Ferramentas de Projeto baseado em Plataforma UML de Sistemas de Gerência de Configuração Multi-Placas em Chassi

Rodrigo de A. Moreira, Alice M. Tokarnia (Orientador)

Departamento de Engenharia de Computação e Automação Industrial (DCA)
Faculdade de Engenharia Elétrica e de Computação (FEEC)
Universidade Estadual de Campinas (Unicamp)
Caixa Postal 6101, CEP 13083-970 – Campinas, SP, Brasil

ramoreira@gmail.com, tokarnia@dca.fee.unicamp.br

Abstract – The design and implementation of embedded software for chassi management is a hidden part of the development of many communication systems. This embedded software is usually developed from scratch for each system using ad-hoc techniques. In this paper, we present a design toolbox for chassi management software that performs several steps of development automatically. Our toolbox includes a set of UML configurable software components; a simplified hardware modeling tool; a software synthesis package that constructs a module from functional components and communication proxies; and a simple estimation tool. A small management system is used to illustrate the use of our toolbox.

Keywords – Embedded system, Chassi management, UML platform, Software synthesis.

1. Introdução

Estruturas de múltiplas placas, agrupadas em chassi, são comumente encontradas em sistemas complexos, que incluem funções implementadas em hardware e software. Como exemplo, considere os sistemas de telecomunicações, que vêm se tornando mais complexos e podem incluir muitos dispositivos [1]. Atualmente, estes dispositivos requerem projeto de hardware e software embarcado para executar suas funções.

Um chassi é composto por placas, montadas em trilhas e conectadas através de um *backplane*. As placas são classificadas em placas de linha e placas de controle, central ou periféricas [2]. As placas de linha executam funções específicas do sistema, enquanto as placas de controle se encarregam de funções de gerenciamento do chassi [2], tais como: i) monitoramento de temperatura, ventilação e potência; ii) realização de inventário do sistema; iii) detecção de falhas e iv) atualização de softwares. Estas funções visam garantir que o sistema satisfaça as especificações de desempenho, temperatura, ventilação e confiabilidade, que são independentes da função específica do sistema e, na verdade, encontradas em várias estruturas multi-placas em chassi. Apesar disto, para cada chassi, geralmente um novo projeto é realizado sem reaproveitar partes de projetos anteriores e o desenvolvimento é baseado em métodos informais de projeto e técnicas manuais de codificação.

O objetivo deste trabalho é tornar mais rápido o projeto de sistemas de gerenciamento de configuração multi-placas em chassi, através de uma metodologia implementada por um conjunto de ferramentas de projeto. Estas ferramentas, reunidas no CMS-SEUP (*Chassi Management Software Synthesis and Estimation with UML Platforms*), dispõe dos seguintes recursos: 1) Biblioteca de componentes configuráveis de software, descritos em UML (*Unified Modeling Language*) [6]; 2) Ferramenta para geração automática de software, com inclusão do software de comunicação entre processadores, localizados na mesma placa ou em placas distintas e 3) Modelo simplificado de hardware que reúne informações para configurar componentes de software e calcular estimadores simples de desempenho, temperatura, potência e tamanho de memória.

Os conceitos e parte das ferramentas empregados no ambiente CMS-SEUP estão disponíveis em outras publicações. Ke *et. al.* descrevem um método de projeto baseado em componentes [3]. Marcio *et. al.* apresentam um método e uma ferramenta para explorar o espaço de projeto de software embarcado usando uma plataforma de software descrita em UML [4], mas não apresenta nenhum método para automatização do projeto de software. Shourong *et. al.* apresenta um método para síntese de componentes de software a partir de modelos UML [5]. Nosso trabalho se diferencia dos anteriores por reunir, num mesmo ambiente, ferramentas de projeto de software baseado em plataforma para gerenciamento de estruturas multi-placas em chassi.

O restante deste artigo está organizado da seguinte forma. A seção 2 apresenta a metodologia de projeto usada no CMS-SEUP. Um exemplo de projeto de software de gerenciamento é descrito na seção 3. A seção 4 traz a conclusão e os trabalhos futuros.

2. Proposta

O software embarcado de gerenciamento de configuração multi-placas em chassi é comumente armazenado em memórias *flash* e executado pelos processadores de uma plataforma de hardware. As ferramentas introduzidas neste trabalho permitem realizar automaticamente várias etapas do projeto de software embarcado, incluindo estimadores simples para alguns requisitos não-funcionais.

