# UMA METODOLOGIA PARA DESENVOLVIMENTO DE SISTEMAS EMBARCADOS CRÍTICOS COM VISTAS A CERTIFICAÇÃO

Onofre Trindade Júnior*, Rosana T. Vaccare Braga*, Luciano de Oliveira Neris†, Kalinka R L J Castelo Branco*

*Departamento de Sistemas de Computação - Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
Av. do Trabalhador Sãocarlense, 400 – São Carlos – SP – CEP 13560-970

†AGX Tecnologia Ltda – São Carlos – SP

Emails: otjunior@icmc.usp.br, rtvb@icmc.usp.br, luciano.neris@agx.com.br, kalinka@icmc.usp.br

**Abstract**— The development of embedded systems presents some peculiarities, such as low-level communication with hardware devices, uninterrupted functioning for many days, hostile environment and limitations of resources like memory, processing and, especially, energy consumption. These systems require specific development methodologies, targeting mainly the development of products with certified quality. This paper presents the methodology used in the development of an embedded system, used for guidance and control of an unmanned aircraft. The methodology was adapted from existing methodologies, aiming at including activities that help to improve the quality of the process and of the final products. The presented case study is quite complex, and represents a real multidisciplinary application, which contributes to a real evaluation of the proposed methodology. The steps regarding modeling and automatic code generation of the methodology were already implemented, and preliminary results indicate the adequacy of the methodology according to the goals established, notably regarding the quality of the software to its certification standards in each case.

**Keywords**— UAV, embedded systems, development methodologies, safe-critical systems, certification standards.

**Resumo**— O desenvolvimento de sistemas embarcados possui características peculiares, tais como comunicação em baixo nível com dispositivos de hardware, funcionamento ininterrupto por muitos dias, ambientes hostis e limitações de recursos, como memória, processamento e, principalmente, consumo de energia. Esses sistemas requerem metodologias de desenvolvimento específicas, visando principalmente o desenvolvimento de produtos com qualidade certificada. Neste artigo, apresenta-se a metodologia utilizada no desenvolvimento de um sistema embarcado utilizado para guiamento e controle de uma aeronave não tripulada. A metodologia foi adaptada a partir de metodologias existentes, visando incluir atividades que auxiliem na qualidade do processo e do produto final. O estudo de caso apresentado é bastante complexo, representando uma aplicação real e multidisciplinar, o que contribui para uma avaliação da metodologia proposta. As fases de modelagem e geração automática de código já foram implementadas e os resultados preliminares indicam até aqui a adequabilidade da metodologia quanto aos aspectos pretendidos, notadamente quanto à qualidade do software produzido visando sua certificação segundos normas aplicáveis.

**Keywords**— VANT, sistemas embarcados, metodologias de desenvolvimento, sistemas críticos, normas de certificação.

## 1 Introdução

Sistemas embarcados (Douglass, 2004; Lavagno et al., 2003; OMG, 2009) são módulos computacionais integrados a equipamentos e dispositivos físicos, que realizam um conjunto de tarefas predefinidas, normalmente com requisitos específicos. Em geral atuam em monitoração e controle em tempo real, como é o caso de sistemas complexos como robôs, veículos autônomos e equipamentos médicos. Esses sistemas são dedicados a tarefas específicas e por meio de técnicas de engenharia pode-se otimizar o seu projeto reduzindo tamanho, recursos computacionais e custo do produto. Sistemas embarcados são considerados críticos (ou voltados para aplicações críticas), quando eles podem colocar vidas humanas ou instalações de alto valor em risco, no caso de falha. Em algumas aplicações, como por exemplo, na aviação, sistemas embarcados críticos devem apresentar taxas de falhas tão baixas como uma falha grave a cada $10^5$ até $10^9$ horas de operação.

A construção de software para sistemas embarcados é normalmente mais complexa do que para outros sistemas computacionais. Em tais sistemas, o software tem que se comunicar em baixo nível com dispositivos de hardware, deve funcionar por dias e até anos sem parar mesmo em ambientes hostis (alta temperatura, umidade, vibração) (Douglass, 2004). Além disso, há as limitações de recursos, como memória e processamento, sendo necessárias soluções para otimização dos programas, o que não ocorre em outros sistemas computacionais que dispõem de recursos mais abundantes e não têm que atender requisitos de tempo real.

