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 DE UMA PROPOSTA PARA ARQUITETURA DE CONTROLE DE UM EFETUADOR ROBÓTICO BASEADA EM LABVIEW

José Marcos Silva Anjos, jmarcos.anjos@gmail.com[1]
Emilia Villani, evillani@ita.br[1]

[1]Instituto Tecnológico de Aeronáutica (ITA), Pça. Mal. Eduardo Gomes, 50, São José dos Campos, CEP 12228-900

Resumo: *Este artigo apresenta a modelagem e verificação de uma solução para arquitetura de controle de um efetuador robótico para aplicações aeronáuticas. Este efetuador tem como objetivo automatizar os processos de furação e instalação de prendedores na montagem e junção de segmentos de fuselagens aeronáuticas. A solução de arquitetura apresentada é responsável por gerenciar as funcionalidades deste efetuador e foi desenvolvida na linguagem de programação gráfica da plataforma LabView. A modelagem e verificação desta arquitetura são baseadas em autômatos temporizados e utiliza a ferramenta UPPAAL. Este artigo também apresenta os resultados obtidos com o estudo de caso desenvolvido para o módulo de aplicação de selante e instalação de prendedores.*

Palavras-chave: *efetuador robótico, modelagem, verificação, autômatos temporizados, LabView*

1. INTRODUÇÃO

Os robôs industriais são amplamente utilizados nas indústrias de manufaturas automotivas, eletrônicas, farmacêuticas, alimentícias, entre outras como descreve Holland et al (1999). Entretanto, no estudo mostrado por Summers (2005) a penetração destes robôs na indústria aeronáutica ainda não é tão grande.

Se por um lado, a fabricação de peças primárias para construção de aeronaves conta com um dos parques industriais mais modernos da atualidade utilizando-se de máquinas de comando numérico de última geração. Por outro lado, a montagem estrutural das aeronaves, principalmente a partir da junção dos seguimentos de fuselagens, é em sua imensa maioria executada de forma manual. Desde seu início, a montagem estrutural de aeronaves tem optado pelo uso de ferramentais de grande porte e baixa flexibilidade, mas que conseguem garantir precisão na montagem de segmentos de fuselagem de massa relativamente elevada comparada com as demais indústrias onde o uso dos robôs industriais é bastante difundido.

Nas últimas décadas, a crescente pressão global para melhoria de desempenho na cadeia de operações do mercado aeronáutico tem pressionado estas indústrias a melhorarem seus processos industriais. Um estudo recente apresentado por Cibiel et al (2006) elegeu a operação de furação e instalação de prendedores como o melhor processo para ser automatizado utilizando robôs industriais, dentre as diferentes tarefas executadas manualmente na indústria aeronáutica.

Neste contexto este trabalho apresenta uma contribuição para o desenvolvimento de um efetuador robótico que automatiza os processos de furação e instalação de prendedores na montagem e junção de segmentos de fuselagens aeronáuticas.

Os objetivos deste trabalho são a modelagem e a verificação de uma proposta de arquitetura de controle para um efetuador robótico. A solução de arquitetura apresentada é responsável por gerenciar as funcionalidades deste efetuador e foi desenvolvida na linguagem de programação gráfica da plataforma LabView. A modelagem e verificação desta arquitetura são baseadas em autômatos temporizados e utiliza a ferramenta UPPAAL.

Este artigo está organizado da seguinte forma. A Seção 2 discute trabalhos relacionados. A Seção 3 introduz o projeto AME e EFIP bem como apresenta suas arquiteturas de hardware e software. A Seção 4 apresenta um estudo de caso para os módulos de selante e instalação de prendedores com modelos e verificação de propriedades. Por fim, a Seção 5 traz algumas conclusões.

2. TRABALHOS RELACIONADOS – LABVIEW E UPPAAL

Os trabalhos relacionados a este artigo estão divididos em duas classes distintas. A primeira se refere a trabalhos que abordam o desenvolvimento de sistemas de aquisição, testes e controle em LabView. A segunda classe se refere ao uso de autômatos temporizados para a modelagem e verificação de sistemas de tempo-real.

O LabView (Laboratory Virtual Instruments Engineering Workbench) da National Instruments nasceu em 1986. Regazzi et al (2005), o define como uma ferramenta gráfica para uso em laboratórios de instrumentação que teve sua abrangência aumentada e se consolidou como uma linguagem de programação gráfica (linguagem "G") altamente produtiva para construção de sistemas de aquisição de dados, instrumentação e controle, entre outras aplicações. Bell et al (2004) ressalta que o LabView tem sido usado com sucesso por vários anos em aplicações de teste e instrumentação devido à facilidade da programação gráfica e o grande range de funcionalidades para interface direta com instrumentos, sensores e atuadores. Esta conectividade tem permitido ao LabView ser usado para aplicações de controle.

O LabView é amplamente usado na indústria aeronáutica - como mostram Chen (2007), em aplicações de simulação, aquisição e controle em plantas de pesquisas aerodinâmicas e Turley et al (1997) com o desenvolvimento de um sistemas de teste para motores.

No Brasil, os trabalhos acadêmicos que se baseiam no uso de LabView não são muitos. Como exemplos citam-se as aplicações de Silva et al (2004), com o desenvolvimento de uma arquitetura de controle aberta aplicada à máquina ferramenta de alta velocidade e Garcia et al (2003) com um programa computacional de aquisição de dados para avaliação de máquinas agrícolas.

A linguagem de programação gráfica LabView possui duas ferramentas para desenvolvimento de aplicações de sistemas baseados em eventos discretos o "State Transition Diagram" e o "Statechart".