As ferramentas do CMS-SEUP auxiliam no projeto de sistemas de gerenciamento de configuração multi-placas em chassi desde a captura da especificação do sistema até a geração do código fonte e estimativa de parâmetros não-funcionais, conforme ilustrado na Figura 1. As ferramentas utilizam as quatro entradas a seguir: i) descrição das funcionalidades do sistema ii) descrição simplificada da plataforma de hardware de cada placa; iii) fatores de redundância e iv) biblioteca de componentes de software [6].

A especificação do sistema é realizada pelo projetista através das seguintes etapas:

1. Escolha das funcionalidades do sistema e especificação de restrições de desempenho, consumo de energia, tamanho de memória e temperatura.
2. Especificação simplificada da plataforma de hardware para configuração dos componentes de software e cálculo dos estimadores.
3. Especificação de *fatores de redundância*, que permitem ao projetista descrever características de operação em caso de falha e de manutenção. Estes fatores influenciam a seleção de componentes e o número de placas de controle do projeto. A Tabela 1 apresenta alguns fatores de redundância e suas implicações no projeto.

Após a captura da especificação, as próximas etapas, demarcadas pelo retângulo da Figura 1 são realizadas automaticamente. Em primeiro lugar, é feita a seleção dos componentes de software de uma biblioteca de acordo com as funcionalidades especificadas. Os componentes são customizados, suprimindo as funções não utilizadas. Em seguida, é realizado o mapeamento dos componentes de software nos componentes de hardware, levando em conta os *fatores de redundância*. O mapeamento define a configuração dos componentes e a necessidade de inclusão de *proxies* de comunicação.

Neste ponto, são calculados estimadores simples de desempenho, tamanho de memória e temperatura, apenas para uma primeira avaliação da especificação. O estimador de desempenho

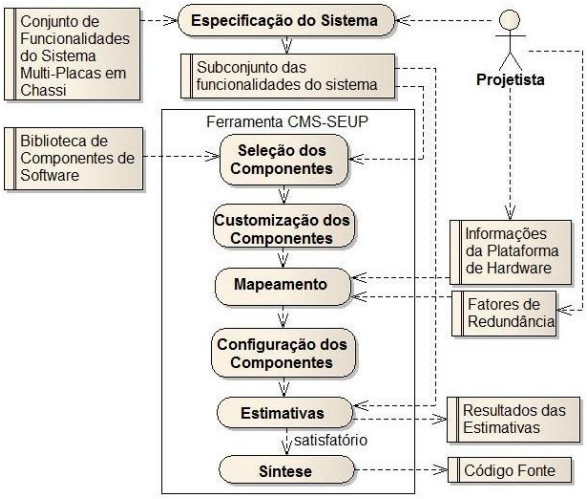

Figura 1 - Fluxo do Projeto CMS-SEUP.

Tabela 1 - Alguns fatores de redundância.

Fator 1: Redundância do placa central
Implicações: Replicação da placa de controle central e uso de componentes de sincronização entre as placas de controle central e de detecção de falha de placa.
Fator 2: Substituição de placas de linha sem a parar o sistema
Implicações: Componentes para armazenar as configurações das placas de linha e realizar a sincronização entre a placa central e placa de linha.
Fator 3: Redundância do sistema de ventilação
Implicações: Placas separadas para ventilação de subconjuntos de placas e componentes individuais de controle de ventilação.

leva em conta o algoritmo de escalonamento, o maior tempo de execução das tarefas (WCET-*worst case execution time*) nos processadores para as quais foram mapeadas e os períodos de cada tarefa. O estimador de temperatura leva em conta informações do fabricante e da plataforma de hardware para determinar se há necessidade de dissipador e/ou de ventilador para manter a temperatura especificada para o chassi. O estimador de tamanho de memória soma os tamanhos dos arquivos de programa e verifica se a memória não-volátil especificada é suficiente. É feita também a soma dos tamanhos dos dados e calculada a fração que pode ser armazenada na memória volátil. Como os dados podem não ser usados simultaneamente, cabe ao projetista avaliar se é suficiente.

A etapa final consiste na síntese de códigos fonte e de arquivos de comandos para geração do código. É possível gerar códigos executáveis, se os compiladores estiverem disponíveis.

3. Exemplo

Descrevemos a seguir as etapas do projeto de um sistema de gerenciamento de chassi e os

resultados obtidos pelas ferramentas CMS-SEUP. As funcionalidades deste sistema de gerenciamento estão descritas na Tabela 2. Neste caso, o projetista selecionou sete funcionalidades e especificou seis restrições de tempo, que são os períodos para execução das tarefas.

As informações da plataforma de hardware, apresentadas na Tabela 3, são usadas para estimativa de desempenho e temperatura e na síntese do software de comunicação. A descrição de memória é usada para verificar se o código sintetizado pode ser armazenado. Neste exemplo não foram especificados fatores de redundância.