Sistemas embarcados críticos devem seguir normas elaboradas por associações de empresas, órgãos governamentais, como por exemplo a RTCA (*Radio Techical Commission for Aeronautics*). Essas normas são importantes para garantir

a confiabilidade e adequabilidade desse tipo de sistema, e mais que isso, equipamentos críticos não podem ser comercializados e aplicados sem a sua homologação por essas normas. Não basta, portanto, projetar e construir um sistema para aplicação crítica que funcione. É preciso que todo o projeto e desenvolvimento sejam baseados em normas e metodologias que tornem esse equipamento comercializável, e, muitas vezes, esse aspecto é mais difícil de ser cumprido do que a técnica envolvida na especificação e implementação do sistema.

O desenvolvimento de sistemas embarcados demanda, dessa forma, uma metodologia de engenharia de software diferenciada da adotada para sistemas de informação (Hugues et al., 2008). Com a criação do INCT-SEC, (Instituto Nacional de Tecnologia em Sistemas Embarcados Críticos), iniciou-se a investigação de metodologias de desenvolvimento que possam atender não somente as necessidades particulares dessa classe de sistemas como também as necessidades específicas das normas e processos de certificação dos produtos gerados. Neste artigo, apresenta-se como estudo de caso o desenvolvimento do Tiriba, uma aeronave de baixo custo e propulsão elétrica, totalmente autônoma, que está servindo para a validação da metodologia proposta.

Inicialmente estão sendo implantadas as fases de análise de requisitos, modelagem, simulação, geração automática de código e testes de integração em bancada. Pretende-se gerar código que possa ser certificado, provando-se a sua correção a partir dos testes efetuados em modelos e a utilização de ferramentas e código básico previamente certificados. A próxima etapa da metodologia a ser abordada é a geração de casos de teste. Esses testes serão empregados durante duas fases do desenvolvimento: na simulação do modelo e nos testes em bancada baseados em um sistema do tipo *hardware in the loop*.

O restante do artigo está organizado da seguinte forma: a Seção 2 apresenta alguns trabalhos relacionados, a Seção 3 apresenta a metodologia proposta, a Seção 4 o estudo de caso e a Seção 5 alguns resultados preliminares obtidos na aplicação da metodologia. Finalmente, na Seção 6 são apresentadas as conclusões e trabalhos futuros.

## 2 Trabalhos Relacionados

Há vários processos de desenvolvimento de software já estabelecidos, como por exemplo o Processo Unificado (Kruchten, 2003) e a Programação Extrema (Beck, 2000), que são predominantemente apoiados pela notação UML (*Unified Modeling Language*). Eles fornecem diretrizes para o desenvolvimento disciplinado de software e já estão bem aceitos pela comunidade, havendo disponibilidade de diversas ferramentas e ambientes para dar suporte à sua utilização, bem como literatura amadurecida e bem conhecida.

No entanto, o desenvolvimento de sistemas embarcados, que anteriormente era visto como predominantemente voltado ao hardware, tem crescido de forma visível, principalmente no tocante ao desenvolvimento do software envolvido. Já os processos que apóiam o desenvolvimento de software não têm acompanhado a demanda por essa classe de sistemas, obrigando os desenvolvedores a realizarem adaptações nos processos tradicionais, o que nem sempre é apropriado. Existem alguns trabalhos nesse sentido, como por exemplo, o uso de UML voltado a aplicações embarcadas (Douglass, 2004; Lavagno et al., 2003; OMG, 2009; Wehrmeister et al., 2005) e ferramentas para projeto de sistemas embarcados baseadas em *model-driven engineering* (Nascimento et al., 2007).

