ABORDAGEM SISTEMÁTICA PARA O CONTROLO SEGURO DE SISTEMAS AEROESPACIAIS

Paulo Borges

José Machado

João Ferreira

Universidade do Minho, Departamento de Engª Mecânica, Centro CT2M, PORTUGAL

José Creissac Campos

Universidade do Minho, Departamento de Informática, Centro CCTC, PORTUGAL

Emilia Villani

Instituto Tecnológico de Aeronáutica, Departamento de Engª Mecânica, BRASIL

Abstract

Formal verification of real-time systems behavior of is a complex task, for several reasons. There are multiple works developed in the domain of formal verification of real-time system behavior by model-checking, and various software tools were developed for this purpose. One of the most complexes problems to be solved in the analysis of real-time controllers is the conversion of programming languages controllers in formal languages, for example finite timed automata to be used as inputs of the existing model-checkers. If the methodology of the programming is well developed and known, this task can be greatly facilitated. Moreover, most real-time systems (especially embedded systems that we intend to study) are programmed in C language This article seeks to establish the methodology of creating programs in C code, from SFC specification formalism, taking into account the formal verification of behavior al properties desired for the system, using the Model-Checking technique and the model-checker UPPAAL. A case study is presented to illustrate the methodology presented. These preliminary studies are presented on the context of a research collaboration project being developed by researchers of CT2M, ALGORITMI and CCTC research centers of University of Minho (Portugal) and the Mechanical Engineering Department of Technological Institute of Aeronautics (Brazil).

Keywords*: safe controllers; real-time systems; embedded systems; formal verification; specification formalisms*

Resumo

A verificação formal do comportamento de sistemas tempo-real é uma tarefa complexa, por várias razões. Há múltiplos trabalhos desenvolvidos na área de verificação formal, por model-checking de sistemas tempo-real, sendo que diversos softwares foram desenvolvidos para o efeito. Um dos problemas mais complexos para serem resolvidos na análise de controladores tempo-real é a conversão das linguagens de programação dos controladores nas linguagens formais, por exemplo autómatos finitos temporizados para depois poderem ser verificados formalmente através dos model-checkers existentes. Se a metodologia de elaboração dos programas for bem desenvolvida e conhecida, essa tarefa pode ser muito facilitada. Por outro lado, grande parte dos sistemas tempo-real (principalmente os sistemas

embebidos que pretendemos estudar) é programado em linguagem C. Neste artigo pretende-se estabelecer uma metodologia de criação de programas em código C, a partir do formalismo de especificação SFC, tendo em conta a verificação formal de propriedades comportamentais desejadas para o sistema, utilizando a técnica Model-Checking e o model-checker UPPAAL. Estes estudos preliminares são efectuados no contexto de colaboração entre Investigadores dos centros de investigação CT2M, ALGORITMI e CCTC da Universidade do Minho (Portugal) e do Departamento de Engenharia Mecânica do Instituto Tecnológico de Aeronáutica (Brasil).

Palavras-chave: *controladores seguros; sistemas tempo-real; sistemas embebidos; verificação formal; formalismos de especificação do comando*

1. Introdução

Sistema embarcado tempo real é um sistema que possui características especificas para aplicações especificas sempre associadas a metas temporais, à confiabilidade, segurança, o tamanho e complexidade do sistema de coordenação de tarefas. Estes sistemas podem ser classificados como críticos e não críticos, dependendo das suas especificações, requisitos e aplicações. A distinção entre estes dois tipos é feita segundo a consequência que uma falha poderá causar, por exemplo, sistemas de processamento bancário é um sistema de tempo real não crítico. Já no caso de sistemas críticos, as consequências poderão ser bastante piores, sendo em certos casos devastadoras, podendo envolver vidas ou um prejuízo económico elevado. Podemos enunciar para estes casos falhas nuns sistemas de controlo de voo, sistemas de controlo de planta nuclear, perda de satélites, etc.

Estes sistemas são geralmente complexos e interagem constantemente com o seu ambiente, recebendo inputs de dados, processando-os em tempo real gerando outputs. São submetidos a condições temporais, por vezes, extremas, tendo de gerar uma resposta em tempos, previamente especificados, a reacções externas.

As acções de resposta em sistemas embarcados tempo real seguem uma sequência programada de actividades especificas, com horários fixos e predeterminados. Estes prazos temporais geralmente são cumpridos quando o sistema funciona normalmente, sem falhas de nenhum dos componentes. O problema surge quando algum componente falha implicando a colocação dos sistema num modo de segurança, podendo no caso de os satélites originar a perda do mesmo devido ao delay de resposta entre os componentes, dependendo se o satélite, está ou não, a executar uma manobra critica.

