

Esqueletotipação: Um Método para Desenvolvimento de Software Embarcado

Diogo Branquinho Ramos¹, Adilson Marques da Cunha¹

¹Divisão de Ciência da Computação
Instituto Tecnológico de Aeronáutica (ITA)
São José dos Campos – SP – Brasil

{diogobr, cunha}@ita.br

Abstract. *This paper describes the Skeletotyping Method designed for embedded software development comprised of five steps: 1) requirements modeling, 2) use cases modeling, 3) design pattern application, 4) source-code generation application techniques, and 5) updates and changes application. The major contributions of this paper are the Skeletotyping Method and its 3rd and 4th steps. The 3rd step involving the design pattern application has improved modeling by allowing proper boundaries between business and infrastructure layers needed to systems' operations. Finally, the 4th step involving source-code generation application techniques was based upon models and helped to ensure the embedded software development according to product specification and design.*

Resumo. *Este artigo descreve o Método de Esqueletotipação concebido em cinco passos para o desenvolvimento de software embarcado: 1) diagramação de requisitos; 2) diagramação de casos de uso; 3) aplicação de um padrão de projeto (design pattern); 4) aplicação de técnicas para geração de códigos-fonte; e 5) aplicação de alterações e mudanças. As principais contribuições deste artigo são o Método de Esqueletotipação e os seus 3º e 4º passos. O 3º passo melhorou a modelagem, permitindo delimitações apropriadas entre as camadas de negócio e de infraestrutura necessárias à operação de sistemas. Finalmente, o 4º passo ajudou a garantir o desenvolvimento de um software embarcado, de acordo com a especificação e o projeto de um produto.*

1. Introdução

Nas duas últimas décadas, constatou-se uma crescente utilização de sistemas e software embarcados anteriormente empregados apenas em ambientes complexos industriais, aeronáuticos e aeroespaciais, atualmente utilizados em veículos, televisores, celulares, entre outros dispositivos [Taurion 2005].

Dentro desse contexto, projetos de sistemas embarcados costumam envolver rigorosas especificações de requisitos, muitas vezes para cumprirem restrições de tempo. Além disto, estes projetos possuem vários fatores limitantes tais como baixa quantidade de memória disponível, pouco poder de processamento, limitação de consumo de energia sem perda de desempenho, curtos espaços de tempo para desenvolvimento, entre outros [Barr 1999].

O aumento do número de funções incorporadas a um único sistema embarcado vem ocasionando a necessidade de utilização de metodologias, técnicas e ferramentas para o gerenciamento de complexidades.

Entre outros fatores, vêm se constatando que as multidisciplinaridades das equipes de projetos devem se utilizar, cada vez mais, de comunicações claras e uniformes, de modo a minimizar os erros causados por interpretações incorretas dos requisitos e das funcionalidades dos sistemas [Douglass 2004].

Logo nas fases iniciais do desenvolvimento dos projetos, a utilização de protótipos vem representando outro fator relevante para o gerenciamento de complexidades e possibilitando oportunidades de produções de versões iniciais (à priori) de sistemas disponibilizáveis.

As construções de protótipos de hardware vêm facilitando os testes de projeto (*design*) e as elaborações de protótipos de software, auxiliando sobremaneira nas elucidações, verificações e validações de seus requisitos funcionais.

A modelagem visual, ao contemplar a possibilidade de abstrações de problemas do mundo real em modelos, vem contribuindo para reduzir a complexidade na compreensão dos requisitos dos sistemas.

A Linguagem de Modelagem Unificada (*Unified Modeling Language* - UML) e a Linguagem de Modelagem de Sistemas (*System Modeling Language* - SysML) representam os padrões notacionais mais utilizados para se abstrair software [Ramos et al. 2008]. Além disso, a modelagem visual possibilita a aplicação do paradigma de Desenvolvimento Dirigido a Modelos (*Model-Driven Development* - MDD), permitindo tanto a execução de modelos quanto a geração automática de códigos-fonte [Loubach et al. 2008].

A partir da crescente complexidade das metodologias da Engenharia de Software, surgiram as ferramentas de Ambientes Integrados de Engenharia de Software Ajudada por Computador (*Integrated - Computer Aided Software Engineering - Environment - I-CASE-E*), para o apoio as comunidades de engenharia, envolvendo gerenciamento, análise, codificação e testes, por meio de métodos e técnicas.

Entretanto, o uso desses ambientes de ferramentas, ainda se apresenta como incipiente, principalmente, por resistência dos desenvolvedores, que consideram suas configurações e usos complexos, não percebendo com isso os ganhos de qualidade, confiabilidade e segurança (*safety*) nos produtos gerados [Cunha 2006]. Cabe ressaltar também que o uso destes ambientes de ferramentas deve sempre levar em conta as necessidades dos projetos.

A principal motivação do trabalho de pesquisa que deu origem a este artigo surgiu da necessidade de se obter métodos e técnicas mais apropriadas para apoio ao desenvolvimento de software embarcado que reduzissem, principalmente, as deficiências existentes quanto aos desenvolvimentos baseados em modelos e as utilizações de ferramentas I-CASE-E.