A geração automática de código é outro recurso que tem sido utilizado no desenvolvimento de sistemas embarcados (Stahl and Volter, 2006), porém esse uso tem ocorrido principalmente na geração da lógica da aplicação a partir de modelos (por exemplo *statecharts* ou diagramas de fluxos de sinais). Além disso, têm sido usados *frameworks* específicos para implementar a infraestrutura técnica de sistemas embarcados, como por exemplo (SCADE, 2009; Microsoft, 2009; Mathworks, 2009). Todos esses recursos também têm sido utilizados neste trabalho, mas investiga-se o uso de geradores em outras partes do sistema. Uma das vantagens do uso de geradores, especialmente se uma arquitetura de linha de produtos for adotada, é que o software conterá apenas as partes realmente necessárias para o produto específico, ao contrário de *frameworks*, que levam consigo a implementação toda, causando problemas de espaço, notadamente um problema crítico em sistemas embarcados.

Como exemplo de trabalho relacionado, a Figura 1 mostra o ciclo de vida da metodologia SEEP para o desenvolvimento de sistemas embarcados (Wehrmeister et al., 2005), em que oito etapas são consideradas: modelagem de alto nível, exploração do sistema, exploração arquitetural, estimativas, compilação do software e geração do sistema operacional de tempo real, síntese da comunicação, síntese da micro-arquitetura e planejamento dos testes. Em algumas das etapas são realizadas atividades de validação, para garantia de qualidade antes de prosseguir o ciclo. Assim como na metodologia SEEP, a metodologia proposta neste trabalho preocupa-se com a modelagem de alto nível o mais independente possível de decisões que comprometam a arquitetura do sistema, como por exemplo que módulos serão implementados em hardware ou software. No entanto, a metodologia proposta coloca em destaque a qualidade dos artefatos obtidos, com vistas a certificação do software.

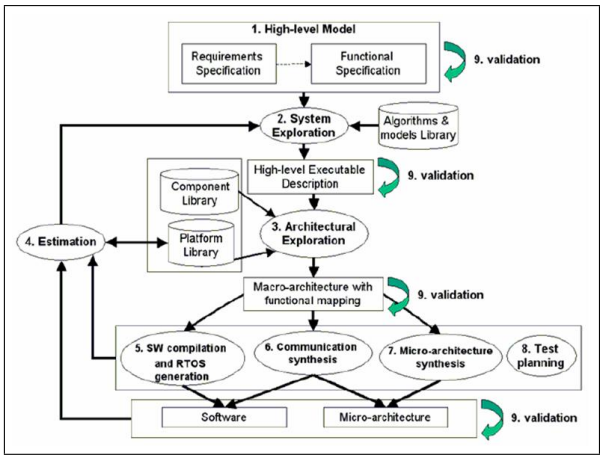


Figura 1: Metodologia SEEP para sistemas embarcados (Wehrmeister et al, 2005)

## 3 Metodologia Proposta

Na definição da metodologia proposta consideraram-se as seguintes premissas:

- O ciclo de vida da metodologia deve ser baseado em uma sequência de fases separadas por fases de validação rígidas, evitando-se ao máximo a revisão de uma fase anterior, como é o caso do desenvolvimento iterativo ou em espiral, bastante utilizados no desenvolvimento de sistemas convencionais;
- Maximizar o uso de ferramentas para modelagem, simulação e geração automática de código, visando diminuir o tempo de desenvolvimento e a possibilidade de erros, facilitando a documentação e os processos de certificação do código gerado, principalmente com o uso de ferramentas previamente certificadas;
- Maximizar a reutilização de código com o uso de módulos previamente testados e certificados, visando não somente a diminuição do tempo de desenvolvimento como também o desenvolvimento de famílias de produtos obtidas pela reorganização dos blocos básicos e seu mapeamento em novas arquiteturas de hardware;
- Ponderar o uso de programação e/ou desenvolvimento orientados a objetos. Apesar das vantagens da orientação a objetos, sua aceitação no domínio do software embarcado nem sempre é recomendada, principalmente pelo custo extra em termos de ciclos e área de memória de programa e dados, como apontado por diversas pesquisas (Chatzigeorgiou and Stephanides, 2002; Bhakthavatsalam and Edwards, 2002)

Deve-se ter em mente que os processos de certificação, como aqueles descritos na norma DO-178B (RTCA, 2001), são fortemente calcados na documentação dos processos de desenvolvimento de software e na quantidade e qualidade dos testes utilizados na sua validação. Também deve ser notado que para cada produto na classe de sistemas embarcados críticos deve-se atender um conjunto específico de normas, que depende não somente da área de aplicação do produto como também do cliente final e o país onde o produto será utilizado.

