃO

Na área aeroespacial a necessidade de componentes confiáveis faz com que os sistemas críticos sejam amplamente empregados. Tais sistemas são projetados de forma que atendam propriedades, como por exemplo, tempo máximo de resposta. Dentre exemplos de sistemas críticos têm-se os computadores de bordo de aeronaves, onde existe um prazo para se obter a resposta de cálculos e uma atuação do sistema. Caso haja uma violação dessa propriedade de tempo algo catastrófico pode ocorre como conseqüência. Para verificar de antemão se tais propriedades são atendidas ou não pelos sistemas pode-se empregar o uso de modelos que descrevam o comportamento do mesmo.

A modelagem e verificação formal são importantes ferramentas na tentativa em se garantir de maneira segura e antecipada que o sistema irá se comportar de maneira correta. Com métodos matemáticos, existe uma exaustiva varredura das possibilidades de comportamento do sistema, obtendo-se um resultado matemático garantido de que algum evento pode ou não acontecer. No entanto conforme a complexidade do sistema e seu tamanho o número de estados possíveis pode crescer exponencialmente e inviabilizar sua verificação.

Neste trabalho, o objetivo é a modelagem e a verificação de requisitos de tempo-real através de verificação formal com a ferramenta UPPAAL. O experimento o qual será o caso de estudo, consiste de uma mesa aero-suspensa que simula um satélite com um grau de liberdade. Esta mesa possui embarcado um microcontrolador, um giroscópio, um motor com uma roda de reação para efetuar as rotações e um sistema de comunicação via radio para troca de dados e comando entre a mesa e a estação base.

Os componentes que serão modelados são: o sistema operacional, o processo de controle da roda de reação, o processo de verificação de mensagens (verificação de telecomandos). Toda comunicação segue embasada na norma ECSS-70-41 (ECSS, 2003), que estabelece a utilização de pacotes de dados de telemetria e telecomando para sistemas aeroespaciais.

São abordados alguns conceitos teóricos como uma introdução básica a sistemas operacionais, autômatos temporizados e o conceito de telemetria e telecomando. Em seguida, tem-se a apresentação de todos os modelos do sistema, finalizando com uma verificação de requisitos de tempo.

2. TRABALHOS RELACIONADOS

Em Naughton *et al.* (2006), uma modelagem utilizando autômatos temporizados para um escalonador de pacotes com a tecnologia ATM (*Asynchronous Transfer Mode*) é realizada. Durante a modelagem são comparados autômatos temporizados com a modelagem UML *Statechart* na qual são analisados diversos aspectos entre as duas modelagens. O UML é considerado bom para se modelar sistemas complexos, porém sem grandes necessidades de verificação de

tempo e sua representação. Por outro lado, os autômatos temporizados têm uma ótima representação e de boa verificação para tempo, mas apenas é conveniente utilizá-los em sistemas de pequena complexidade. Contudo, não há nenhuma realização de verificação formal de propriedades de tempo.

Já em Wang e Song (2007), é feita uma modelagem utilizando autômatos temporizados com a ferramenta UPPAAL de um sistema de mecânico de plataformas para aplicação em teatros. Depois de realizada a modelagem é feita uma confrontação de dados experimentais com os dados teóricos, sendo que o resultado é obtido é próximo do real e satisfatória as discrepâncias encontradas. Também existe uma verificação formal de certas propriedades como se o sistema ultrapassa valores limites, se estar em um estado implica que outra parte do sistema possa estar em um estado não possível, etc.

A modelagem de sistemas operacionais também é feita em Bang *et al.* (2004), no qual o caso de estudo é o sistema µC /OS II. No artigo Bang especifica e verifica formalmente as funcionalidades e serviços do µC /OS II através da modelagem em *Statechart* com a ferramenta VERSA, porém nenhuma verificação de desempenho é realizada, somente uma verificação se o modelo contém algum *deadlock* é executada.

3. SISTEMAS OPERACIONAIS

Uma definição para sistemas operacionais é que são dispositivos de *software* com o objetivo de manter um sistema computacional complexo, composto de memória, dispositivos de E / S e processadores, funcionando de maneira otimizada (Tanenbaum, 2003). Eles podem ser vistos sob duas abordagens: como uma extensão da máquina ou um gerenciador de recursos.