Segundo Stankovic, (1996) “um Sistema de Tempo Real (STR) é aquele na qual a sua correctitude depende não apenas da lógica da computação, mas também do cumprimento do tempo na produção dos resultados.”

De acordo com Shaw, (2001), um sistema de tempo real consiste de dois grandes blocos: o sistema de controlo, que compõe a interface homem-máquina e os sistemas controlados. O sistema de controlo é responsável por responder aos estímulos do ambiente em tempo útil. “É dito reactivo porque sua tarefa primária é responder ou reagir aos sinais do ambiente". Por exemplo, numa fábrica automatizada, o sistema de controlo consiste no(s) computador(es) e interfaces homem-máquina que gerem e coordenam as actividades na fábrica. As interfaces são as portas de comunicação entre o sistema de controlo e o sistema controlado. Geralmente, são sensores, actuadores, receptores de sinais de rádio, entre outros. O sistema controlado é o ambiente com que o computador interage, por exemplo, em linhas de montagem e suas várias peças (Stankovic, 1996). Vale salientar que o processo deve obedecer ao tempo de resposta.”

No campo industrial, em automação, podemos ter sistemas complexos com uma criticidade elevada, sendo as falhas pouco desejadas ou mesmo impossíveis de acontecer. O mesmo se verifica, num satélite, pois uma falha de software pode implicar a perda do mesmo, causando avultados prejuízos. Em face do anteriormente exposto, confiabilidade e segurança em sistemas tempo real são factores muito importantes e problemáticos, sendo por isso necessário que o projecto de um STR seja realizado com cautela e organização, para que possa tornar-se confiável.

Portanto, garantir a previsibilidade de um sistema de tempo-real distribuído envolve uma série de técnicas complementares, que vão desde a especificação e verificação formal do sistema. A verificação neste tipo de sistemas é por natureza complexa.

Técnicas de verificação de algoritmos têm sido exaustivamente estudadas na ciência nos últimos anos (Kurshan, 1994), (Clarke, Grumberg & Peled, 1999), e têm sido aplicadas com sucesso para analisar, por exemplo, circuitos digitais e software (Clarke & Wing, 1996). No contexto do projecto de um controlador, a finalidade óbvia de verificação consiste em verificar se o sistema controlado satisfaz um dado conjunto de requisitos. Várias abordagens que empregam verificação para o projecto do controlador podem ser encontradas na literatura, ver, por exemplo (Tomlin et al. 2003) (Havelund, Larsen, & A-Skou, 1999) (Kapinski & Krogh, 2002) (Stursberg et al. 1998). Eles diferem no que diz respeito à representação do sistema e do controlador, as propriedades que ele pode controlar, e as técnicas computacionais.

Este artigo tem como objectivo propor uma metodologia para o projecto de sistemas embarcados tempo-real que são utilizados nos computadores de Bordo dos Satélites.

O que se pretende é a utilização de técnicas já muito utilizadas e bem testadas na obtenção de controladores seguros utilizados em Automação Industrial e utilizar essas técnicas na obtenção de controladores seguros para sistemas espaciais, sendo que certas especificidades destes sistemas devem ser consideradas (Capítulo 2). A utilização dessas técnicas permitirá a reutilização de códigos de programas, permitirá uma leitura gráfica e estruturada da especificação do comportamento de um sistema deste tipo através da utilização de formalismos para a especificação do comportamento desejado (Capítulo 3). Pretende-se, também, fazer uma analogia entre controladores tempo-real e Autómatos Programáveis Industriais (Capítulo 4) e, finalmente, pretende-se utilizar as técnicas de Simulação e Verificação formal (Capítulo 5) na obtenção de controladores seguros para aplicação nestes sistemas. No Capítulo 6 deste artigo é feita uma representação esquemática da abordagem apresentada e, finalmente, no Capítulo 7 aparecem as conclusões e trabalho futuro.

2. Especificidades dos Sistemas Aeroespaciais

Cada vez mais os satélites têm de ser mais autónomos pelo simples facto que cada vez mais se querer ir mais além, sendo a comunicação e seu comando, em certos casos, insustentável. Para o desenvolvimento dos sistemas embarcados tempo-real nos satélites, com o objectivo de os tornar mais autónomos, estes cada vez mais tendem a se embutirem sistemas mais complexos sujeitos a falhas. De maneira a tornar o seu desenvolvimento mais rápido e económico torna-se necessário reutilizar o código, contendo nele, por vezes, especificações com erros.