As fases da metodologia proposta podem ser visualizadas na Figura 2 e são descritas nas etapas que seguem:

1. Análise de requisitos funcionais e não funcionais: descrição textual em formulário específico dos requisitos. Essa etapa deve ser conduzida pelo cliente final, com assessoria da equipe de desenvolvimento.
2. Validação dos requisitos: montam-se matrizes de requisitos que devem ser validadas por desenvolvedores e engenheiros dos sistemas onde serão utilizados os módulos embarcados em desenvolvimento. Deve-se notar que essas matrizes devem servir de base para a elaboração das rotinas de teste, cujo principal propósito é a validação funcional desses requisitos. Comportamentos observados na fase de teste e não previstos ou descritos na análise de requisitos são considerados não conformidades do sistema.
3. Modelagem do sistema utilizando-se uma hierarquia de blocos funcionais com especificação/projeto baseados em algoritmos, máquinas de estado e diagramas de fluxo de dados. A utilização mista desses meios de especificação em um mesmo modelo visa facilitar a especificação de cada componente do sistema, utilizando-se a ferramenta mais conveniente em cada caso.
4. Simulação e testes funcionais. Constitui a validação do modelo resultante da fase anterior. Os testes devem ser baseados nos requisitos funcionais definidos na etapa 1. Nesta etapa, é importante que tenha sido utilizada uma ferramenta CASE na etapa 3.
5. Geração automática de código para o hardware destino. Nesta fase é definida a arquitetura de hardware alvo do sistema e efetuada a partição e alocação do sistema entre os diversos processadores da arquitetura destino. Utilizam-se Sistemas Operacionais em Tempo Real multitarefas, RTOS (*Real Time Operating System*). Nesta etapa é necessária a utilização de Blocos Básicos de Entrada e Saída (previamente validados/certificados) para os processadores da arquitetura de hardware destino, além dos RTOS mencionados. É importante que a geração de código seja feita

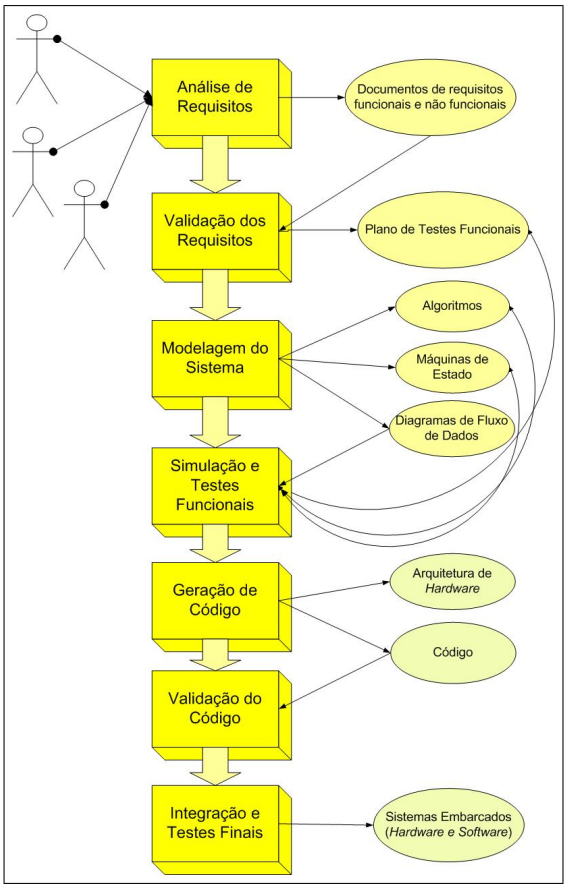


Figura 2: Metodologia proposta para o desenvolvimento de sistemas embarcados críticos

de forma completamente automática visando facilitar os processos de certificação segundo as normas aplicáveis (existem normas diferentes segundo os requisitos de qualidade de cada processo/produto). Um exemplo pode ser visto na Figura 3.