Como mostra Falcon (2006), o "State Transition Diagram" ou simplesmente "State Diagram" disponibiliza um editor para construção da lógica que define a máquina de estados. A desvantagem desta ferramenta é que ela não incorpora hierarquia e modularidade. O que torna esse tipo de arquitetura extremamente grande e difícil de gerenciar quando aplicado a sistemas complexos.

O "Statechart" foi proposto por David Harel em 1987 e se coloca como um método mais avançado, comparado ao "State Diagram", para descrever sistemas reativos e máquinas de estados. Harel (1987) descreve o Statechart como um meio de simplificar a descrição visual de sistemas complexos baseados em eventos discretos. O módulo LabView statechart é compatível com a especificação UML para statecharts.

O trabalho descrito neste artigo foi inicialmente proposto para implementação usando o módulo Statechart do LabView 8.6, no entanto foram encontradas algumas limitações – como dificuldade para se atingir determinismo na aplicação, inconsistência ao se usar transições temporizadas, devido à magnitude do sistema a sub-rotina do statechart possuía tamanho elevado da ordem de 10 MB e lentidão no ambiente de desenvolvimento do módulo statechart.

O UPPAAL é uma ferramenta para modelagem, simulação e verificação de sistemas de tempo real. Nasceu em 1995 pela colaboração entre as universidades Uppsala na Suécia e Aalborg na Dinamarca. Behrmann et al (2004), mostra que esta ferramenta tem sido aplicada com sucesso em estudos que variam de protocolos de comunicação a aplicações de multimídia. Havelund (1999) utiliza a ferramenta UPPAAL na modelagem e verificação formal de um sistema de controle de componentes de áudio e vídeo. Burns (2003) avalia uso de verificação de modelos e autômatos temporizados aplicados a uma célula de produção que de forma semelhante à aplicação tratada neste artigo, possui uma célula de produção com uma seqüência de atividades predeterminadas e também faz interface com um robô industrial.

3. O PROJETO AME E O EFIP

O Projeto AME, sigla para Automação de Montagem Estrutural, representa uma iniciativa da indústria aeronáutica brasileira em parceria com o ITA (Instituto Tecnológico de Aeronáutica) para o desenvolvimento de tecnologia nacional em automação de montagem aeronáutica. Este projeto é suportado pela agencia governamental FINEP (Financiadora de Estudos e Projetos).

Dentre os subprojetos vinculados ao AME está o EFIP (Efetuador de Furação e Inserção de Prendedores). O EFIP tem como objetivo integrar as funcionalidades de posicionamento de um robô industrial, medição do desvio de posicionamento por meio de um sistema de visão, verificação de perpendicularidade para correção da orientação do robô, aplicação da força de clamp necessária tanto para medição da perpendicularidade quanto para permitir a furação, execução da furação, aplicação de selante e inserção de prendedores. O EFIP ainda integra alguns sistemas de segurança e proteção, como uma célula de carga que é acoplada ao EFIP a fim de monitorar em seis eixos a força exercida entre e a fuselagem e o efetuador.

3.1. Módulos do EFIP

Para atender aos requisitos o EFIP foi subdividido em módulos funcionais conforme ilustrado na Figura (1).

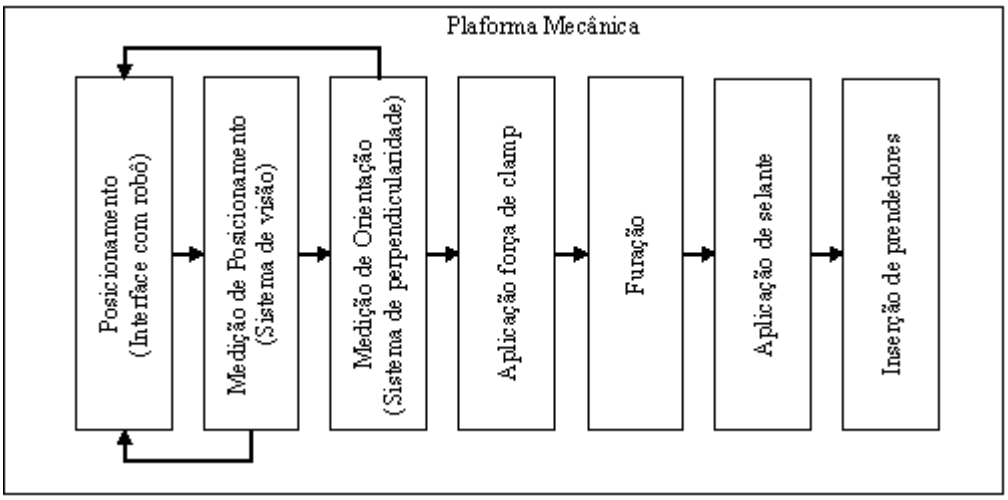

Figura 1. EFIP em módulos funcionais.

Basicamente cada funcionalidade está mapeada em um módulo. O módulo de visão é responsável por determinar a posição do EFIP em relação a um prendedor de referência, chamado de "tack rivet", fixado à fuselagem aeronáutica. É composto por uma câmera industrial CCD, um subsistema de iluminação e também um sensor para medição da distância, entre o efetuador e a fuselagem. O módulo de perpendicularidade é responsável por medir e identificar a orientação do EFIP em relação à fuselagem. O módulo de clamp é responsável por gerar uma força de fixação entre o efetuador e a fuselagem. Esta força é necessária para execução da furação e instalação do prendedor. O módulo de furação é responsável pela usinagem do furo na fuselagem. O módulo de prendedores e selante são responsáveis por receber o prendedor, aplicar selante e inserir o prendedor na fuselagem. A plataforma mecânica é responsável por acomodar e promover o deslocamento necessário aos módulos do efetuador.