3.1. Seleção dos Componentes

A ferramenta CMS-SEUP faz uma busca na biblioteca pelos componentes de software necessários para satisfazer as funcionalidades da Tabela 2. Esta biblioteca, apresentada em [6], inclui componentes de software específicos para projeto do sistema de gerenciamento de configuração multi-placas em chassi. Os componentes eventos selecionados para este projeto são mostrados na Tabela 4: Interface de Linha de Comando (*CLI*), Diagnóstico, Controlador Central e Gerenciador de Eventos.

3.2. Customização dos Componentes

Esta etapa suprime os componentes passivos do componente evento Diagnóstico, que não são usados para implementar as funcionalidades da especificação, descrita pela Tabela 2.

Tabela 2 – Subconjunto de funcionalidades.

Funcionalidades	Restrições
	Período
Monitorar Temperatura	7
Monitorar *Slots*	7
Controlar *Slots*	12
Monitorar Ventiladores	7
Controlar Ventiladores	12
Enviar Eventos	20
Interface de Linha de Comando	-

Tabela 3 - Informações da plataforma de hardware.

Processador	
• Nome	P1
• Freqüência	500 MHz
• Potência dissipada	10 W
• Temperatura do Chip	130 ºC
Comunicação (Com.) entre as placas	
• Ethernet	100 Mpbs
Com. entre processador e sensores/atuadores	
• Serial RS232	19900 bauds
Memória Flash / Memória RAM	
• Tamanho	64 KB / 32 KB
Chassi	
• Número de Slots	7
• Temperatura Máxima	60 ºC
• Dimensão das placas (mm)	233x250x22

Tabela 4 - Especificação do Sistema e Componentes.

Funcionalidade	Algumas Funções no Projeto de Software	Componentes
Monitorar *Slots*	GET_SLOT_STATUS GET_SLOT_STATE	Diagnóstico e Ger. de Eventos.
Controlar *Slots*	SET_SLOT_ENABLE SET_SLOT_DISABLE	Cont. Central e Ger. de Eventos.
Monitorar Temperatura	GET_TEMPERATURE	Diagnóstico e Ger. de Eventos.
Monitorar e Controlar Ventiladores	GET_FAN_RPM GET_FAN_STATUS SET_FAN_ROTATION	Cont. Central e Ger. de Eventos.

3.3. Mapeamento

Como não há redundância, a ferramenta utiliza uma implementação *básica* com duas placas de controle, uma central e outra periférica. Na placa de controle central são mapeados os componentes evento: *CLI*, Diagnóstico, Controlador Central e Gerenciador de Eventos. Na placa de controle periférica, que contém os ventiladores, é mapeada a função passiva Monitor de Velocidade dos Ventiladores.

3.4. Configuração dos Componentes

Com base nas informações de hardware, a ferramenta configura os componentes conforme descrito na Tabela 5. De acordo com o modelo de software básico [6], a comunicação entre um componente passivo e o componente evento que o contém é realizada através de chamadas com bloqueio. Por outro lado, a comunicação entre os componentes eventos é realizada sem bloqueio através de um *Proxy*. Um *Proxy* utiliza chamadas para rotinas *IPC* (*Inter Process Communication*), responsáveis pela troca de dados entre processos [8]. Desta maneira, como os componentes eventos são implementados como processos, é necessário configurar o tipo de *IPC* do *Proxy* para efetuar a comunicação entre os componentes *CLI* e Diagnóstico. Como eles estão mapeados no mesmo processador, a ferramenta configurou o *IPC* como uma *fifo* (*named pipe*). O *IPC* do *Proxy* entre os componentes Diagnóstico da placa central e o Diagnóstico da placa periférica foi configurado como um *Socket-Udp*, pois estão mapeados em processadores diferentes e possuem uma rede ethernet de comunicação.

Tabela 5 - Configuração dos componentes.

Componentes		
Evento	**Passivo**	**Configuração**
CLI	-	Comandos da interface
Diagnóstico	Monitor de Temperatura	Temperatura máxima
	Monitor de Slot	Quantidade de slots do chassi
	Monitor de Velocidade dos Ventiladores	Quantidade de placas de ventiladores

3.5. Estimativas

Neste exemplo, o estimador de desempenho considera um modelo de processo simples, onde todos os processos têm seus *prazos* (D) iguais aos períodos (T) especificados. As prioridades são atribuídas segundo o critério de taxa monotônica e é realizada uma análise de tempo de resposta para escalonamento baseado em prioridades (P) com preempção descrita em [7]. A ferramenta leva em conta o período de cada tarefa e o tempo de computação (C) dos componentes anotado na Tabela 2.