6. Validação do código gerado utilizando-se, por exemplo, a técnica *hardware in the loop*, amplamente empregada em sistemas embarcados. Nesta etapa, é importante utilizar os mesmos testes da etapa 4 (eventualmente acrescidos de outros que não poderiam ter sido aplicados naquela etapa) e comparar os resultados. Quaisquer discrepâncias, mesmo que irrelevantes em relação à análise de requisitos feita na etapa 1, devem ser completamente investigadas, corrigidas e/ou justificadas, podendo induzir à definição de novas baterias de teste.

7. Integração com o sistema final e realização dos testes finais de aceitação, que devem ser definidos sob coordenação do cliente final.

## 4 Estudo de Caso

A metodologia proposta na Seção 3 está sendo aplicada no desenvolvimento do Tiriba (Figura 4),

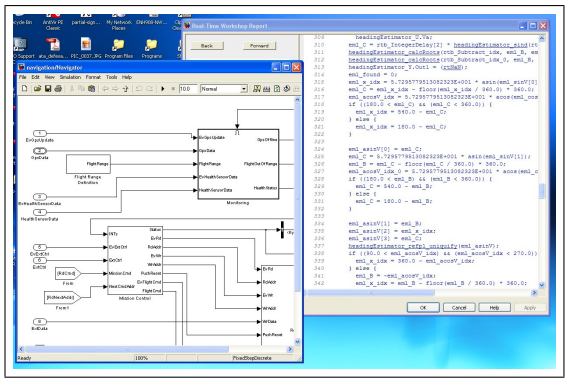

Figura 3: Bloco lógico e o código correspondente gerado de forma automática

uma aeronave não tripulada de pequenas dimensões, cujas especificações principais são apresentadas na Tabela 1.

Tabela 1: Especificações básicas da aeronave Tiriba

| | |
|---|---|
| Propulsion | Electric, 1.2KW |
| Max Takeoff weight | 3 Kg |
| Payload | 0.7 Kg |
| Endurance | 40min/1h30min |
| Cruisier speed | 100Km/h/60Km/h |
| Takeoff | hand launch/catapult |
| Landing | Automatic, parachute |
| Missions | Autonomous |
| Ground Station | Smartphone based |
| Assembly time | 10 min |

Figura 4: Tiriba - aeronave não tripulada de pequena dimensão

Nesta primeira utilização da nova metodologia, espera-se não somente validá-la quanto à sua aplicabilidade e versatilidade no desenvolvimento de sistemas embarcados críticos, como também verificar a qualidade dos produtos gerados e o impacto no tempo de desenvolvimento provocado pela sua utilização. Espera-se um ciclo de desenvolvimento total de 6 meses para o projeto, incluindo o hardware, o software e a aeronave. Um esforço combinado de 2100hs de trabalho foi previsto fazendo uso de uma equipe multi-disciplinar incluindo engenheiros e bacharéis de computação,

engenheiros eletrônicos, engenheiros aeronáuticos, técnicos em eletrônica, materiais compostos e operação de aeronaves não tripuladas.

Entre diversas opções de ferramentas para modelagem e simulação de sistemas, optou-se pelo uso neste trabalho do Matlab Simulink (Mathworks, 2009). Esta ferramenta tem se mostrado até o momento perfeitamente adequada para a metodologia e o tipo de desenvolvimento proposto, particularmente pela disponibilidade de módulos específicos para a aplicação em foco. Para a geração automática de código também está sendo empregado o Simulink. Outras ferramentas, entretanto, estão sendo avaliadas, como por exemplo, a SCADE (SCADE, 2009), cuja motivação para utilização se dá pela geração de código certificado. Nesta direção foi selecionado o RTOS uCOS (Shi et al., 2008), que está sendo integrado na metodologia permitindo o mapeamento de blocos ou conjunto de blocos funcionais, de forma automática, em processos do sistema operacional. Um bloco básico de E/S está em desenvolvimento para a integração do Simulink com a família de processadores utilizados no projeto. A arquitetura de hardware utiliza 4 processadores em uma placa única, conforme pode ser observado no diagrama de blocos apresentado na Figura 5. O primeiro processador, o de missão, é responsável pelo cumprimento de missões previamente planejadas para execução pela aeronave. Esse processador também é responsável por toda a comunicação da aeronave com a rede de estações de solo, rede esta que está sendo desenvolvida em um projeto em paralelo. O segundo processador, o de controle, é responsável pelo controle de vôo da aeronave, aceitando comandos básicos oriundos do processador de missão e da rede de estações de solo. Em uma primeira versão, a rede de estações em solo é implementada por um transmissor de controle por rádio. Os dois últimos processadores implementam os dois conjuntos principais de sensores de vôo: a unidade inercial integrada a um receptor GPS e um módulo de medida do campo magnético terrestre, responsável pelo cálculo da atitude da aeronave, e a unidade barométrica, responsável pela medição da velocidade aerodinâmica, altitude barométrica e taxa de subida da aeronave.