3.2. Arquitetura de Hardware

Com a premissa de desenvolver um efetuador industrial, a arquitetura de hardware para o EFIP baseou-se em dispositivos COTS, Commercial Of The Shelf, e de eficácia comprovada no ramo aeronáutico. O hardware de interface para visão, clamp, furação, selante, instalação de prendedores e a célula de carga utiliza uma plataforma da National Instruments o que possibilita integração completa com o software LabView. A interface para posicionamento do robô é realizada utilizando o protocolo OPC.

3.3. Arquitetura de Software

O sistema de controle para o efetuador EFIP deve ser capaz de gerenciar e garantir a integração dos módulos funcionais do EFIP e ainda disponibilizar ao usuário do sistema uma interface gráfica para monitoração e controle.

Considerando que o EFIP pertence a um ambiente de desenvolvimento, o controle deve ser de tal maneira flexível que permita inclusões e desligamentos de componentes tanto de hardware como software, alterando a estratégia de controle do efetuador. Assim, o software de controle deve possuir características de fácil expansão e manutenção, além de permitir a execução de tarefas de forma automática ou manual.

A proposta de arquitetura de software do EFIP desenvolvida e implementada na plataforma LabView 8.6 parte de uma estrutura produtor-consumidor dirigida por fila. Esta estrutura é customizada para ser capaz de capturar eventos da interface gráfica, gerenciar a execução de processos paralelamente, executar seqüências de tarefas predeterminadas, permitir a priorização de processos críticos e garantir determinismo à aplicação.

A Figura (2) ilustra a concepção da arquitetura de software utilizada para desenvolvimento do controle do EFIP. Para possibilitar a visualização, foram omitidos componentes de hardware e "cases" das seqüências internas. Basicamente esta estrutura possui dois *loops* principais temporizados e sincronizados por uma fila. Estes dois *loops* que tem a função de servir como base da arquitetura e configurar o seu determinismo. O *loop* superior é chamado de produtor, onde está hospedado o *case* para gerenciamento e captura de eventos gerados pela interface gráfica. O *loop* inferior é denominado de consumidor, onde está localizado um laço do tipo *for* que gerencia o processamento paralelo de atividades. Alocado internamente a este laço encontra-se a estrutura que hospeda cada um dos processos paralelos A Figura (2) também apresenta em destaque uma descrição dos principais componentes desta arquitetura.