Vários acidentes em satélites já ocorreram devido a erros de especificação (Leveson, 2003) ou mesmo devido à falta desta. No 4 de Junho de 1996, Ariane 5, no seu voo inaugural despenhou-se 40 segundo após o inicio da sequência de voo, a uma altitude de 2700 metros. Ficou reconhecido, no relatório, que a principal causa do acidente foi devido a perda completa de orientação e informação de atitude 37s. Após o inicio da ignição do motor. A perda de informação referida foi devido a erros de especificação no desenvolvimento do software do sistema de referência inércial. O software tinha

sido reaproveitado de Ariane 4, e continha pedaços de código desnecessários para o Ariane 5 que também já eram desnecessários para Ariane 4.

No 3 de Abril de 1999, Titan IV B-32/Centaur TC-14/Milstar-3 foi lançado de Cabo Canaveral (Pavlovich, 1999), cujo objectivo era a colocação em orbita geoestacionária. Devido a insuficiência no desenvolvimento do software o satélite perdeu o controlo de atitude desviando-se da sua órbita colocando-se numa órbita incorrecta e inútil.

De realçar, o software desempenha cada vez mais um papel importante em sistemas aeroespaciais, mas por outro lado também tem sido a causa de mais acidentes. Deste facto e já referido a reutilização de código em sistemas espaciais é uma realidade, realidade esta que tem os seu convenientes e inconvenientes. Como foi referido acima, Ariane 5 explodiu por má especificação em código que tinha sido utilizado no satélite anterior, com funções que já existiam em Ariane 4 mas que não eram necessárias em nenhum deles. Boas especificações que incluem requisitos de rastreabilidade e raciocínio são fundamentais para a concepção de sistemas complexos, especialmente naqueles em que parte do código é reutilizado. As especificações devem de ser claras e de fácil percepção por parte dos engenheiros.

Para aumentar a possibilidade de reutilização de código, a informação específica é deixada for a da especificação ou então é incluído mas identificada como potencial necessidade de a alterar.

De referir ainda que, a reutilização do código, apenas é possível porque a maioria dos satélites exigem muitas vezes praticamente as mesmas funções.

2.1 Programação em C

A linguagem de programação C, criada na década de 70 e normalizada pela ANSI em 1983, é uma linguagem de nível médio, em que o código pode ser de baixo e alto nível possibilitando também a manipulação de bits ou instruções de memória. É utilizada para programar micro-controladores e é também utilizada na maioria dos sistemas embarcados. Embora seja uma linguagem muito generalista, é uma linguagem pouco flexível (Gupta, 2000), com muitos problemas de manutenção e sem qualquer estrutura gráfica. Para além destes pontos negativos, tem outro problema, não menos importante, que é a ausência de técnicas de verificação formal assim como a falta de métodos de especificação, apesar de já haver Model-Checkers que aceitam como entrada esta linguagem de programação (Clarke & Kroening, 2004).

A linguagem C, tal como outras, permite pouca estruturação dos programas, pouca reutilização de código e pouca flexibilidade na programação, daí que seja necessário utilizar outros formalismos que auxiliem a obtenção do código para este controladores.

As linguagens de programação estão vocacionadas para a materialização de expressões algébricas tal como as que resultam da síntese de problemas de natureza combinacional ou sequencial. Porém, o que se pretende na engenharia, é a programação de expressões algébricas resultantes da tradução algébrica dos modelos definidos, aquando da utilização de formalismos como, por exemplo, o SFC (EN 2002), os statecharts (Harel, 1987) ou as Redes de Petri(Meda-Campaña & López-Mellado, 2002) entre outras.

3. Controladores Industriais *versus* Controladores Aeroespaciais

Os autómatos programáveis industriais (PLCs) são cada vez mais utilizados nas mais diversas áreas de aplicação, principalmente em áreas críticas de segurança.

À semelhança de um sistema embarcado, um PLC é programado com uma linguagem específica. Os sistemas embarcados e os autómatos têm alguma similaridade, ambos

são controladores lógicos programáveis, reagem com o exterior recebendo inputs e gerando outputs. São também baseados em tecnologias de microprocessadores, de lógica programável com as respectivas limitações, como por exemplo a limitação na frequência de operação interna. É com base nestes princípios que alguns autores (Bayó-Puxan et al. 2008) começam a tratar sistemas embarcados como autómatos desfrutando das ferramentas e suporte ao dispor da automação industrial.