A adoção de quatro processadores em um projeto deste tipo teve como motivação inicial facilitar, em uma primeira etapa, a divisão das tarefas da equipe de desenvolvimento. Mais importante que isso, entretanto, ela permite de forma imediata o aproveitamento de partes do sistema como subprodutos, notadamente as unidades barométrica e inercial. Uma segunda versão da arquitetura, consistindo de apenas um processador, está prevista como teste. Esse teste visa averiguar a facilidade proposta na metodologia no que tange a novos mapeamentos de conjuntos de blocos lógicos em novas arquiteturas de hardware.

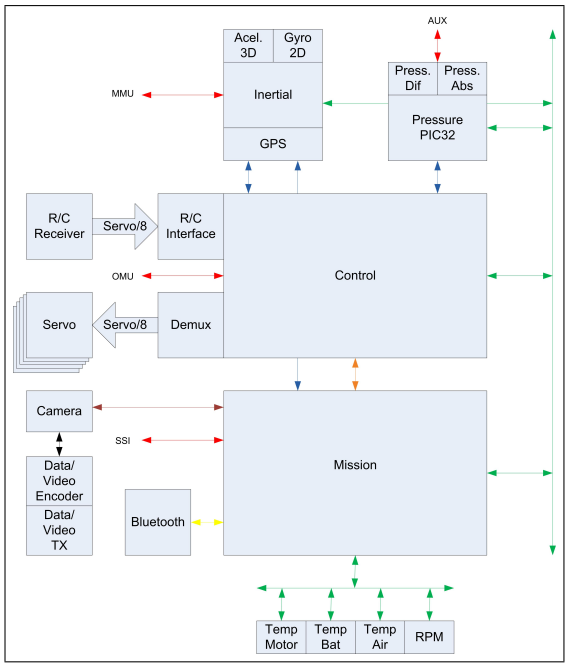


Figura 5: Diagrama de Blocos Simplificado da Aeronave

## 5 Resultados Preliminares

O estágio de desenvolvimento do projeto, no momento da escrita deste artigo, pode ser sumarizado como segue:

- Aeronave projetada, protótipo em fase final de construção, vôos de teste previstos em 3 semanas;
- Hardware do sistema projetado. Montagem e testes em 2 semanas;
- Fase de requisitos completada, modelagem em estágio final e prevista para ser concluída em 2 semanas. Alguns blocos já foram total ou parcialmente testados de acordo com a fase 4 da metodologia proposta;
- Geração de código automática testada, tendo-se feito uma inspeção e avaliação da qualidade do software produzido em relação a construções básicas de programa, utilização de memória e desempenho esperado. A conclusão inicial foi bastante favorável à ferramenta utilizada;
- Bloco básico de E/S em desenvolvimento, com prioridade para as funções que serão efetivamente utilizadas no projeto.

## 6 Conclusões

Apresentou-se neste trabalho uma metodologia para desenvolvimento de sistemas embarcados

aplicada a um sistema para controle de uma aeronave não tripulada. Pretende-se também averiguar futuramente a adequação da metodologia na geração de famílias de produtos, com diferentes níveis de complexidade e desempenho. No caso do estudo apresentado, o produto considerado é um sistema de guiamento e controle automático. Outros produtos similares poderiam ser criados com esforço relativamente pequeno fazendo-se uma reorganização e novo mapeamento em hardware dos blocos funcionais desenvolvidos.