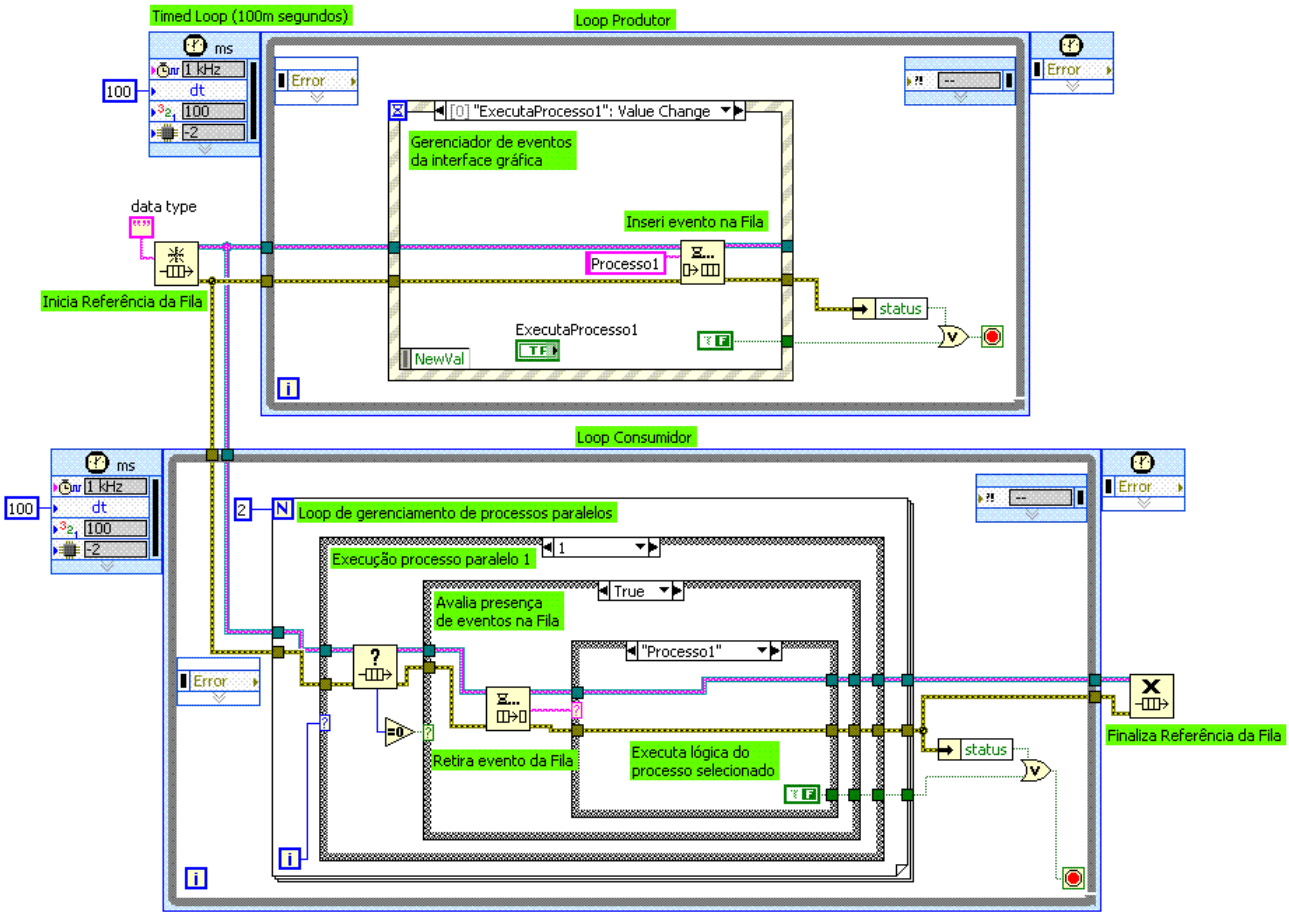

Figura 2. Arquitetura de software desenvolvida para o sistema EFIP

4. ESTUDO DE CASO – INSTALAÇÃO DE PRENDEDORES E APLICAÇÃO DE SELANTE

4.1. Descrição do Processo

O problema abordado neste artigo é a verificação da arquitetura de software proposta por meio da modelagem desta arquitetura em autômatos temporizados.

Como estudo de caso, apresenta-se a modelagem e verificação do módulo de instalação de prendedores e aplicação de selante.

O módulo de instalação de prendedores e aplicação de selante. têm como função receber o prendedor, aplicar selante, inserir e fixar o prendedor em furos previamente realizados pelo módulo de furação. Ele deve receber os prendedores, um a um, por meio de uma mangueira conectada ao EFIP. O prendedor, já com selante aplicado, é inserido na ponta de uma torqueadeira pela plataforma mecânica. A operação do módulo de prendedor pode ser dividida em dois processos: (1) o recebimento do prendedor e aplicação de selante, e (2) a instalação do prendedor.

Neste artigo, como exemplo detalha-se o processo de instalação do prendedor. Quando o módulo de prendedores é posicionado em frente à rótula do EFIP, um cilindro pneumático deve avançar a torqueadeira e posicionar o prendedor no furo. A torqueadeira é acionada, fixando o prendedor na fuselagem.

A Figura (3) mostra o fluxo de tarefas executadas pelo módulo de instalação de prendedores durante a fase de carga do prendedor a ser instalado na fuselagem aeronáutica.

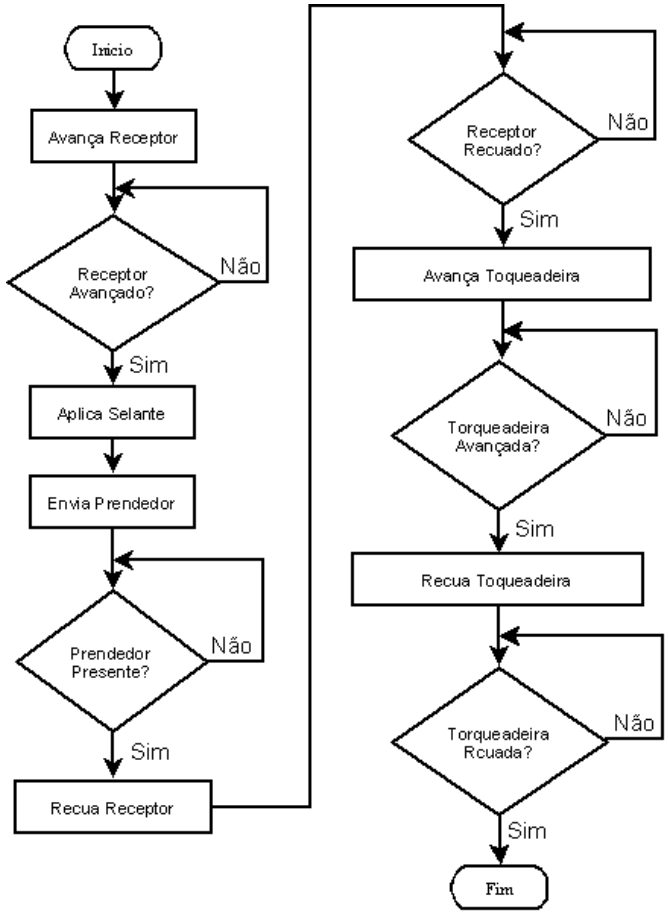

Figura 3. Fluxograma das tarefas de seleção de prendedores.

Da mesma maneira a Figura (4) mostra o fluxograma do conjunto de atividades desempenhado pelo módulo de prendedores durante a etapa de instalação de prendedores.