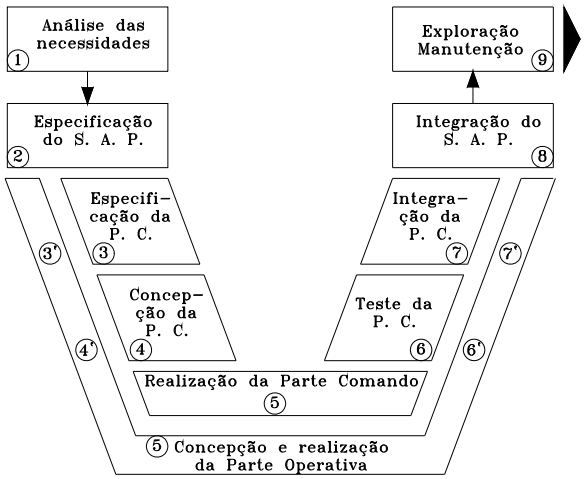
A fim de evitar complexidades adicionais no processo de verificação, podemos encontrar alguma informação sobre técnicas de verificação do código C (Clarke & Kroening, 2004). Há autores (Bayó-Puxan et al. 2008) que dizem que tratam esta conversão de maneira formal, matemática, conseguindo assim eliminar possíveis erros ou ambiguidades humanas. Referem também a possibilidade de poder programar um sistema embarcado como se fosse um PLC, com isto ganharíamos uma facilidade na manutenção destes sistemas assim como a possibilidade de reutilização de código devido à padronização da metodologia.

Os PLCs, devido à sua flexibilidade acabaram por ser muito utilizados em vários campos da indústria. Muitos PLCs são programados utilizando diagramas de Ladder ou linguagens similares. Programar um PLC implica conhecer várias linguagens diferentes. Para efeitos de uniformizar, estandardizar as várias linguagens disponíveis para programar autómatos, a organização IEC criou a norma IEC 61131, que suportava a maioria dos PLCs (Lewis, 1998).

4. Formalismos a utilizar

Qualquer que seja o formalismo utilizado para a especificação do comportamento de um sistema de automação, há várias etapas, no projecto do comendo desse sistema que são incontornáveis, desde o caderno de encargos (que reflectem os requisitos do sistema) passando pela concepção, realização à implementação e exploração de um sistema (figura 1).

Figura 1: Fases da concepção, realização e verificação do comando de um sistema de Automação Industrial



Uma reflexão total das necessidades (objectivos de produção, objectivos de automatização) permite uma definição precisa das especificações funcionais do caderno de encargos de um sistema de automação. Pode-se fazer o estudo paralelo do controlador e do processo físico. Cada um desses estudos leva à existência das

fases de concepção, realização, teste e integração do sistema. Após isso é a fase de colocação em funcionamento.

A integração de todas as fases desse estudo, de uma forma coordenada, e o teste final do conjunto, permitem que o sistema de automação possa entrar na sua fase de exploração.

Numa análise feita aos formalismos já existentes para a modelação do comportamento pretendido para os sistemas aeroespaciais há uns mais ou menos utilizados, pelas suas características: desde os autómatos finitos (Klein, 2005), redes de Petri (Meda-Campaña & López-Mellado, 2002), SFC (EN 2002) Statecharts (Harel, 1987) fica a impressão de que qualquer um destes formalismos pode ser utilizado na especificação do comportamento destes sistemas. A escolha que recair sobre a preferência sobre um ou outro formalismo será apenas por pormenores. Por exemplo, os comportamentos que se conseguem modelar com autómatos finitos ou com redes de Petri serão os mesmos, apenas mudando a complexidade e inteligibilidade do modelo obtido. Além disso o modelo obtido pode ser mais ou menos compacto. O factor "tempo", muito importante na análise de sistemas tempo-real, também pode ser considerado. Assim, a escolha vai recair sobre o formalismo ou formalismos que melhor estiverem adaptados, por exemplo à aplicação das técnicas Simulação e Verificação Formal que se pretende para os controladores obtidos.