O pior caso para os tempos de resposta dos componentes (W), calculado usando a técnica apresentada em [7], é mostrado na Tabela 6. Neste caso, todos os componentes satisfazem os tempos de os prazos de execução.

O estimador de temperatura utiliza fórmulas fornecidas pelo fabricante e informações da plataforma de hardware para determinar o dissipador e os ventiladores necessários para manter a temperatura especificada para o chassi.

Para estimar o tamanho de memória, a ferramenta leva em conta as informações da plataforma de hardware na Tabela 3 e o tamanho anotado de cada componente utilizado. Estas informações são mostradas na Tabela 7.

3.6. Síntese

A etapa de síntese gera automaticamente o software executável de cada componente para cada processador. Para isto, a ferramenta utiliza: i) o código fonte dos componentes que estão armazenados na biblioteca; ii) as informações da plataforma de hardware e iii) o endereço dos *cross-compiladores* de cada processador [9].

4. Resultados

A Figura 2 apresenta o projeto de software em UML gerado automaticamente pelas ferramentas do CMS-SEUP seguindo as etapas descritas na seção 2.

Tabela 6 - Componentes Eventos Utilizados (processos).

Componentes Eventos	T	C	P	W
Diagnóstico	7	3	3	3
Controlador Central	12	3	2	9
Gerenciador de Eventos	20	5	1	20

Tabela 7 - Estimativa de tamanho de memória.

Componentes	Tamanho	Restrição	Res.
Diagnóstico	450 (KB)	64000(KB)	OK
Ger. de Eventos	300 (KB)		
Cont. Central	300 (KB)		
Total	1050 (KB)	64000 (KB)	OK

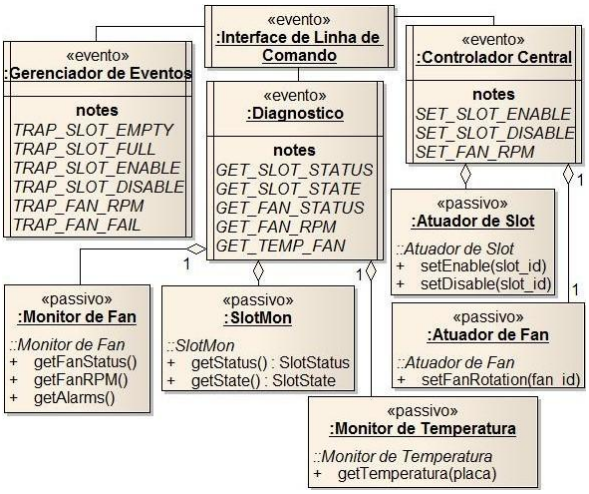

Figura 2 - Exemplo de Projeto de Software.

5. Conclusões

As ferramentas do CMS-SEUP tornam mais simples o reaproveitamento de partes de projetos anteriores; utilizam métodos formais de projeto e técnicas de síntese automática de código; permitem uma primeira verificação dos requisitos não-funcionais e reduzem o tempo de codificação.

Trabalhos futuros considerados são o aperfeiçoamento dos estimadores e a introdução de novos fatores de redundância e de uma etapa com exploração automática com novas opções de mapeamento e seleção de componentes.

Referências

[1] T. Sridhar, “Designing Embedded Communication Software”. Elsevier: CMP Books, 2003.

[2] “PICMG” http://www.picmg.org, Oct., 2009.

[3] X. Ke, K. Sierszecki, C. Angelov, “COMDES-II: A Component-Based Framework for Generative Development of Distributed Real-Time Control Systems”, Proc. of Embedded and Real-Time Computing Systems and Applications, 199-208, 2007.

[4] M. Oliveira, L. Brisolara, “Early Embedded Software Design Space Exploration Using UML-based Estimation”, Proc. of the 17th IEEE Intern. Workshop on Rapid System Prototyping, 2006.

[5] L. Shourong, W. Halang, L. Zhang, “A component-based UML profile to model embedded real-time systems designed by the MDA approach”, Proc. of Embedded and Real-Time Computing Systems and Applications, 563-566, 2005.

[6] R. Moreira, A. M. Tokarnia, “Ferramentas de Projeto baseado em Plataforma UML de Sistemas de Gerência de Configuração Multi-Placas”, Plano de Dissertação de Mestrado, FEEC, Unicamp, 11/2009.

[7] A. Burns, A. Wellings, “Real-Time Systems and Programming Languages”, Addison Wesley, 1997.

[8] “Inter-Process-Communication”, http://en.wikipedia.org/wiki/, Feb., 2010.

[9] “Linux Target Image Builder”, http://ltib.org. Feb., 2010.