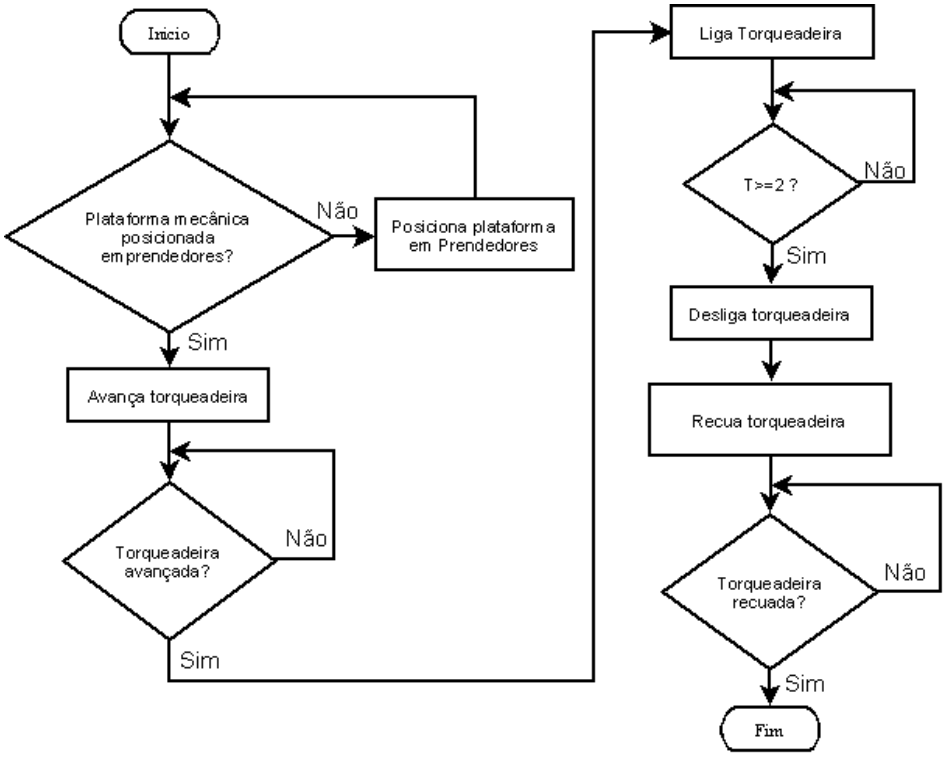

Figura 4. Fluxograma das tarefas de instalação de prendedores.

4.2. Software de Controle

Com base nas seqüências de tarefas executadas pelo módulo de instalação de prendedores e aplicação de selante e utilizando a arquitetura de controle apresentada na Seção 3.3 foi implementada a lógica para a automação destes módulos. O software de controle foi estruturado para receber apenas o módulo em estudo A interface gráfica também foi customizada para o mesmo propósito. Neste estudo de caso o usuário tem as possibilidades de solicitar a execução dos processos automáticos de carga do prendedor, instalação do prendedor ou parar a execução. Logo, estes são os eventos possíveis para estrutura que gerencia os eventos gerados pela interface gráfica. Os *loops* de processamento paralelo foram reduzidos a dois: o primeiro executa a monitoração dos sinais dos sensores e o segundo é responsável por realizar as seqüências automáticas de carga e instalação dos prendedores. O determinismo estipulado para esta aplicação foi fixado em 100 ms. Isso significa que neste intervalo de tempo é executado um ciclo completo do *loop* consumidor. Como os dispositivos físicos utilizados nesta aplicação possuem dinâmica lenta, existe uma estratégia de não permitir que o fluxo de dados fique estagnado em algum ponto do código e desta maneira comprometendo o determinismo da arquitetura.

A Figura (5) mostra a tela de interface gráfica, desenvolvida para este estudo de caso, de onde o usuário visualiza os status dos componentes do sistema e realiza o acionamento dos processos de carga e instalação de prendedores.

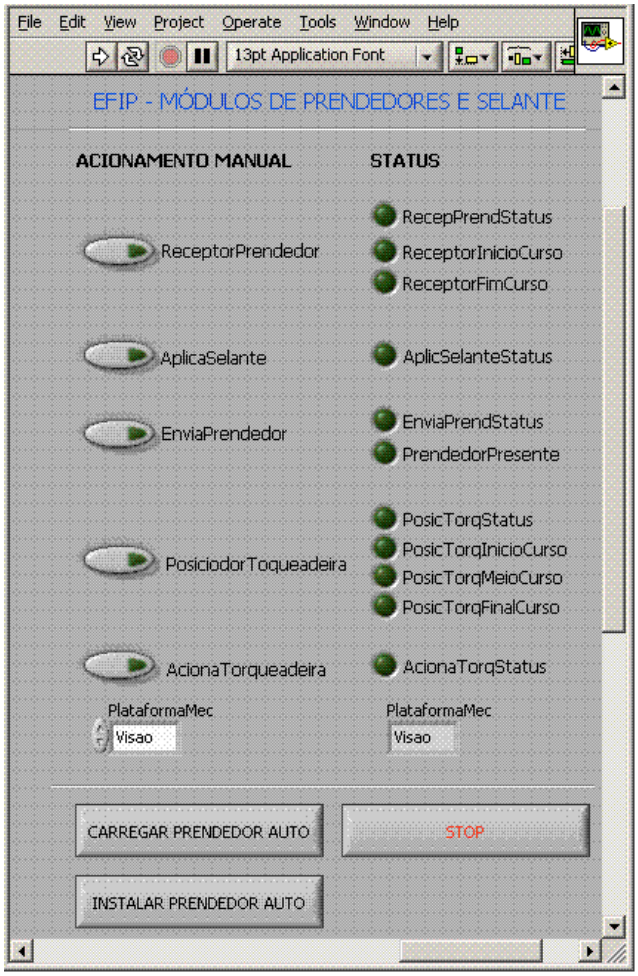

Figura 5. Tela de interface gráfica – Módulos de prendedores e selante

4.3. Modelos UPPAAL

Nos dias atuais os desenvolvedores de softwares não se limitam apenas à codificação de um aplicativo, mas também, com o desenvolvimento de testes que minimizem a possibilidade de erros ou "*bugs*" nos sistemas desenvolvidos. As linguagens de programação textuais de alto nível contam com funcionalidades dedicadas à aplicação de testes em seus códigos. Este contexto, porém, não é verdadeiro para as linguagens de programação gráficas que se apóiam basicamente na habilidade do programador. Com isso, é comum que softwares desenvolvidos em linguagem de programação gráfica apresentem um alto índice de retrabalho e um grau de dificuldade elevado para manutenção. Desta forma, este trabalho apresenta a utilização do ambiente de modelagem, validação e verificação de sistemas de tempo real (UPPAAL) para modelar o comportamento da arquitetura de controle proposta para o EFIP e aplicá-la ao módulo Selante e instalação de prendedores através de um estudo de caso.