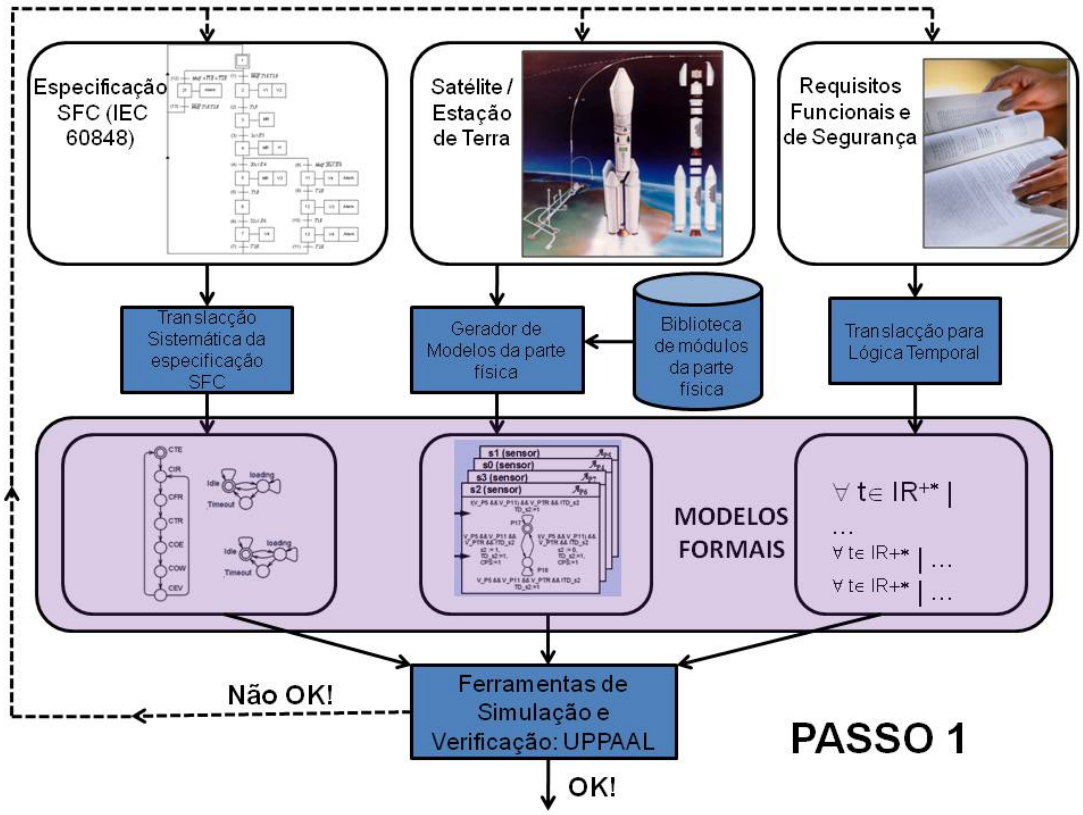
Os autómatos finitos são bastante utilizados para modelação e verificação formal de sistemas, amplamente desejáveis para o projecto de software espacial sendo importante para qualquer processo de controlo. Em geral, o software está intimamente ligado com o sistema que tem de ser controlado. A verificação deste é feita pela construção de um modelo abstracto do sistema, sendo em seguida, verificado se este preenche os requisitos pretendidos. Como há bastantes model-checkers cuja entrada é um modelo de estados, as tarefas de verificação ficariam facilitadas, mas o grande problema da utilização de autómatos finitos é a complexidade dos modelos que, muitas vezes, são necessários para descrever comportamentos mais complexos. Assim, a escolha terá de recair num formalismo com uma maior capacidade de abstracção, que seja de fácil percepção gráfica e que, apesar do seu grau de abstracção esteja suficientemente próximo da implementação, visto que, como anteriormente referido, estes sistemas são programados na linguagem de programação C.

Tendo em conta as características enumeradas no parágrafo anterior, as Redes de Petri, o SFC e os Statecharts estariam em pé de igualdade. Pensamos que o formalismo SFC poderá estar em ligeira vantagem pela sua intuitiva interpretação gráfica e pela normalização a que está sujeito, além de ser possível, também, considerar e modelar o tempo (EN 2002). Além disso, há trabalhos consolidados de translação deste formalismo para a linguagem de programação C (Bayó-Puxan et al. 2008) e também trabalhos consolidados de tradução deste formalismo (Remelhe et al. 2004) (Stursberg & LohmannEngell, 2005) para o model-checker UPPAAL (Gourcuff, 2004), especialmente desenvolvido e concebido para a verificação de sistemas tempo-real. Assim, se for desenvolvida uma especificação para estes sistemas inteiramente em SFC e posteriormente for simulada e verificada formalmente em model-checker apropriado e for garantido que essa especificação será traduzida de forma sistemática para linguagem C poderemos afirmar que o nosso software é um software confiável e seguro. Teremos, também, sempre a possibilidade de verificar esse software, desenvolvido em linguagem C, utilizando model-checkers que aceitam a linguagem C como linguagem de entrada para os mesmos (Bayó-Puxan et al. 2008).