2. Sistemas Embarcados

Os sistemas embarcados são sistemas computadorizados contendo integração de hardware e software fortemente acoplados, podendo ainda combinar possíveis partes adicionais mecânicas, elétricas ou de software para executar funções dedicadas e específicas nos contextos dos sistemas [Li and Yao 2003].

Os projetos de sistemas embarcados distinguem-se razoavelmente dos projetos de computadores pessoais, mesmo envolvendo componentes de software, de hardware e de mecânica. Os computadores pessoais não foram concebidos para desempenhar funções específicas e possuem capacidade de servir a muitos propósitos, de forma geral e não aprofundada.

2.1. Software Embarcado

O Software, como citado anteriormente, pode fazer parte de um Sistema Embarcado, assim como o hardware e as demais partes mecânicas. Neste segmento de software embarcado, empregam-se técnicas de desenvolvimento diferenciadas das normalmente utilizadas no desenvolvimento do software de propósito geral. Enquanto o software de propósito geral busca portabilidade de plataformas, o software embarcado deve ser especificado para plataformas previamente designadas, acarretando o aumento de complexidades para o seu desenvolvimento [Barr and Massa 2006].

Ultimamente, algumas técnicas de desenvolvimento de software de propósito geral vêm sendo adaptadas para o desenvolvimento de software embarcado. Entre elas cabe destacar a aplicação de: padrões de projetos (*design pattern*); técnicas de modelagem; geração automática de códigos-fontes; técnicas de testes de software; e testes de modelos.

Um padrão de projeto representa uma solução generalizada para problemas recorrentes. Neste caso, considera-se que a solução do padrão de projeto seja suficientemente geral para abranger um maior conjunto de aplicações, em diferentes domínios de conhecimento [Douglass 2003].

3. Desenvolvimento Dirigido a Modelos

As técnicas de modelagem visual vêm auxiliando o desenvolvimento de software e a sua utilização vem contribuindo sobremaneira nos seguintes aspectos:

- facilidades nos entendimentos dos problemas;
- enriquecimento de comunicações entre desenvolvedores e usuários;
- preparação de documentações dos sistemas;
- auxílio nos projetos (*design*) de software; e
- aproveitamento de automatizações de códigos-fonte e de testes.

Além disso, considerando-se a possibilidade de geração automática de códigos-fonte baseada em modelos, a modelagem visual auxiliou no advento do paradigma de MDD, focando muito mais no modelo ou abstração do sistema de software do que no aplicativo final [Azevedo 2008]. Desta forma, a aplicação de MDD evidencia as vantagens da modelagem dos sistemas nos diversos níveis de abstração da solução e nas suas integrações e fluxos de informação.

Os modelos executáveis caracterizam-se como um componente-chave do MDD. As aplicações dos seguintes conceitos possibilitam a interoperabilidade entre as diversas ferramentas baseadas em MDD: transformações automáticas de modelos; validações de modelos; e suas padronizações [Kossiakoff and Sweet 2003].

No paradigma de MDD, modelos não são utilizados somente como rascunhos de um novo Sistema. Eles fazem parte da solução, compondo um conjunto de artefatos primários que, por meio de transformações automáticas ou manuais, podem gerar implementações mais eficientes.

A UML, como uma das linguagens de notação de modelos do MDD, oferece um alto grau de flexibilidade necessária para os engenheiros de software entenderem e expressarem as complexas relações e interações de sistemas embarcados. Ela, entretanto, não é uma linguagem formal, oferecendo uma certa liberdade na modelagem. Em contrapartida, possui algumas desvantagens, devido às poucas regras envolvidas [UML 2003].

A SysML, assim como a UML, tem a intenção de unificar o modo pelo qual o desenvolvimento de sistemas é realizado. Ela representa uma linguagem de desenvolvimento de aplicações de domínio específico para Sistemas de Engenharia.

Derivada da UML, a SysML suporta as fases de especificação, análise, projeto, verificação e validação de uma gama de Sistemas e de Sistemas de Sistemas - SdS (*Systems-of-Systems* - SoS), que podem incluir hardware, software, informações, processos, pessoas, e instalações [Azevedo 2008].

Segundo [Haywood 2004], um dos pontos fracos do paradigma de MDD, que representa ainda uma questão em aberto (*open issue*), refere-se a “como desenvolver software utilizando modelos tanto abstratos quanto suficientemente completos que permitam execuções diretas e transformações automáticas em um programa executável para uma plataforma específica”.

4. O Método de Esqueletotipação

Considerando as principais vantagens da utilização do paradigma de MDD e suas limitações, esta seção apresenta o Método de Esqueletotipação como um possível auxílio para minimizar as principais dificuldades existentes e inerentes ao desenvolvimento de software embarcado dirigido a modelos.

A Engenharia de Sistemas prevê, para o desenvolvimento de sistemas embarcados, as fases de elicitação de requisitos, concepção do produto, análise e projeto, bem como a sua divisão em partes sistêmicas, envolvendo software, hardware, partes mecânicas, entre outras. O método proposto de Esqueletotipação concentra-se apenas no software embarcado e, mais especificamente, no seu desenvolvimento e integração com as demais partes dos sistemas embarcados. Além disso, a respectiva aplicação do método deve ser iniciada logo após a conclusão das fases de definições da Engenharia de Sistemas.