Os resultados preliminares indicam que a utilização da metodologia proposta, além de proporcionar maior confiabilidade ao sistema proposto, também possibilita uma maior facilidade na geração da documentação necessária nos processos de certificação, por exemplo, aqueles baseados na norma DO-178B. Como resultado secundário, porém não menos importante, espera-se uma considerável redução e flexibilidade no tempo de desenvolvimento do produto, proporcionada tanto pelos ensaios de validação nos modelos quanto pelo processo de geração automática de código. O desempenho da equipe de desenvolvimento tem sido medido, o que permitirá uma comparação dos resultados obtidos em ganho de produtividade com o desenvolvimento de projetos similares sem o uso da metodologia proposta.

## Agradecimentos

Os autores agradecem ao CNPq e FAPESP pelo financiamento ao INCT-SEC, processos 573963/2008-8 e 08/57870-9.

## Referências

Beck, K. (2000). *Extreme Programming Explained: Embrace Change*, Addison-Wesley Professional.

Bhakthavatsalam, S. and Edwards, S. (2002). Applying object-oriented techniques in embedded software design, *CPES 2002 Power Electronics Seminar and NSF/Industry Annual Review*.

Chatzigeorgiou, A. and Stephanides, G. (2002). Evaluating Performance and Power of Object-Oriented Vs. Procedural Programming in Embedded Processors, *Reliable Software Technologies-ADA-Europe 2002: 7th Ada-Europe International Conference on Reliable Software Technologies, Vienna, Austria, June 17-21, 2002: Proceedings*, Springer, p. 65.

Douglass, B. P. (2004). *Real Time UML: Advances in The UML for Real-Time Systems*, 3 edn, Boston: Addison Wesley.

Hugues, J., Paris, G., Perrotin, M. and Tsiodras, T. (2008). Using MDE for the Rapid Prototyping of Space Critical Systems, *19th IEEE/IFIP International Symposium on Rapid System PrototypingI* pp. 10–16.

Kruchten, P. (2003). *The Rational Unified Process – An Introduction*, 2 edn, Addison-Wesley-Longman, Reading, MA.

Lavagno, L., Martin, G. and Selic, B. (2003). *UML for Real: Design of Embedded Real-Time Systems*, 1 edn, Kluwer Academic Publishers.

Mathworks (2009). Simulink - simulation and model-based design. Available on 2009, May 2 at http://www.mathworks.com/products/simulink/.

Microsoft (2009). .Net Micro Framework. Available on 2009, June 2 at http://www.microsoft.com/netmf/default.mspx.

Nascimento, F. A. M., Oliveira, M. F. S. and Wagner, F. R. (2007). Modes: Embedded systems design methodology and tools based on mde, *Model-Based Methodologies for Pervasive and Embedded Software, International Workshop on* **0**: 67–76.

OMG (2009). Uml profile for marte: Modeling and analysis of real-time embedded systemss. Available on 2009, March 27 at http://www.omg.org/docs/ptc/08-06-08.pdf.

RTCA, I. (2001). Final Report for Clarification of DO-178B – Software Considerations in Airborne Systems and Equipment Certification., *Technical report*, RTCA/DO-248B, 12 October 2001.

SCADE (2009). Scade - safety critical application development environment. Available on 2009, May 2 at http://www.esterel-tecnologies.com.

Shi, L., Cai, G., Huang, J. and Yao, L. (2008). Implementation of RTOS 8 mC/OS-II based on Samsung S 3 C 2410 A, *Jisuanji Gongcheng yu Sheji(Computer Engineering and Design)* **29**(10): 2523–2525.

Stahl, T. and Volter, M. (2006). *Model-Driven Software Developmen*, 1 edn, Wiley.

Wehrmeister, M. A., Becker, L. B., Wagner, F. R. and Pereira, C. E. (2005). Modeling with UML Component-Based and Aspect Oriented Programming Systems, *ISORC´05 - 8th IEEE Symposium on Object-Oriented Real-Time Distributed Computing, IEEE Computer Society Press* pp. 125–128.