5. Proposição de uma Metodologia Sistemática para o projecto de Sistemas Aeroespaciais

A metodologia proposta neste paper, para a análise de controladores seguros para sistemas aeroespaciais, divide-se em dois passos fundamentais: figuras 2 e 3.

Figura 2: Verificação formal da especificação SFC, considerando modelos da parte física do sistema: passo 1 da abordagem sistemática proposta para a obtenção de controladores confiáveis e seguros para sistemas aeroespaciais



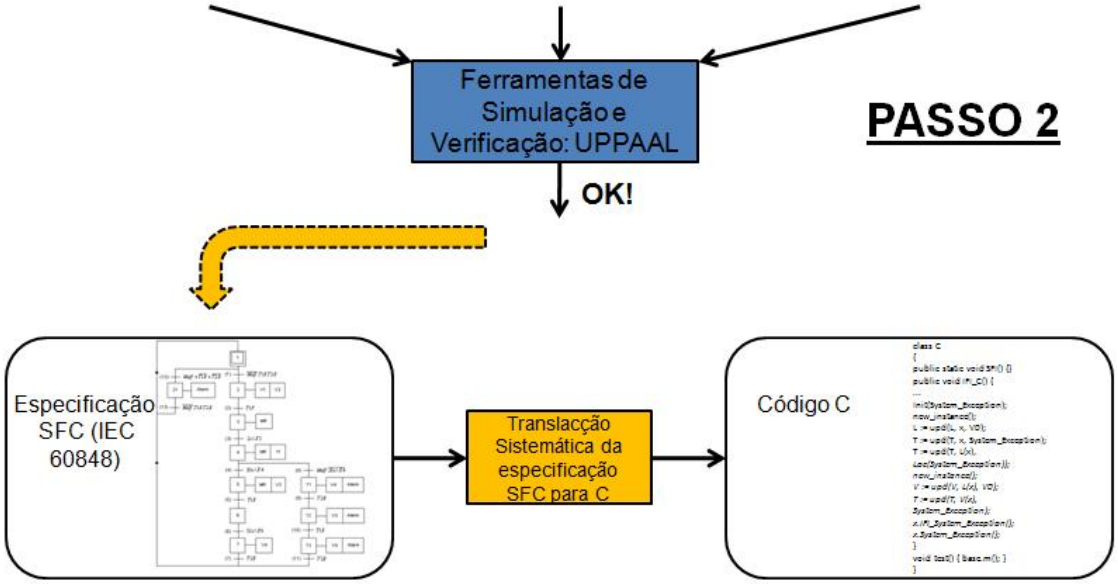
O primeiro passo (figura 2) consiste na verificação formal da especificação SFC considerando um modelo da própria especificação que é traduzida de forma sistemática para autómatos finitos temporizados (Remelhe et al., 2004). Essa verificação formal deve considerar também modelos de comportamento dos satélites e da estação de solo. Se há certas, propriedades de segurança, cuja prova não deve considerar modelos da parte física do sistema há outras, as propriedades de vivacidade, que devem imperiosamente considerar esses modelos (Machado, Denis & Lesage, 2006). As propriedades de comportamento desejado e de segurança devem ser traduzidos em "TimedComputationTreeLogic" (TCTL) (Alur, Courcoubetis & Dill, 1993).

Até que a especificação - através da utilização sucessiva, no Model-Checker UPPAAL, das técnicas Simulação e Verificação Formal – esteja formalmente verificada, deve-se seguir as indicações da figura 2, alterando a especificação até que tal aconteça. Acontece, por vezes que a análise de traços sucessivamente dados pelo model-checker leva a que se possa considerar a hipótese de fazer alterações, também, nos modelos da parte física ou até de certas especificações de funcionamento que se julgavam correctamente formuladas mas que se verifica não estarem correctos.

Depois de a especificação estar formalmente verificada deve ser executado o passo 2 (figura 3) onde, por uma translação sistemática da especificação (Bayó-Puxan et al.

2008) poder-se-á obter um código C com elevados níveis de confiabilidade e segurança.

Figura 3: Translação sistemática da especificação SFC (verificada formalmente no model-checker UPPAAL) para código C: passo 2 da abordagem sistemática proposta para a obtenção de controladores confiáveis e seguros para sistemas aeroespaciais



Desta forma, pensamos que é possível obter elevados níveis de confiabilidade e segurança destes programas. Certo é, porém, que uma posterior verificação formal do código C, propriamente dito, poderá elevar ainda mais esses níveis, mas apesar dos trabalhos existentes nessa área da verificação do código C, ainda há muitas funções do código de difícil verificação formal (Ball, Podelski & Rajamani, 2001).