A aplicação do método de Esqueletotipação não exige a adoção de um processo de desenvolvimento específico, esperando-se apenas que o processo utilizado permita o uso de técnicas de modelagem de software e de ferramentas de I-CASE-E, envolvendo UML, SysML e MDD.

Este método de Esqueletotipação é composto por cinco passos estruturados: 1) diagramação de requisitos; 2) diagramação de casos de uso; 3) aplicação de um padrão de projeto; 4) aplicação de técnicas para geração de códigos-fonte; e 5) aplicação de alterações e mudanças. A Figura 1 ilustra a sua estruturação em passos.

A escolha da forma de implementação do software embarcado encontra-se fortemente interligada com as características do Produto a ser desenvolvido. Segundo [Barr and Massa 2006], os sete seguintes requisitos afetam as escolhas de projeto (*design*) no desenvolvimento de sistemas embarcados: capacidade de processamento; quantidade de memória; número de unidades a serem fabricadas; consumo de energia; custo de desenvolvimento; custo do produto; tempo de vida do produto; e confiabilidade do produto.



Figura 1. Estrutura do Método de Esqueletotipação

Os autores deste trabalho sugerem que, durante a aplicação deste método em projetos de sistemas embarcados, sejam sempre levadas em consideração: a capacidade de processamento; a quantidade de memória; o custo de desenvolvimento; e o custo do produto. Estas características são fundamentais para a realização de cada passo do método de Esqueletotipação e servem para auxiliar na abordagem do desenvolvimento de software embarcado.

4.1. 1º Passo - Diagramação dos Requisitos

No primeiro passo, a diagramação de requisitos deste método foca na organização e classificação dos requisitos que envolvem o projeto de software embarcado.

Neste passo, sugere-se a utilização do Diagrama de Requisitos da SysML, para representar graficamente os requisitos baseados em textos, além de possibilitar o relacionamento entre eles, definindo suas hierarquias, derivações, satisfações, verificações e refinamentos [OMG 2007].

Este diagrama pode ainda estar contido em outros diagramas, relacionando-se com outros elementos da modelagem. Por exemplo, com o diagrama de Casos de Uso. Segundo [Jacobson et al. 1999], esta funcionalidade, chamada de rastreabilidade, tem como metas ajudar a:

- Compreender a origem dos requisitos;
- Gerenciar as mudanças nos requisitos;
- Avaliar o impacto na mudança de um requisito; e
- Verificar se a implementação satisfaz a todos os requisitos do sistema.

4.2. 2º Passo - Diagramação de Casos de Uso

O segundo passo do método de Esqueletotipação traduz os requisitos para um modelo de casos de uso, representando as funcionalidades externa e internamente observáveis do sistema e suas interações. O modelo de casos de uso é parte integrante da especificação de requisitos, moldando os requisitos do sistema por meio do diagrama de casos de uso.

O diagrama de casos de uso possui em sua composição atores, casos de uso e relações. O caso de uso representa a especificação de uma sequência de interações entre os agentes externos do sistema, definindo o uso de suas funcionalidades. Os atores representam elementos que não fazem parte do sistema, interagindo com ele no envio e na recepção de informações. Completando o diagrama, os elementos podem se relacionar de acordo com os seguintes tipos: comunicação, inclusão, extensão e generalização [Bezerra 2002].

4.3. 3º Passo - Aplicação de um Padrão de Projeto (*Design Pattern*)

O padrão de projeto concebido, considerado uma das principais contribuições desta pesquisa, possui seu foco na organização da modelagem dos componentes de software embarcado em duas camadas bem definidas: a Camada de Negócio e a Camada de Infraestrutura. Diferentemente de outros padrões que visam a organização arquitetural do software, este padrão de projeto visa, além da organização arquitetural, a estruturação dos modelos e componentes de software para transformações em códigos-fonte.

A camada de negócio contém entre outros elementos: a lógica do funcionamento do software, reunindo as regras relativas a solução do problema sistêmico; a definição da ordem de execução das tarefas; e a transformação de dados.

A camada de infraestrutura reúne os procedimentos de controle dos dispositivos de hardware e possui como principal função manter um baixo acoplamento com a camada de negócio, fornecendo e recebendo dados de sensores, atuadores, comunicadores, entre outros dispositivos.

Desta forma, a substituição de um equipamento de infraestrutura ou até mesmo de uma tecnologia, minimizam o impacto de desenvolvimento na camada de negócio do software embarcado, tornando-a independente de plataformas de infraestrutura e conservando, como parte mais valiosa do sistema, as suas regras de funcionamento.

A aplicação do padrão de projeto na modelagem de software embarcado não exclui outros padrões já existentes, que podem ser utilizados em conjunto, atentando-se apenas para que o padrão de projeto deste método de Esqueletotipação seja o último a ser aplicado.