Na primeira abordagem, no formato *Top-Down*, o sistema operacional estende a máquina e mascara várias dificuldades do usuário em relação à arquitetura do sistema. Dentre elas, podemos citar a complexa operação de acesso a arquivos dentro de um Disco rígido. Tal operação envolve diversos endereços e parâmetros físicos dentro do disco o que torna demasiadamente complexo um acesso a disco para ser feito manualmente. Ao invés de se definir blocos de dados, setores, trilhas e etc, o usuário deve apenas digitar um caminho de acesso e o sistema operacional executa todo o resto, facilitando a interface entre aplicações e recursos do sistema.

Em outra abordagem, no formato *Bottom-Up*, os sistemas operacionais podem ser vistos como gerenciadores de recursos. Essa tarefa serve para que as ações entre processos e o compartilhamento de recursos seja sincronizado. Seja o controle de acesso a uma região critica, a coordenação de um certo recurso (como uma impressora), etc o sistema deve realizar ações de coordenação para que resultados indesejados sejam evitados. Um exemplo pode ser dado na seguinte situação: têm-se vários processos querendo enviar seus dados à impressora. Sem um Sistema operacional, poderíamos ter que determinado instante a linha que impressa seja do processo 1, ou do processo 2 e assim por diante. Com um software que seja capaz de garantir esse compartilhamento de recursos, temos a garantia que primeira será realizada a impressão dos dados de um processo e quando este processo terminar de realizar sua tarefa, o recurso será entregue a outro processo de maneira ordenada.

A necessidade em se tirar o proveito máximo da eficiência do sistema e facilitar o trabalho, foram os principais motivos para o surgimento dos sistemas operacionais. Considerados uns doa primeiros sistemas operacionais, o FMS (*Fortran Monitor System*) e o IBSYS, eram utilizados nos grandes computadores de segunda geração entre os anos de 1955 e 1965. Esses foram os primeiros passos rumo a sistemas mais complexos como o CTSS (*Compatible Time Sharing System*) e o MULTICS (*Multiplexed Information and Computing Service*) que começavam a se aproximar das funções, hoje amplamente implementada pelos sistemas atuais.

Atualmente existe uma ampla variedade de sistemas operacionais, alguns deles voltados a um tipo de aplicação específica. Essas aplicações podem ser, para multiprocessadores, computadores pessoais, *mainframes*, Tempo-Real, sistemas embarcados, etc.

Para este trabalho a aplicação modelada e desenvolvida será de Tempo-Real *Hard* e por isso tem-se como escopo Sistemas Operacionais de Tempo-Real.

3.1. Sistemas Operacionais de Tempo-Real

Algumas aplicações, além de ser necessário que os resultados sejam corretos, existe a demanda por um tempo de resposta máximo. Caso esse tempo de resposta seja violado, o sistema pode degradar ou até mesmo causar uma falha catastrófica. Para isso, um sistema operacional que seja capaz de se previsível (i.e não importa a carga de pico, ou falha, o tempo de resposta permanece constante) é indispensável nesse tipo de aplicação.

Um RTOS (*Real-Time Operating System* em inglês) deve ser capaz de administrar os recursos do sistema computacional como um todo e garantir que os processos que administra não violem seus *deadlines.*

Dentre os sistemas operacionais de Tempo-Real, temos duas classificações: *Hard* e *Soft*. Na primeira classe, são sistemas em que uma falha pode resultar em algo catastrófico, como perda de vidas humanas. Dentre esses sistemas tem-se como exemplo computadores de bordo, sistemas de controle de usinas nucleares, freios ABS, etc. Na outra classe, são sistemas em que uma perda de prazos apenas causa uma degradação, mas não resulta em perdas grandes, estão incluídos nessa classe sistemas voltados para aplicações de áudio e multimídia.

Como exemplo de sistemas operacionais dessa variedade de Sistemas Operacionais têm-se o VxWorks, o QNX e o RTEMS (sistema o qual modelo será embasado no artigo).

3.1.1 RTEMS

O RTEMS (*Real-Time Executive for Multiprocessor Systems*) (RTEMS, 2010) foi criado para ser um sistema operacional capaz de prover alta performance para sistemas embarcados com suas primeiras versões disponíveis para aplicações não militares em meados dos anos 90. Ele pode suportar várias arquiteturas de CPU tais como SPARC, Motorola, ARM, Blackfin, etc. Apesar de ser um sistema voltado a multi-processador tanto homogêneo quanto heterogêneo, o RTEMS tem a capacidade de funcionar em sistemas com um único processador.