6. Conclusões e trabalho futuro

Neste paper foi apresentada uma abordagem sistemática para o projecto de controladores seguros para sistemas aeroespaciais. Os trabalhos referentes a este estudo estão em curso de realização e esta abordagem parece promissora, pois consegue aproveitar, da melhor forma, as características dos formalismos e ferramentas de software utilizadas.

Este artigo corresponde aos estudos preliminares efectuados no âmbito de uma colaboração de investigação entre os centros CT2M, ALGORITMI e CCTC da Universidade do Minho, em Portugal, e o Departamento de Engenharia Mecânica do Instituto Tecnológico de Aeronáutica, do Brasil, com a finalidade de serem desenvolvidas e aplicadas técnicas para a obtenção de controladores embutidos tempo-real confiáveis e seguros. Os resultados da aplicação da abordagem apresentada estão a tornar-se satisfatórios, sendo que esses resultados não estão, aqui, descritos nem especificados. Sê-lo-ão noutras publicações futuras.

O facto de termos conseguido abordar estes sistemas espaciais e termos também utilizado formalismos e técnicas de análise conhecidas e que nos garantem a obtenção de software confiável e seguro, deixa-nos esperançados no sucesso dos trabalhos de investigação que estão em curso de realização.

Referências

Alur R., Courcoubetis C., & Dill D. L. (1993). *Model-checking in dense real-time*, Information and Computation, vol. 104, n1, pp. 2-34.

Ball T., Podelski A., & Rajamani, S.K. (2001). *Boolean and cartesian abstractions for model checking C programs*. In T. Margaria & W. Yi (eds.) *LNCS*, (pp. 268–283). Springer Berlin, Heidelberg

Bayó-Puxan, O., Rafecas-Sabaté, J., Gomis-Bellmunt, O., & Bergas-Jané, J., (2008) *A GRAFCET-compiler methodology for C-programmed microcontrollers,* (pp. 55–60) Assembly Automation Emerald Group Publishing Limited

Clarke, E. & Wing, J., (1996, December) *Formal methods: state of the artand future directions,* ACM Comp. Surveys, (vol. 28, no.4, pp 626-643).

Clarke, E., Grumberg, O., & Peled, D., (1999) *Model Checking.* MIT Press.

Clarke, E. M., Grumberg, O., & Peled , D. A., (2000) Model Checking. The MIT Press.

Clarke, E. & Kroening, D., (2003, January) Hardware verification using ANSI-C programs as a reference. In Proceedings of ASP--DAC 2003, (pp 308–311). IEEE Computer Society Press.

Clarke, E., Kroening, D., & Lerda, F., (2004) *A tool for checking ANSI-C programs*. In K. Jensen and A. Podelski, editors, TACAS 2004, (vol 2988 of Lecture Notes in Computer Science, pages 168–176). Springer.

Clarke, E., Kroening, D., Sharygina, N., & Yorav, K., (2004) Predicate abstraction of ANSI–C programs using SAT. (FMSD) *Formal Methods in System Design*, vol 25: (pp. 105–127).

Clarke, E. & Kroening, D., (2006) ANSI-C bounded model checker user manual. Technical report, School of Computer Science, Carnegie Mellon University.

EN 2002 (2002) - European Standard 60848: *GRAFCET specification language for sequential function charts*.

Gourcuff, V., (2004) *Verification of a timed multitask system with UPPAAL*, Memory of DEA, Lurpa, ENS de Cachan.

Gupta, G. (2000), *Reliable software construction: a logic programming based methodology*, Proced. of fith IEEE International Symposium on High Assurance Systems Engineering, IEEE, (pp. 140-1).

Harel, D., (1987) Statecharts: *A visual formalism for complex systems. Science of Computer Programming*, (pp. 231–274)

Havelund, K., Larsen, K., & A-Skou, (1999) *Formal verification of a power controller using the real-time model checker UPPAAL2K*, in Proc.5th AMAST Workshop, (pp. 277-298).

Kandl, S., Kirner, R., & Puschner, P., (2007) *Automated Formal Verification and Testing of C Programs for Embedded Systems* Proc. of the 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'07) 0-7695-2765-5/07.

Kapinski, J. & Krogh, B., (2002) *A new tool for verifying computer controlled systems*. Conf. on Computer-Aided Control System Design, *IEEE* (pp. 98-103).

Klein S., (Juny 2005) isttable; *Fault detection of discrete event systems using an identification approach*, Doctoral Thesis, University of Kaiserslautern, Kaiserslautern.