Esta pesquisa considera cada modelo/diagrama como protótipos executáveis [Asur and Hufnagel 1993]. Desta forma, o seu desenvolvimento pode ser testado, incrementalmente, verificando-se o seu funcionamento, a medida que novas funcionalidades são associadas ao projeto. Consequentemente, a execução dos modelos auxilia nos testes visuais de verificação da solução do problema modelado, incluindo testes de caixa-preta, de caixa-branca, entre outros.

Os protótipos executáveis podem ser implementados, segundo dois métodos: de prototipagem descartável (*throwaway*) e de prototipagem evolutiva.

Segundo [Kotonya and Sommerville 1998], a prototipagem do tipo descartável auxilia na validação da implementação dos requisitos e não é utilizada no produto final. Já a prototipagem do tipo evolutiva minimiza o tempo de desenvolvimento do software, podendo ser utilizada no desenvolvimento do produto. No caso deste método de Esqueletotipação, um modelo evolutivo será transformado em códigos-fonte.

A Figura 2 ilustra a aplicação do padrão de projeto. Nela, pode-se observar a divisão entre camadas e a classificação dos protótipos executáveis, por meio de esteriótipos.

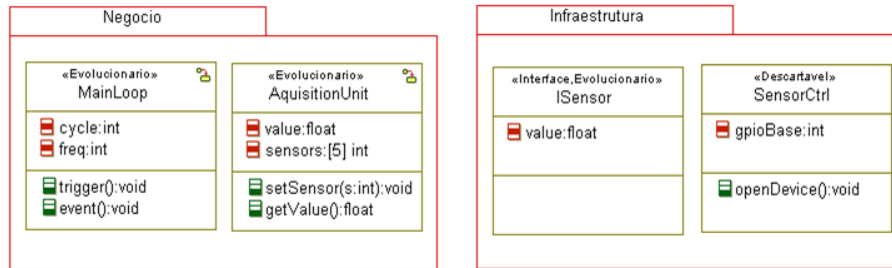


Figura 2. Exemplo de aplicação do padrão de projeto

4.4. 4º Passo - Aplicação de técnicas para geração de códigos-fonte

Concluído o terceiro passo de aplicação do padrão de projeto, o quarto passo inclui a geração de códigos-fonte baseada em modelos que pode ser realizada a partir de uma das três seguintes técnicas de: geração manual, geração automática ou geração híbrida.

A escolha de uma das técnicas encontra-se fortemente ligada com a natureza do projeto de sistema embarcado. Como citado anteriormente, os requisitos de capacidade de processamento, quantidade de memória, custo de desenvolvimento e custo do produto devem ser considerados para a escolha da técnica de geração.

Por exemplo, a geração automática requer uma ferramenta de I-CASE-E para realização desta funcionalidade, podendo impactar no custo de desenvolvimento, considerando também a sua aquisição. Além disso, a ferramenta de I-CASE-E exige um espaço maior de memória para o arquivo binário, impactando no custo do produto. Outra consideração importante é o ganho de produtividade.

Entre outros fatores, essas características devem ser analisadas e julgadas antes da escolha das técnicas de geração de códigos-fonte.

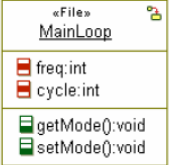
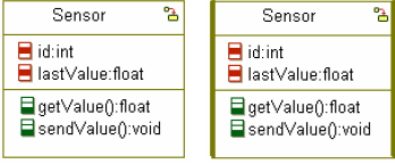
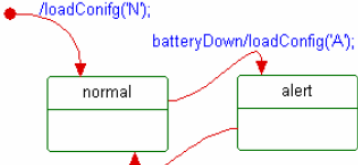
4.4.1. Técnica de Geração Manual

A técnica de geração de códigos-fonte manual consiste na transformação visual da modelagem de software, utilizando-se um roteiro ou guia transformacional para cada elemento do modelo. Esta técnica é indicada para o desenvolvimento de sistemas embarcados com acoplamentos altos ou complexos entre software e hardware. Estes sistemas embarcados apresentam pouco poder de processamento e de memória disponível, porque necessitam de implementações tecnológicas específicas nas interfaces de dispositivos de hardware e de um melhor aproveitamento dos recursos do Sistema Operacional Embarcado - SOE.

Considerando a modelagem do software em camadas e os testes de execução do modelo, a partir deste ponto, basta se realizar a transposição dos modelos para os códigos-fonte. As transformações, neste caso, não devem ser consideradas como uma relação 1 para 1, mas sim como um roteiro ou guia para facilitar a concepção do algoritmo a ser desenvolvido.