Dentre os recursos disponíveis tem-se a pilha de TCP/IP, sistemas de arquivos como FAT e NFS, capacidade de se configurar os recursos do *Kerne*l (como o algoritmo de escalonamento, preempção, tempo de *quantum*). No entanto não possui nenhuma forma de gerenciamento de memória ou processos, tendo apenas um processo com várias *threads* dentro do sistema.

4. AUTÔMATOS TEMPORIZADOS

Uma maneira para se realizar uma modelagem e depois verificar propriedades de maneira formal seria através do uso de autômatos (Aho, 2008). Entre as diversas extensões de autômatos, têm-se os autômatos temporizados (Alur e Dill, 1994). Nesta extensão, tem-se a inserção do tempo por meio de relógios, com isso a gama de sistemas que podem ser modelados é maior. Dentre esses sistemas que se podem modelar, os sistemas de Tempo-Real estão inclusos. Como estes tipos de sistema devem levar em consideração o parâmetro tempo, essa modelagem é adequada para representar o seu comportamento.

A diferença entre autômatos e autômatos temporizados é a inserção de relógios que representam um tempo absoluto. Esses relógios, depois de iniciados, não podem ser mais parados, porém podem ser reiniciados em qualquer ponto do modelo. Além disso, os estados ganham um campo de controle chamado “invariantes”, este campo permite que o sistema fique no estado atual até que essa “invariante” seja violada. As transições ganham um campo chamado “guarda”, que permite que a transição se torne habilitada depois de determinado intervalo de tempo, podendo ou não ser disparada. Caso uma invariante seja violada e nenhuma transição esteja habilitada, o sistema entra em *deadlock*, resultando em uma inconsistência do modelo e uma necessidade de reprojeto do modelo. Nos autômatos temporizados, existem estados especiais que são chamados de comitted, estes estados tem como característica especial não haver a passagem do tempo.

Na figura (1), tem-se um exemplo de autômato temporizado composto de dois estados Com sua invariante na cor roxo e sua guarda em cores verdes, também está representado uma atualização de relógio (no caso um reinicialização) em azul. Neste exemplo, o estado inicial é dado pelo duplo circulo.

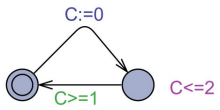

Figura 1:exemplo de autômatos temporizados.

Para realizar a modelagem do sistema deste trabalho, utilizou-se uma ferramenta para modelagem e verificação de autômatos temporizados, o UPPAAL (Behrmann, 2004). Ela foi desenvolvida em conjunto pela Universidade Uppsala na Suécia e pela Universidade de Aalborg, na Dinamarca. Possui uma interface que permite a modelagem e a simulação dos modelos feitos, além de um verificador com o método de *model checking*. O verificador tem como entrada uma proposição em lógica proposicional, o qual verifica se tal propriedade é satisfeita ou não.

5. TELEMETRIA E TELECOMANDO

É uma tecnologia que permite a troca de dados e comandos entre sistemas remotos e suas respectivas bases de operação. Do grego *tele*= remoto e *metron*= medida, telemetria tem como objetivo o envio de dados de uma estação remota para uma base, técnica usada em esportes como Formula 1, corridas de *dragster* e etc. Normalmente, a comunicação entre os sistemas que utilizam essa tecnologia é feita sem fio (wireless).

Uma aplicação de interesse na área aeroespacial é na comunicação de estações solo e satélites / sondas espaciais. Existem vários padrões de utilização de telemetria e telecomando, dentre eles o padrão PUS (*Packet Utilization Standard*) da ECSS (European Cooperation for Space Standardization) (ECSS, 2003), que será utilizado neste trabalho. O princípio de funcionamento é descrito na figura (2). Uma estação de solo envia um pacote contendo um telecomando para o satélite, este processa o comando e responde com um ou mais pacotes de telemetria.

Figura 2: Princípio de funcionamento de Telemetria-Telecomando.

De acordo com este padrão, vídeo e áudio não podem ser considerados pacotes de telecomando ou pacotes de telemetria. Vários serviços, implementados por um ou mais processos, podem ser definidos dentro de um satélite, estas definições dependem do grau de complexidade de construção (se é uma arquitetura distribuída ou centralizada) e dos recursos a bordo do satélite (instrumentação de experimentos). Para uma funcionalidade correta do serviço existe um conjunto mínimo de funcionalidades que deve ser implementado de acordo com a norma, conjunto esse chamado de conjunto mínimo de funcionalidades do serviço. Neste trabalho, o enfoque será no conjunto mínimo de funcionalidade dos serviços de verificação de telecomando (*TC Verification)* e no serviço de gerenciamento de função (*Function Management Service*).