Para representar o sistema deste estudo de caso utilizando a ferramenta UPPAAL foram utilizadas 12 classes. Uma classe modela as ações do usuário. Um conjunto de cinco classes modela a arquitetura de controle e outras seis classes modelam os equipamentos físicos do sistema. Variáveis do tipo "boleana" foram usadas para representar os sensores vinculados aos dispositivos físicos e variáveis do tipo inteira para descrever o comportamento da fila. Foi convencionado que cada unidade de tempo no UPPAAL é equivalente a 10 ms. As Figuras a seguir mostram alguns dos modelos desenvolvidos com a ferramenta UPPAAL.

A Figura (6) apresenta o modelo em autômatos temporizados para as ações do usuário

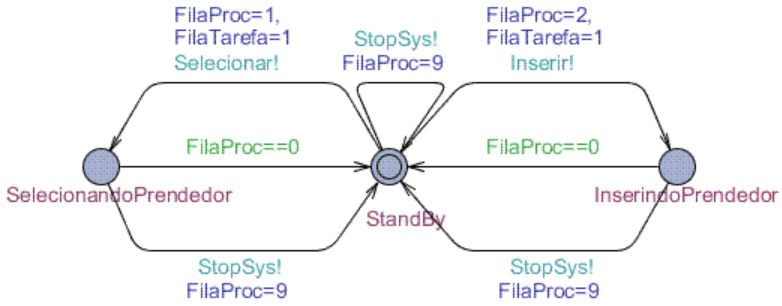

Figura 6. Modelo em UPPAAL das ações do usuário.

A Figura (7) apresenta o modelo em autômatos temporizados da plataforma mecânica. Este dispositivo físico é o responsável por posicionar os módulos do EFIP em frente à rótula, posição central do efetuador, ponto de interface entre o EFIP e a fuselagem aeronáutica. Por questão de espaço os modelos dos outros dispositivos físicos foram omitidos.

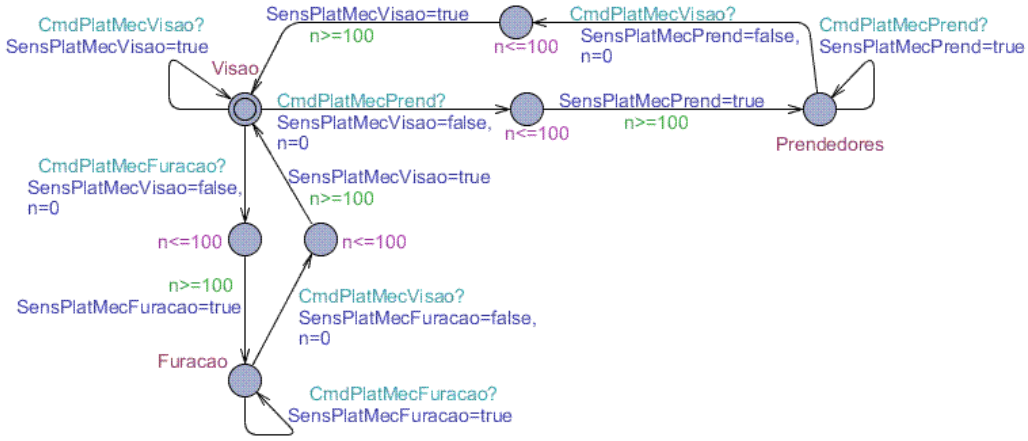

Figura 7. Modelo em UPPAAL da plataforma mecânica.

A Figura (8) apresenta o modelo em autômatos temporizados para representar o comportamento macro da estrutura de controle.

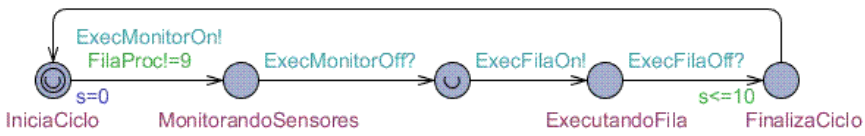

Figura 8. Modelo em UPPAAL da arquitetura

A Figura (9) apresenta o modelo em autômatos temporizados para representar o comportamento do processo paralelo de monitoração dos sensores do módulo em estudo.

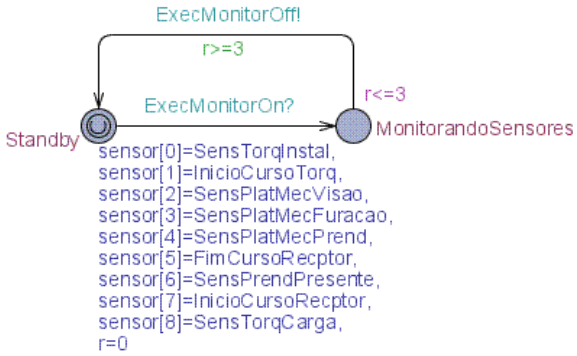

Figura 9. Modelos UPPAAL do monitor dos sensores.

A Figura (10) apresenta o modelo em autômatos temporizados para representar o comportamento da estrutura de captura de eventos da interface gráfica gerado pelo usuário e o direcionamento para fila vinculada ao sistema.

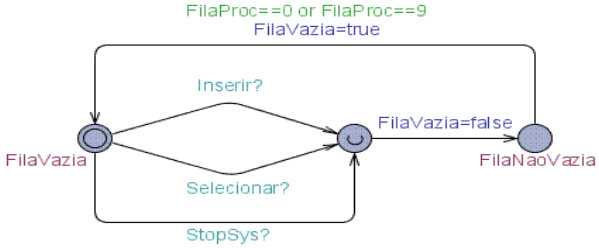

Figura 10. Modelos em UPPAAL da estrutura de fila

A Figura (11) apresenta o modelo em autômatos temporizados para representar o processo de instalação de prendedores.