A Tabela 1 apresenta as relações entre os elementos do modelo e o guia transformacional para códigos-fonte. Neste trabalho, foi utilizada a ferramenta de I-CASE-E *IBM Rational Rhapsody 7.4*. Note-se que, apesar da utilização da UML, algumas representações gráficas dos modelos poderão ocorrer de forma diferente em outras ferramentas:

Tabela 1. Guia transformacional para os elementos do modelo

Elementos do Modelo	Guia de transformação
 <p>Arquivo - C</p>	<p>O elemento <i>File</i> do modelo representa, na linguagem C, um arquivo .c e .h. No primeiro contêiner, encontra-se o nome do arquivo; já no segundo, as variáveis e seus tipos; e no terceiro contêiner, as funções e seus retornos. Mais informações podem ser acessadas, ao se navegar no modelo.</p>
 <p>A B</p> <p>Classe - C++</p>	<p>O elemento a esquerda representa uma classe na linguagem C++. O elemento classe possui, no primeiro contêiner, o seu nome; no segundo, a relação de seus atributos; e no terceiro, os métodos da classe. A representação gráfica distinta significa que a classe A é executada na mesma <i>thread</i> do sistema e a B é executada em uma <i>thread</i> própria.</p>
 <p>Máquina de Estados Finitos</p>	<p>O elemento Máquina de Estados Finito pode ser transformado em um simples comando <i>Switch</i> (em C ou C++), onde cada <i>Case</i> representa um estado da máquina. A mudança de transição (<i>batteryDown/Up</i>) é representada na variável de escolha do <i>Switch</i>, armazenada em uma Enumeração. As ações da transição (<i>loadConfig('A/N')</i>) representam os comandos, dentro de cada <i>Case</i>.</p>

4.4.2. Técnica de Geração Automática

Esta técnica consiste na geração automática de códigos-fonte, utilizando-se as regras transformacionais já existentes nas ferramentas de I-CASE-E. Ela é indicada, principalmente, para o desenvolvimento de software embarcado em plataformas de hardware de propósitos mais gerais ou com recursos de processamento e de memória menos escassos.

A geração automática de códigos-fontes pode ser considerada também para a comunicação entre dispositivos de hardware e software, contendo acoplamentos baixos ou simples, como por exemplo: RS-232, RS-485, *Ethernet*, entre outros.

4.4.3. Técnica de Geração Híbrida

A aplicação desta técnica no quarto passo do Método de Esqueletotipação, considerada também como uma das principais contribuições desta pesquisa, reúne os pontos fortes das gerações de códigos-fonte manual e automática, de forma sincronizada, no desenvolvimento de software embarcado.

Nesta técnica, a geração manual é fortemente indicada para a camada de infraestrutura, visando um melhor controle de dispositivos de hardware com acoplamentos altos ou complexos com o software. Já a geração automática ajusta-se melhor para transformação da camada de negócio em códigos-fonte. No entanto, não existem restrições para que um elemento da camada de negócio seja gerado pela técnica de geração manual e vice-versa.

4.5. 5º Passo - Aplicação de Alterações e Mudanças

O quinto e último passo do Método de Esqueletotipação pode ser considerado como um passo alternativo utilizado no decorrer da aplicação deste método no desenvolvimento de software embarcado. A função principal deste passo é a de garantir o sincronismo do projeto em relação as alterações e mudanças necessárias para o sistema de software.

5. Estudo de Caso

A verificação da aplicação do Método de Esqueletotipação foi realizada por meio de um estudo de caso que contempla uma parte de um projeto real denominado Projeto de Integração e Cooperação Amazônica para a Modernização do Monitoramento Hidrológico - ICA-MMH. Este projeto, financiado pela Financiadora de Estudos e Projetos - FINEP, encontra-se em fase final de desenvolvimento no Instituto Tecnológico de Aeronáutica - ITA, em parceria com a Agência Nacional de Águas - ANA e com o Instituto de Pesquisas Hidráulicas - IPH.

O objetivo geral do Projeto ICA-MMH é desenvolver, para uma região hidrográfica de referência, um sistema-piloto que contemple a modernização e a integração de pontos de coletas telemétricas de dados hidrometereológicos.

O estudo de caso envolveu a aplicação do Método de Esqueletotipação em uma parte significativa do desenvolvimento do software embarcado da Plataforma de Coleta de Dados - PCD do Projeto ICA-MMH, responsável pela aquisição e transmissão de dados hidrometereológicos via satélite, celular ou rádio-frequência.

5.1. O Projeto ICA-MMH

Na concepção sistêmica do Projeto ICA-MMH, realizada a partir de uma análise de requisitos, concebeu-se um cenário com 5 grandes grupos de requisitos com propósitos comuns e coesos, possibilitando a formação de 5 sistemas que se constituíram em um sistema maior, o Sistema de Sistemas ICA-MMH (SdS ICA-MMH).

Os seguintes Sistemas compõem o SdS ICA-MMH: Sistema de Aquisição de Dados - SAD; Sistema de Tratamento de Dados - STD; Sistema de Banco de Dados - SBD; Sistema de Monitoramento, Controle e Apoio à Decisão - SMCAD; e Sistema de Difusão de Dados - SDD.

O Sistema de Aquisição de Dados é composto pelo Sistema Embarcado da PCD e por um Sistema de Software para recepção de dados hidrometereológicos e envio de configurações remotas.

5.2. A Plataforma de Coleta de Dados - PCD

A PCD disponibiliza informações hidrometereológicas, visando um melhor acompanhamento das condições hidrológicas e uma melhor previsão de eventos extremos tais como secas, inundações, entre outros.

Cada PCD deve ser capaz de propiciar a transmissão dos dados coletados via um desses meios de comunicação previstos: celular, rádio-frequência ou satelital. A redundância de comunicação deve ser controlada por um algoritmo de comutação automática dos meios de comunicação.