O serviço de verificação de telecomando tem como objeto a verificação estrutural do pacote recebido pelo satélite, além do monitoramento dos mais diversos estágios de execução de um dado telecomando. No entanto, nem todos os telecomandos precisam passar pela verificação em cada estágio distinto. Esta definição se o telecomando deve ser verificado ou não é definida de acordo com a missão em questão. Já o serviço de gerenciamento de função tem como função o envio de parâmetros ou valores para uma determinada aplicação dentro do satélite que não esteja especificada na norma. Neste caso, por exemplo, pode ser um telecomando com uma referência de rotação do eixo do satélite para um controle de atitude de órbita (uma roda de reação), como no experimento a ser modelado.

6. MODELO DO EXPERIMENTO

O experimento a ser modelado consiste em uma mesa aero-suspensa por um mancal de ar, que simula um simples eixo de um satélite em órbita. Esta mesa tem como componentes, uma roda de reação, um microcontrolador com um sistema operacional embarcado e um giroscópio. Uma arquitetura do sistema pode ser vista na figura (3):

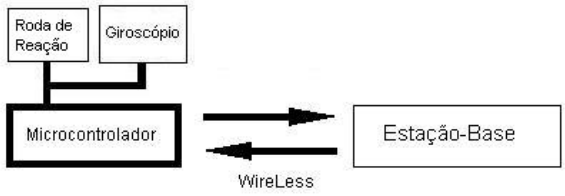

Figura 3:descrição do experimento a ser modelado.

Dentro deste experimento, foram modelados os seguintes processos que estariam dentro do microcontrolador embarcado: o processo de controle da roda de reação, o processo de verificação de telecomando (*TC Verification)* e o sistema operacional baseado no RTEMS.

O primeiro modelo, apresentado pela fig. (4), é o modelo do usuário no qual para cada requisição de rotação da mesa existem duas possíveis respostas: falha ou sucesso. Além disso, existe um estado no modelo no qual é feita uma análise do tempo de resposta. Este estado será usado para verificação formal no calculo dos tempos de resposta.

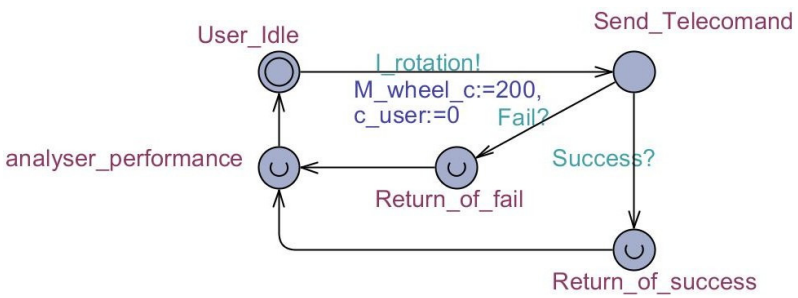

Figura 4: Modelo do comportamento do usuário.

O próximo modelo, fig. (5), representa a estação de solo. Seu estado inicial é demonstrado como a inicialização do sistema e só quando todo os sistemas estiverem operacionais (tanto a estação quanto os processo no satélite) é que se permite o envio de comando ao satélite. Para cada envio de comando, um pacote seguindo o modelo de telecomando da ECSS é gerado e enviado, em seguida espera-se o pacote de telemetria informando falha ou sucesso. Dentro deste

modelo existem diversos caminhos ser testados com erros injetados propositalmente para demonstrar a reação do modelo do processo de verificação de pacotes. O caminho correto é expresso por *service_type*=8, *service_subtype*=1, *APID=001 e checksum*=true (que significa que o *checksum* o qual fora recebido pelo satélite é o mesmo que foi envido pela estação solo). Qualquer caminho diferente deste terá como resposta um pacote de telemetria com uma resposta de falha, desta maneira representado uma possível interferência do meio físico alterando a consistência do pacote.