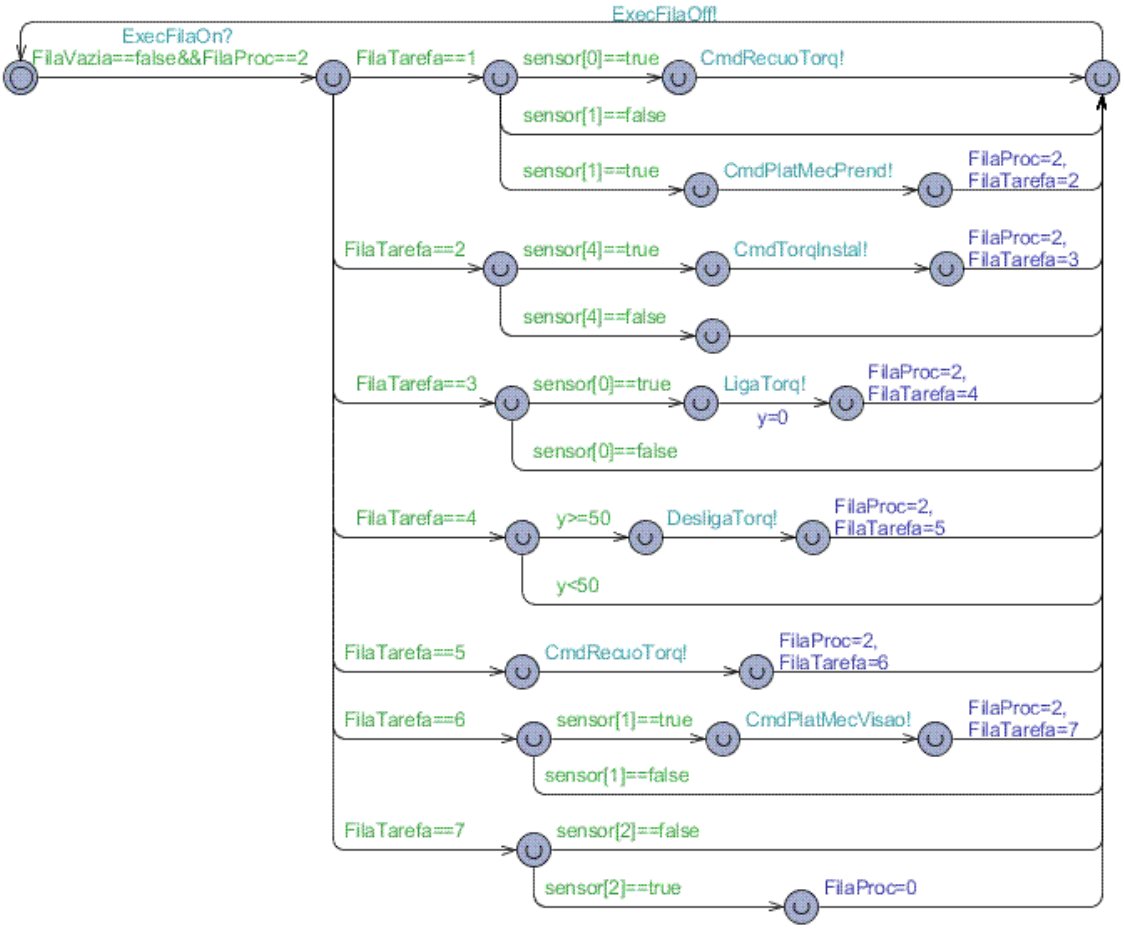

Figura 11. Modelos em UPPAAL para o processo de instalação dos prendedores.

4.4. Verificação de Propriedades

A verificação de propriedades dos modelos funcionais do módulo de instalação de prendedores e aplicação de selante realizada no UPPAAL está direcionada para observar a ausência de *deadlock* e a ocorrência de situações perigosas durante a operação que coloquem em risco a integridade do EFIP e dos usuários do sistema.

A primeira verificação observa a ausência de *deadlock* durante a operação do sistema:

(A[] not deadlock).

Uma condição importante para integridade do sistema deve observar que durante a operação a torqueadeira só deve estar na posição Instalar Prendedor, ou seja, totalmente avançada se e somente se a plataforma mecânica estiver posicionada em Prendedores. Essa condição poderia danificar o efetuador:

A[] Proc5.TorqPosicInstala imply Proc8.Prendedores

O envio de prendedores não deve ser executado se o receptor não estiver posicionado em seu fim de curso. Esta condição faria com que um prendedor fosse lançado dentro do efetuador o que também poderia danificá-lo:

A[] Proc10.PrendCarregado imply Proc9.ReceptorAvancado

Para que o prendedor possa ser carregado na torquedeira a plataforma mecânica deve ser posicionada em visão.

A[] Proc5.TorqPosicCarga imply Proc8.Visao

Para manter o determinismo o gerenciador deve respeitar a condição de tempo

A[] Proc2.FinalizaCiclo imply s<=10

Outras verificações de vivacidade e alcançabilidade utilizando o modelo em UPPAAL também foram executadas para assegurar que tanto os requisitos especificados para a arquitetura quanto o funcionamento do módulo em estudo foram satisfeitos

Por meio da verificação de propriedades realizada com a ferramenta UPPAAL foi possível identificar erros tanto de implementação no LabView quanto no próprio modelo desenvolvido para representá-lo. Dentre estes erros ressaltam aqueles que envolviam condições de segurança, ou seja, se não fossem detectados danificariam o EFIP ou colocariam em risco usuários do sistema.