De acordo com as especificações sistêmicas do Projeto ICA-MMH, o software embarcado da PCD deve ser desenvolvido em um *Kit de Desenvolvimento ARM9 - Congent CSB 637*. Esta plataforma possui um microprocessador AT91RM9200 de 184Mhz, com 64 MByte SDRAM e 8 MByte de memória *flash*, com a utilização de um Sistema Operacional Embarcado - SOE Linux.

5.3. Aplicação do Método de Esqueletotipação no Projeto da PCD

A partir das definições sistêmicas do Projeto ICA-MMH, das divisões dos sistemas, das definições da plataforma embarcada, entre outras definições, tornou-se possível aplicar o Método de Esqueletotipação no sistema embarcado do Projeto.

Para a aplicação neste Estudo de Caso, considerou-se as seguintes funcionalidades da PCD: modos de operação (normal, alerta e crítico); atualização dos parâmetros de configuração; e aquisição de dados dos sensores (pluviômetro, sonda multiparamétrica e pressão barométrica).

Inicialmente, dependeu-se algum tempo considerado irrelevante para a configuração e integração do ambiente de ferramentas de I-CASE-E. As principais integrações ocorreram entre a ferramenta de gerência de requisitos *IBM Rational RequisitePro 7.1* e a ferramenta de modelagem de software embarcado *IBM Rational Rhapsody 7.4*.

5.3.1. Aplicação do 1º Passo no Projeto da PCD

Antes de se iniciar a diagramação de requisitos, realizou-se a importação para a ferramenta de modelagem, dos requisitos textuais contidos na ferramenta de gerência de requisitos. A partir da relação dos requisitos do SAD, decorrentes da importação anterior, envolvendo o Projeto da PCD, realizou-se a diagramação dos requisitos e as definições das relações entre eles. A Figura 3 ilustra a representação gráfica dos requisitos diagramados.

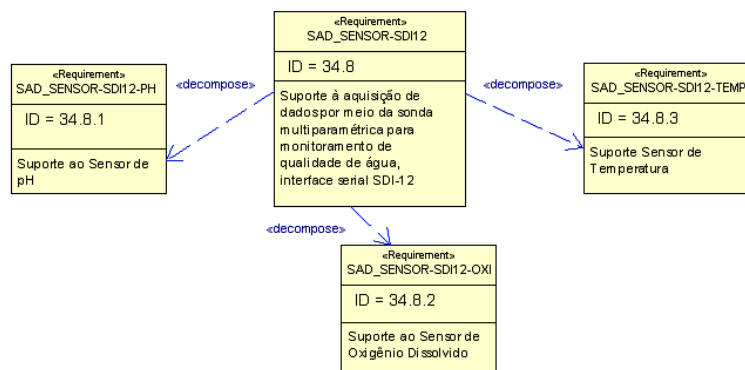


Figura 3. Diagramação dos Requisitos do software embarcado da PCD

5.3.2. Aplicação do 2º Passo no Projeto da PCD

Após a realização da organização dos requisitos, a diagramação dos modelos de casos de uso foi elaborada, de forma relativamente simples, identificando-se os atores externos do sistema, as suas conexões e as interações entre o sistema.

A partir deste ponto, os requisitos diagramados puderam ser relacionados com os casos de uso, obtendo-se a rastreabilidade do requisito, tornando possível a verificação dos requisitos contemplados pelo sistema.

A Figura 4 ilustra um exemplo do modelo de casos de uso relacionado com os requisitos, por meio da ligação *trace*.

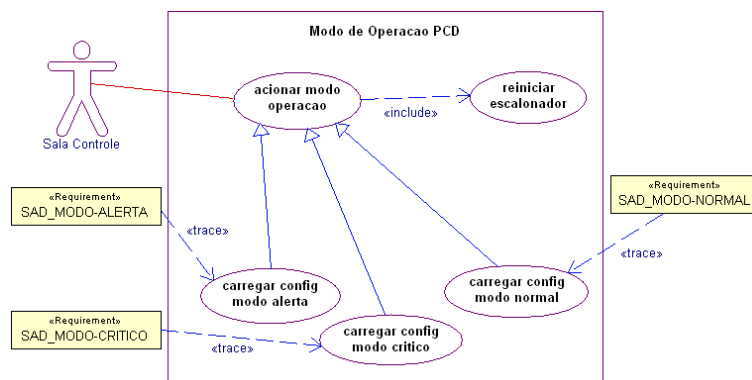


Figura 4. Diagramação dos Casos de Uso do software embarcado da PCD

5.3.3. Aplicação do 3º Passo no Projeto da PCD

Considerando o escopo dos requisitos e seus detalhamentos nos modelos de casos de uso, pôde-se iniciar a aplicação do terceiro passo.

Este passo, considerado uma das principais contribuições da pesquisa, consistiu na aplicação do padrão de projeto concebido, que dividiu o software embarcado em camadas de negócio e infraestrutura. Durante a concepção da modelagem, observou-se a divisão das camadas e a classificação dos modelos/protótipos em evolucionário ou descartável, ambas previstas no terceiro passo do Método de Esqueletotipação. A classificação auxiliou na identificação de quais elementos deveriam participar da geração dos códigos-fonte.