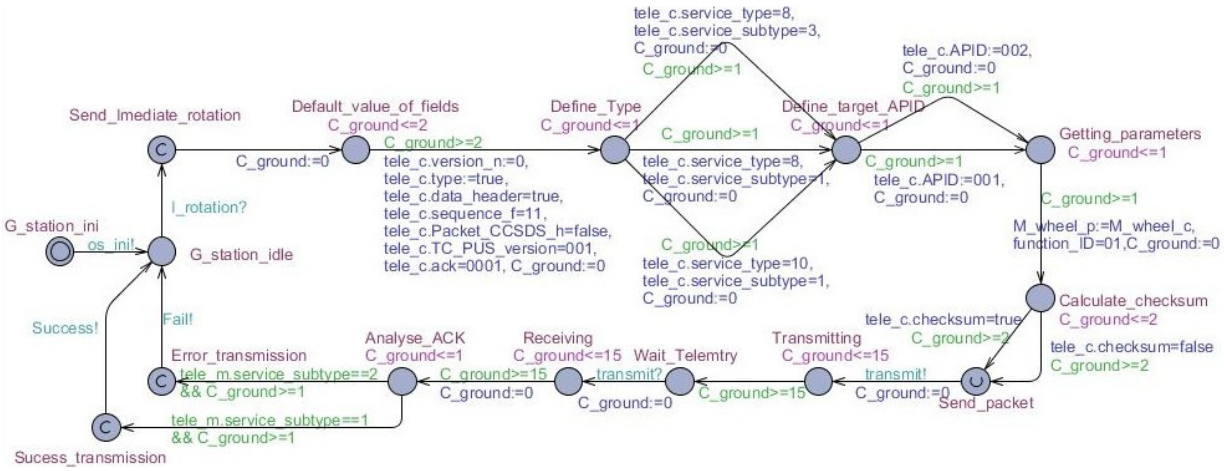

Figura 5: modelo de comportamento da estação Solo.

O terceiro modelo, fig. (6), é o processo da roda de reação. Inicialmente, o processo precisa ser inicializado e então começa a realizar uma série de cálculos para se obter o resultado de atuação correto e envia-lo ao motor da roda de reação em forma de uma saída PWM. O modelo segue o estilo de controle em malha fechada em que primeiro faz-se uma leitura do sensor, no caso o giroscópio embarcado na mesa. Este valor está disposto de forma absoluta, de tal maneira que ira representar o quanto mesa está rotacionada mesmo após realizar várias rotações. Depois se executa um cálculo do erro com um valor de referencia desejado (*setpoint*) e então gera uma saída para o atuador (motor da roda de reação). Essa verificação do valor de referência deve ser feita de forma atômica, pois esta é uma região compartilhada por dois processos distintos (*TC Verification* e controle da roda de reação).

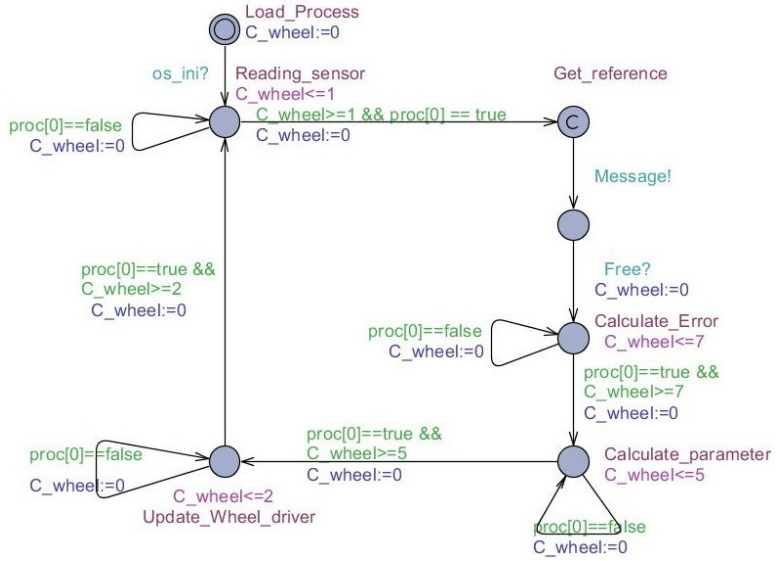

Figura 6: Modelo do processo de controle do motor da roda de reação.