5. CONCLUSÓES

Este trabalho apresenta uma arquitetura de controle para gerenciar eventos relativos a um efetuador robótico para aplicações aeronáuticas. A arquitetura proposta inclui a interface gráfica com o usuário, a condução de uma seqüência de tarefas do processo do EFIP e a execução de processos paralelos, mantendo o determinismo necessário e possibilitando fácil manutenção e expansão.

O programa de controle, desenvolvido em LabView, é modelado e verificado utilizando autômatos temporizados da ferramenta UPPAAL. Neste sentido considera-se como principal contribuição deste trabalho a análise da viabilidade de aplicar técnicas de verificação formal para um programa desenvolvido em uma linguagem gráfica distinta, como a utilizada pelo LabView. Foi constatado que para a arquitetura proposta a conversão do programa em LabView para autômatos é possível e relativamente simples, permitindo que os erros detectados pela ferramenta UPPAAL sejam corrigidos no modelo em autômatos e que as correções sejam mapeadas no programa original.

6. AGRADECIMENTOS

Os autores agradecem aos órgãos governamentais FINEP, CAPES, CNPq e FAPESP pelos auxílios financeiros. Nossos agradecimentos também a Samir Raad Boutros, Engenheiro de Desenvolvimento do setor aeronáutico, que desenvolveu a versão inicial da arquitetura de controle mostrada neste artigo.

7. REFERÊNCIAS

Behrmann, G., David, A. and Larsen, K.G., 2004, “A tutorial on uppaal, formal methods for the design of real-time systems”, SFM-RT Springer-Verlag, pp. 200-236.

Bell, I., Falcon, J., Limroth, J. and Robinson, K., 2004, “Integration of hardware into the labview environment for rapid prototyping and the development of control design applications” UKACC Control Mini Symposia Publication, pp.79-81

Bruns, A., 2003, “How to Verify a Safe Real-Time System: The Application of Model Checking and a Timed Automata to the Production Cell Case Study”, Real - Time System Journal, Vol 24, No 2, Netherland, pp. 135-151.

Chen, M., 2007, “Object Oriented Programming in LabVIEW for Acquisition and Control Systems at the Aerodynamics Laboratory of the National Research Council of Canada”, 22nd International Congress on Instrumentation in Aerospace Simulation Facilities - ICIASF, Pacif Grove, Canada, pp. 1-6.

Cibiel, C. and Prat, P., 2006, “Automation for the Assembly of the Bottom Wing Panels on Stringers for the A320”, SAE Transactions 2006-01-3143.

Falcon, J.S., 2006, “LabVIEW State Diagram Toolkit for the Design and Implementation of Discrete-Event Systems”, Proceedings of the 8th International Workshop on Discrete Event Systems, Ann Arbor, Michigan, USA, pp. 469-470.

Garcia, R.F., Queiroz, D.M., Miyagaki, O.H. and Pinto, F.A.C, 2003, "Programa computacional para aquisição de dados para avaliação de máquinas agrícolas" Revista Brasileira de Engenharia Agrícola e Ambiental, Vol. 7, No. 2, pp. 375-381,

Harel, D., 1987, "Statecharts: A Visual Formalism for Complex Systems", Science of Computer Programming, Vol. 8, Issue 3 North-Holland, pp. 231-274.

Havelund, K., Larsen, K.G. and Skou, A., 1999, "Formal Verification of a Power Controller Using the Real-Time Model Checker UPPAAL", BRICS Report Series - Basic Research in Computer Science, Denmark, pp. 1-26.

Holland, S.W. and Nof, S.Y., 1999, "Handbook of industrial robotics: Emerging Trends and Industry Needs", Ed. John Wiley & Sons, 2 nd edition, New York, USA, 1348p.

Regazzi, R.D., Pereira, P.S. and Silva, M.F., 2005, "Soluções Práticas de Instrumentação e Automação – Utilizando a Programação Gráfica LabView", Rio de Janeiro, Brasil, 456p.

Silva, E.J., Biffi; M. and Oliveira, J.F.G., 2004, "The development of an open architecture control system for CBN high speed grinding", J. Braz. Soc. Mech. Sci. & Eng., Vol. 26, No.1, Rio de Janeiro, Brazil.

Summers, M., 2005, "Robot Capability Test and Development of Industrial Robot Positioning System for the Aerospace Industry", SAE Transactions Vol. 114, Part. 1, pp. 1108-1118.

Turley, P., and Wright, M., 1997, "Developing engine test software in LabVIEW", IEEE Autotestcon Proceedings", pp. 575-579.

8. DIREITOS AUTORAIS

MODELING AND VERIFICATION OF A PROPOSE FOR THE CONTROL ARCHITECTURE OF A ROBOTIC END-EFFECTOR BASED ON LABVIEW

José Marcos Silva Anjos, jmarcos.anjos@gmail.com [1]
Emilia Villani, evillani@ita.br [1]

[1]Aeronautics Technological Institute (ITA), Pça. Mal. Eduardo Gomes, 50, São José dos Campos-SP, Zip Code 12228-900

Abstract. *This paper presents modeling and verification of a solution to the control architecture for a robotic end-effector used in aeronautic applications. The aim of this end-effector is to automate drilling and fastener installation processes in the junction assembly of aeronautic fuselage segments. The control architecture showed herein is responsible for managing the end-effector functions and was developed in LabView, a graphical programming language. Modeling and verification of this control architecture are based on timed automata using the Rea-Time Model Checker UPPAAL. This paper also presents the results of a case study developed to the sealant and fastener installation module.*

Keywords*: robotic end-effector, modeling, verification, timed automata, LabView*