Nesta oportunidade, constatou-se que os modelos puderam ser executados no ambiente *host*, para fins de simulação e verificação da solução em desenvolvimento. Constatou-se também que algumas das ferramentas de I-CASE-E possuíam a capacidade de integração com outras ferramentas de modelagem. Por exemplo, a ferramenta de I-CASE-E *Rhapsody 7.4* pôde ser integrada com a ferramenta de modelagem *MathWorks Matlab Simulink*, viabilizando possíveis futuras simulações e verificações de integrações de produtos.

No Projeto da PCD, existem alguns sensores hidrometeorológicos que possuem interfaces de acoplamentos altos ou complexos com o hardware. Neste caso, constatou-se, durante a aquisição de dados, via *General Purpose Input/Output* - GPIO ou via con-

versores Analógicos/Digitais - A/D, que estas interfaces não conseguem ser bem representadas pela UML e as tentativas de representações são todas baseadas em adaptações de baixa qualidade dos elementos do modelo. Além disso, constatou-se também que a implementação desta interface torna-se mais vantajosa, ao se considerar a utilização de técnicas de SOE. Por exemplo, quando se considera a implementação de módulos do *kernel* (*devices drivers*).

A Figura 5 ilustra os elementos de modelagem da PCD e suas classificações, quanto aos modelos evolucionários ou descartáveis. Neste caso, os sensores que possuem interfaces de baixo nível foram considerados descartáveis.

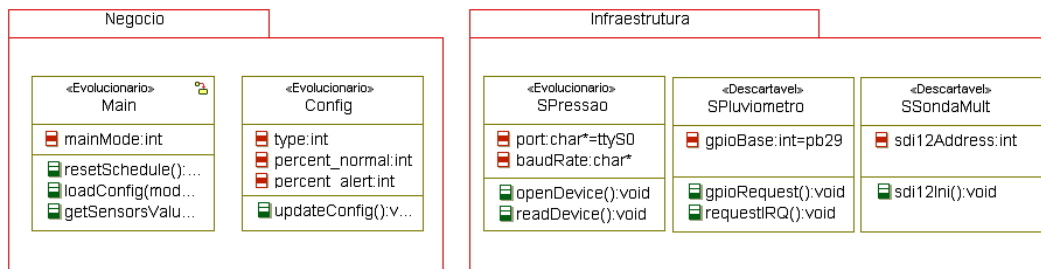


Figura 5. Aplicação do padrão de projeto na modelagem do software embarcado da PCD

5.3.4. Aplicação do 4º Passo no Projeto da PCD

Após a modelagem do software embarcado, a aplicação do quarto passo consistiu na escolha de uma das técnicas de geração de códigos-fonte.

Este passo, também considerado com uma das principais contribuições desta pesquisa, consistiu na aplicação da técnica de geração automática. Esta técnica foi considerada, inicialmente, como a mais adequada para o Projeto da PCD pelas suas características e especificações da plataforma de hardware. Porém, ao se constatar a existência de alguns sensores com necessidades de implementações especiais, a técnica de geração híbrida mostrou-se mais eficiente para a geração de códigos-fonte baseados em modelos.

Desta forma, toda a modelagem da camada de negócio foi gerada automaticamente pela ferramenta *Rhapsody 7.4*, incluindo os elementos de interface e os sensores com acoplamentos baixos ou simples entre o software e o hardware da camada de infraestrutura. Quanto aos demais sensores, foram construídos baseados nas técnicas de geração manual de códigos-fonte.

A Figura 6 apresenta, no item A, as configurações necessárias para a geração automática de códigos-fonte na ferramenta de desenvolvimento. Esta figura, no item B, apresenta a implementação do módulo do *kernel* do SOE utilizado no Projeto da PCD.

6. Conclusão

A aplicação do Método de Esqueletotipação no desenvolvimento da Plataforma de Coleta de Dados - PCD do Projeto de Integração e Cooperação Amazônica para a Modernização do Monitoramento Hidrológico - ICA-MMH mostrou-se satisfatória, principalmente

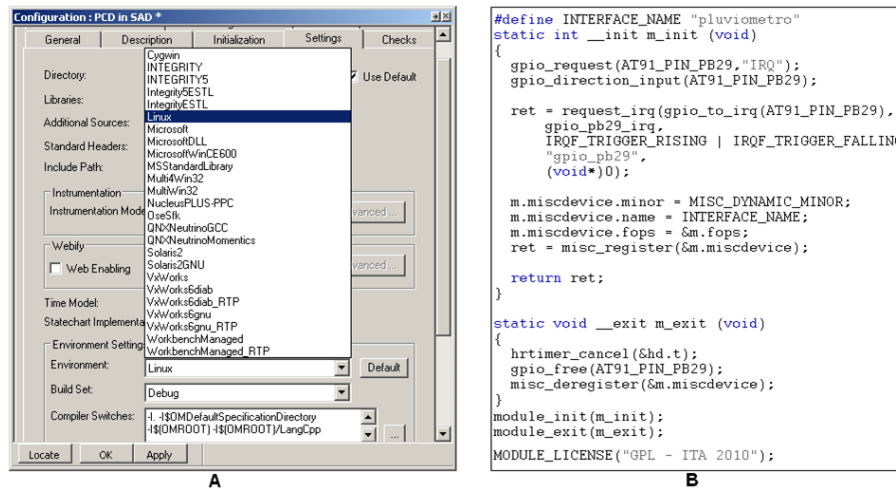


Figura 6. A - Configuração da ferramenta de I-CASE-E e B - Implementação do módulo do kernel

quanto a possibilidade de documentação, via modelos, de todos os elementos de software e de interfaceamento de hardware.