O modelo do sistema operacional, fig (7), foi proposto tomando-se como base uma possível configuração do RTEMS, na qual estabeleceu-se que o algoritmo de escalonamento é o Round-Robin (i.e existe uma fração de tempo para cada processo dentro do CPU) com uma política de filas multi-nível (a fração de tempo será maior ou menor dependendo da prioridade do processo). Essa modelagem foi necessária devido ao fato em se garantir que o processo da roda seja escalonado ao processador a cada certo tempo. Neste caso, têm-se inicialmente apenas dois processos modelados, mas pode ser facilmente extensível a mais.

O sistema operacional também trata a atualização da variável que cuida do valor de referência para qual a mesa deve ser rotacionada. O modelo expressa uma passagem de mensagem através do sistema de fila de mensagens, na qual o processo de verificação de telecomando envia uma mensagem para o processo de controle do motor da roda, utilizando o sistema operacional. Este tipo de operação, garante a atomicidade e a consistência do valor de referencia de rotação da mesa.

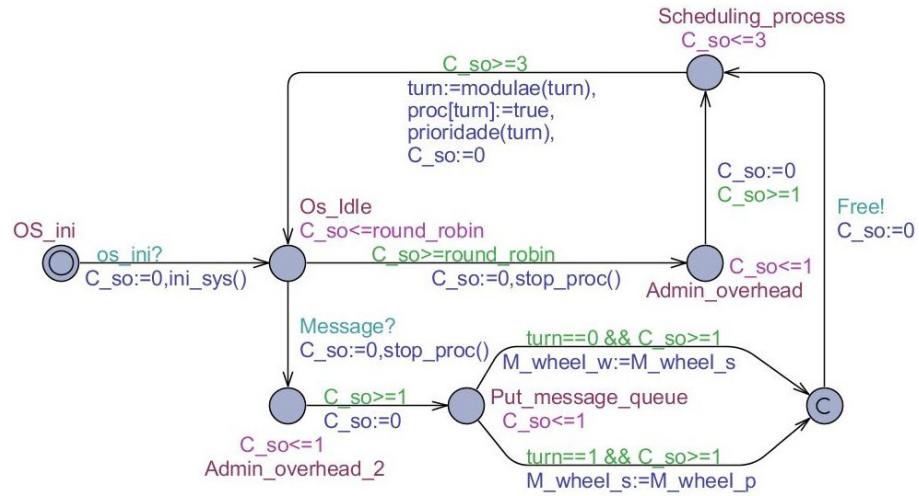

Figura 7: Modelo do sistema operacional baseado em uma configuração do RTEMS.

Dentro desta parte do modelo existem as funções que servem para melhorar a legibilidade e reduzir a poluição que as linhas de código deixaram dentro da figura. Para isso montou-se na tab (1) a demonstração do código de cada uma das funções e o que ela representa no modelo.

Por último, o modelo a ser apresentado, fig (8), é o do processo de verificação de telecomando (*TC verification*) que verifica a estrutura do pacote de telecomando recebido pela mesa. Nem todos os erros foram tratados, apenas exclui-se dos casos de erros quando o pacote não contém todos os bits necessários (assumiu-se que o pacote era perfeito) e quando o processo dentro do satélite era invalido.

Devido a uma certa limitação da modelagem, pode-se se ter casos em que o *quantum* não é aproveitado totalmente. Se o autômato de qualquer um dos processos (roda de reação ou verificação de telecomando) estiver em um dado estado (estado 1) que ao mudar para outro estado (estado 2) viole a quantidade *quantum* restante, resultará que quando o processo for escalonado para o CPU novamente, ele deverá partir do estado que estava anteriormente (estado 2). Com isso todas as operações que foram realizadas no resto *quantum* anterior terão que ser refeitas. Para se evitar isso o caminho do sistema operacional (com exceção da passagem de mensagens entre processos) era sempre a ultima escolha entre os possíveis caminhos.

Tabela 1: Tabela com funções utilizadas no modelo do sistema operacional.

Nome da função	Código Fonte	Objetivo

Ini_sys	void ini_sys()
	{
	proc[0]:=true;
	turn:=0;
	round_robin:=20;
	}
		Inicia o sistema operacional e coloca o processo da roda de reação como sendo o primeiro processo a ser escalonado, com seu tempo de *quantum* de 20ms.

Stop_proc	void stop_proc()
	{
	int i;
	
	for(i=0;i<=(num_proc-1);i++)
	{
	proc[i]:=false;
	}
	}
		Realiza a retirada de todo e qualquer processo que esteja no CPU.