Kurshan, R., (1994) *Computer-Aided Verification of Coordinating Processes: the Automata-Theoretic Aproach.* Priceton Univ. Press.

Lampérière-Couffin, S., Rossi, O., Roussel, M. & Lesage. J., (1999, August) *Formal validation of PLC programs: A survey*, Proceedings of the ECC'99, n741, 6 pages, Karlsruhe Germany,.

Lampérière-Couffin, S. & Lesage. J., (2000, August) *Formal Verification of the Sequential Part of PLC Programs*, Proceedings of the 5th Work- shop on Discrete Event Systems, WODES'2000, Ghent, Belgium.

Leveson, N.G., (2001, June) Evaluating Accident Models using Recent Aerospace Accidents: Part I. Event-Based Models, MIT ESD Technical Paper, ESD-WP-2002-06,.

Leveson, N.G. (2003), *The role of software in aerospace accidents*, AIAA Journal of Spacecraft and Rockets (in press).

Leveson, N., & Turner, C., (1993, July), *An investigation of the Therac-25 accidents*. Computer, (vol 26, n.7, p.18-41).

Lewis, R.W. (1998), Programming Industrial Control Systems Using IEC 1131-3, The Institution of Electrical Engineers, Londres, (pp. 5-6), Revised edition.

Lions, J.L. (1996), Ariane 501 Failure: Report by the Inquiry Board, 19 July, European Space Agency.

Machado J., Denis B. & Lesage J-J. (2006, July) A generic approach to build plant models for DES verification purposes. *Proc. of Wodes'2006 – 8th Workshop on Discrete Event Sistems*, (pp.10- 12). Ann Arbor, Michigan, USA.

Meda-Campaña, M.E. & López-Mellado E., (2002, December) Incremental synthesis of Petri net models for identification of discrete event systems, *Proceedings of the 41st IEEE Conference on Decision and Control*, (pp.805-810). Las vegas, Nevada USA.

Machado, M. J., Denis, B., Lesage, J-J, Faure, & J-M, Silva, J., (2003, May) Increasing the efficiency of PLC Program Verification using a plant model, *International Conference on Industrial Engineering and Production Management* (IEPM'2003), Porto, Portugal.

Machado, J. M., Denis, B., Lesage, J-J, Faure, J-M, & Silva, J., (2003, November) Model of Mechanism behavior for verification of PLC programs, *17th Edition of the International Congress of Mechanical Engineering* (COBEM'2003), São Paulo, Brazil.

Mertke, T., & Frey, G., (2001, October) Formal Verification of PLC-programs generated from signal interpreted Petri Nets, *Proceedings of the SMC 2001*, Tucson (AZ), (pp. 2700-2705).

Mertke, T. & Menzel, T., (2000) Methods and tools to the verification of safety-related control software, *Proceedings of the IEEE SMC*.

Moody, J.O. & Antsaklis, P.J., *Petri net supervisors for DES with uncontrollable and unobservable transitions*. IEEE Transactions on Automatic Control, 1999. http://citeseer.ist.psu.edu/article/moody98petri.html

Pavlovich, J.G., (1999) *Formal Report of Investigation of the 30 April 1999 Titan IV B/Centaur TC-14/Milstar-3 (B-32) Space Launch Mishap*, U.S. Air Force.

Remelhe, M.P., Lohmann, S., Stursberg, 0., & Engell, S., (2004). Algorithmic Verification of Logic Controllers given as Sequential Function Charts. *IEEE*

International Symposium on Computer Aided Control Systems Design Taipei, Taiwan

Shaw, & Alan C. (2001), *Real-time Systems and Software*, (1a edição), New York.

Stankovic, & John A. (1996), *Real-time and embedded systems*, 28(1). ACM Computing Surveys.

Stursberg, O., Kowalewski, S., Preussig, J., & Treseler, H., (1998) *Block-diagram based modelling and analysis of hybrid processes under discrete control*, J. Europ. des Syst. Automatises, (vol. 32, no. 9-10, pp 1097-1118).

Stursberg, O., Lohmann, S., & Engell, S., (2005). Improving dependability of logic controllers by algorithmic verification *World Congress IFAC*, (Volume 16, Part 1), Czech Republic.

Tomlin, C., Mitchell, I., Bayen, A., & Oishi, M., (2003) Computational techniques for the verification of hybrid systems, *Proc. of the IEEE*, (vol. 91, no. 7,pp. 986-1001).

Correspondencia (Para más información contacte con):

João Ferreira
Universidade do Minho
Departamento de Engª Mecânica, Centro CT2M, PORTUGAL
E-mail : j.f.ferreira@hotmail.com