Este artigo apresentou como principais contribuições: a concepção de um método denominado de Esqueletotipação para o desenvolvimento de Software Embarcado; o desenvolvimento e a aplicação de um padrão de projeto (*design pattern*), no terceiro passo deste método; e a elaboração e aplicação de uma técnica de geração híbrida de códigos-fonte, no seu quarto passo.

A aplicação de um padrão de projeto permitiu uma melhor divisão e delimitação das fronteiras entre as camadas de negócio e de infraestrutura necessárias ao funcionamento do sistema. Desta forma, qualquer dispositivo de infraestrutura (sensores, atuadores, comunicadores, entre outros) passou a poder ser substituídos por dispositivos equivalentes, sem causar mudanças e/ou adequações à camada de negócio do sistema.

Finalmente, a geração híbrida de códigos-fonte permitiu um melhor aproveitamento dos recursos do Sistema Operacional Embarcado - SOE, considerando a abordagem de geração manual para a camada de infraestrutura. Com isto, constatou-se um controle mais eficiente dos dispositivos de hardware do tipo *device drivers*, durante a implementação dos módulos do *kernel* do Sistema Operacional Embarcado.

7. Agradecimentos

Os autores deste artigo agradecem as seguintes Instituições que apoiaram, direta ou indiretamente, a realização deste trabalho: Comando-Geral de Tecnologia Aeroespacial - CTA; Instituto Tecnológico de Aeronáutica - ITA; Agência Nacional de Águas - ANA; Financiadora de Estudos e Projetos - FINEP; e Fundação Casimiro Montenegro Filho - FCMF.

Referências

Asur, S. and Hufnagel, S. (1993). Taxonomy of rapid-prototyping methods and tools. *Rapid System Prototyping, 1993. Shortening the Path from Specification to Prototype. Proceedings., Fourth International Workshop on*, pages 42–56.

- Azevedo, R. d. C. (2008). Colibri: Um framework colaborativo para engenharia de sistemas embarcados baseada em modelos híbridos. Master's thesis, Instituto Tecnológico de Aeronáutica.
- Barr, M. (1999). *Programming embedded systems in C and C++*. O'Reilly.
- Barr, M. and Massa, A. (2006). *Programming embedded systems: with C and GNU development tools*. O'Reilly.
- Bezerra, E. (2002). *Princípios de Análise e Projeto de Sistemas com UML*. Campus.
- Cunha, A. M. d. (2006). Notas de aula da disciplina ce-235 sistemas embarcados de tempo real, do programa de pós-graduação em engenharia eletrônica e computação, na Área de informática - pg/eec-i. Instituto Tecnológico de Aeronáutica - ITA.
- Douglass, B. P. (2003). *Real-Time Design Patterns: robust scalable architecture for Real-time systems*. Addison Wesley.
- Douglass, B. P. (2004). *Real Time UML Third Edition - Advances in The UML for Real-Time Systems*. Person Education, Inc.
- Haywood, D. (2004). Mda: Nice idea, shame about the... Disponível em: <http://www.theserverside.com/news/1365166/MDA-Nice-idea-shame-about-the>.
- Jacobson, I., Booch, G., and Rumbaugh, J. (1999). *The Unified Software Development Process*. Addison-Wesley.
- Kossiakoff, A. and Sweet, W. N. (2003). *Systems Engineering Principles and Practice*. John Wiley & Sons.
- Kotonya, G. and Sommerville, I. (1998). *Requirements Engineering: Processes and Techniques*. Wiley.
- Li, Q. and Yao, C. (2003). *Real-Time Concepts for Embedded Systems*. CMP Books, Lawrence, KS, USA.
- Loubach, D., Ramos, D., Saotome, O., and da Cunha, A. (2008). Comparing source codes generated by case tools with hand coded. *Information Technology: New Generations, 2008. ITNG 2008. Fifth International Conference on*, pages 1292–1292.
- OMG (2007). Omg systems modeling language (omg sysml tm). <http://www.sysmlforum.com/>.
- Ramos, D. B., Loubach, D. S., and da Cunha, A. M. (2008). Developing a distributed real-time monitoring system to track uavs. *Digital Avionics Systems Conference, 2008. DASC 2008. IEEE/AIAA 27th*, pages 4.C.6–1–4.C.6–9.
- Taurion, C. (2005). *Software Embarcado - A nova onda da Informática*. Editora Brasport.
- UML (2003). Profile for scheduability, performance and time, ptc2003-03-02. <http://www.omg.org/>.