Modulae	int modulae(int m)
	{
	stop_proc();
	m++;
	if(m>(num_proc-1))
	{
	m:=0;
	return m;
	}
	return m;
	}
		Calcula qual processo vai para o escalonador, reiniciando quando atinge o numero máximo de processos.

Prioridade	void prioridade (int m)
	{
	if(m == 0){round_robin:=10;}
	if(m >= 1){round_robin:=5;}
	}
		Alterna o valor de quantum conforme o processo que é escolhido na CPU. (O processo de roda de reação possui maior tempo dentro do CPU)

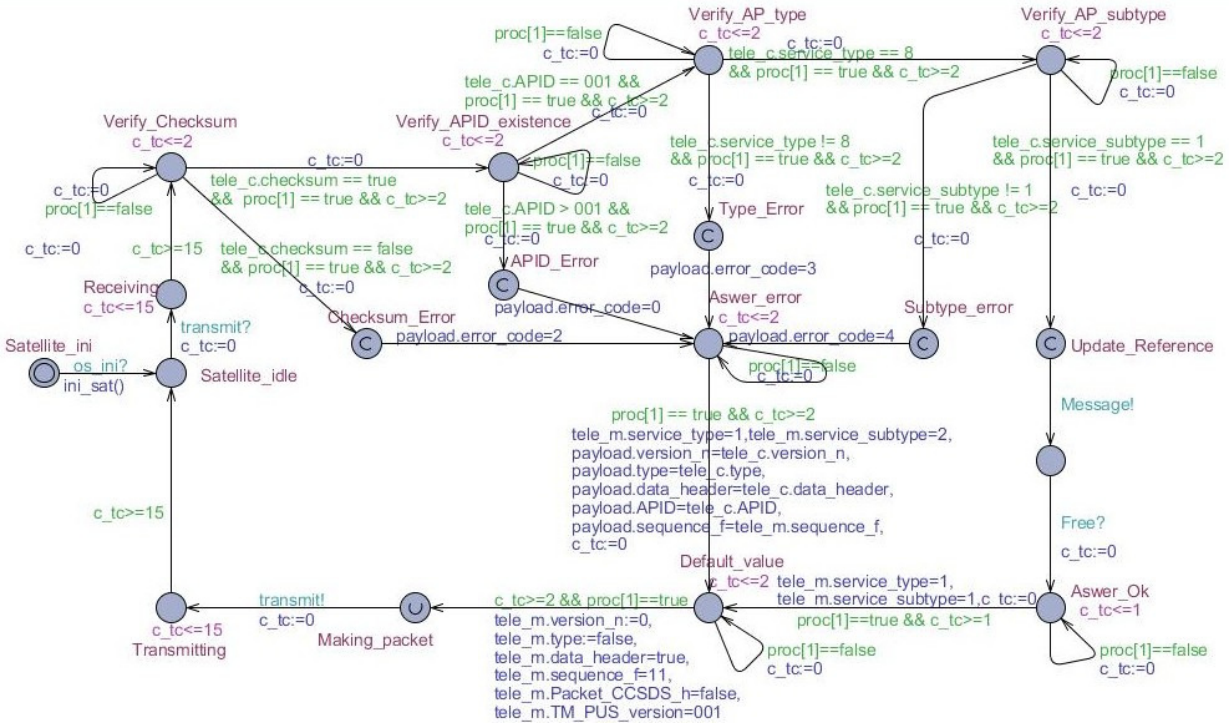

Figura 8:Modelo do processo de verificação de Telecomando.

A única função que é usada no processo de verificação de pacotes é *ini_sat()* que inicializa o valor de referencia do motor da roda de reação para zero graus.

7. VERIFICAÇÃO FORMAL

Com o verificador que implementa o método de *model checking* é possível verificar propriedades do sistema com relação a tempos de resposta de maneira formal. Neste experimento, foram obtidos os tempos de resposta possíveis (baseado no modelo) do tempo total desde quando o telecomando é enviado pelo usuário até quando a resposta retorna da mesa.

Para isso também se variou a quantidade de processos (i.e a carga) dentro do sistema embarcado. No primeiro caso, têm-se apenas os processo de verificação de telecomando e da roda de reação, em seguida foram colocados mais slots vazios na política de escalonamento. Estes slots seriam o *quantum* alocado para outros processos além dos modelados. O objetivo é o estudo do tempo de resposta em aumento da carga do sistema. Na tabela (2), é explicito se o experimento atende o requisito ou não.

Tabela 2: Tempo de resposta em função da quantidade de processos no sistema.

	Prazos de Tempo de Resposta (ms)										
Número de Processos	110	120	130	140	150	160	170	180	190	200	210
2	n	s	s	s	s	s	s	s	s	s	s
3	n	n	n	s	s	s	s	s	s	s	s
4	n	n	n	n	n	n	s	s	s	s	s
5	n	n	n	n	n	n	n	n	n	s	s

8. CONCLUSÃO

A necessidade em se garantir certas propriedades de tempo e comportamento de um sistema faz com que a modelagem se torne uma importante ferramenta. Na área aeroespacial a modelagem de sistemas críticos ajuda a prever o comportamento a priori de um certo componente. Com esse comportamento é possível prever possíveis falhas ou estados indesejáveis que possam ser alcançados, assim evitando que alguma catástrofe.

Com ferramentas como o *software* UPPAAL é possível verificar formalmente certas propriedades e garanti-las. No caso do experimento, verificou-se que no sistema operacional modelado o requisito de tempo máximo de resposta varia de acordo com a carga. Dessa maneira é possível saber de antemão o impacto causado pela inserção de mais processos no tempo de resposta e assim prever o comportamento do sistema mesmo em períodos de picos, garantindo a segurança no seu funcionamento.

Como principal trabalho futuro, destaca-se a validação dos resultados obtidos por meio da comparação com experimentos realizados no sistema real.

9. AGRADECIMENTOS

Os autores agradecem o apoio financeiro do Projeto Pró-Defesa (CAPES) em Segurança de Vôo e Aeronavegabilidade Continuada.

10. REFERÊNCIAS

Aho A. V. , 2008, "Compiladores : Princípios, Técnicas e Ferramentas**",** Ed. Pearson Addison-Wesley, 2° ed, S. Paulo,Brazil.

Alur, R. and Dill, D., 2004, *"*A Theory of Timed Automata*"*. Theoretical Computer Science, Volume 126 , Issue 2 ,April, pp. 183 – 235.

Bang, K.S., Choi, J.Y. and Jang, S.H., 2004, "Formal Specification and Verification of Embedded System with Shared Resources", Proceedings of the 15th IEEE International Workshop on Rapid System Prototyping (RSP'04), IEEE,

Behrmann, G., David, A., Larsen, K., 2004, "A Tutorial on Uppaal". LNCS, Formal Methods for the Design of Real-Time Systems (revised lectures), Italy.

ECSS, 2003, "ECSS 70-41 Telemetry and Telecommand Packet Utilization", http://www.ecss.nl/.

Naughton,M.,McGrath, J. and Heffernan, D., 2006, "Real-Time Software Modelling Using Statecharts and Timed Automata Approaches", ISSC, Dublin Institute of Technology, June 28-30, pp.129-134.

RTEMS, 2010, 25 mar 2010, <www.rtems.org>.

Tanebaum, A.S., 2003, "Sistemas Operacionais Modernos", Ed. Pearson Prentice Hall,2° ed, S.Paulo, Brazil.

Wang, R., Song, X. and Gu, M., 2004, "Modeling and Verification of Program Logic Controllers Using Timed Automata", IET Softw, pp. 127–131.

11. DIREITOS AUTORAIS

MODELING AND FORMAL VERIFICATION OF EMBEDDED REAL-TIME SYSTEM FOR AEROSPACE APPLICATION

Rhenzo Losso, rhenzo@gmail.com[1]
Emilia Villani, evillani@ita.br[1]
Osamu Saotome, osaotome@ita.br[1]
Luis Carlos Sandorval Góes, goes@ita.br[1]

[1] Instituto Tecnológico de Aeronáutica, R. Mal do Ar Eduardo Gomes, n° 50, Vila das Acacias , 12.228-900, São José dos Campos – SP - Brasil

Abstract The need of trustful system in aerospace industry requires an improvement in studying critical systems. A critical system is a system that is predictable and free of fault which can cause great catastrophes. In hardware, something factual and tangible, the fault detection is simple than detecting fault in software, something abstract and intangible. The application of formal modeling using model checking method can inspect of some properties of requirements. In this paper, the case of study is air maintained table that simulates a satellite with one degree of freedom. This system will be model grounded under the services set by the standard ECSS (European Cooperation for Space Standardization

Keywords: *Formal Verification, Real-Time, Embedded System, UPPAAL, Aerospace Application*

RESPONSIBILITY NOTICE