Construindo Sistemas Embarcados com o OpenEPOS

- - Minicurso - -

Giovani Gracioli e Antônio Augusto Fröhlich

[1]Laboratório de Integração Software/Hardware (LISHA)
Universidade Federal de Santa Catarina (UFSC)
88040-900 – Florianópolis – SC – Brasil
{giovani,guto}@lisha.ufsc.br

Resumo. *Este mini-curso aborda a construção de sistemas embarcados sob a perspectiva do OpenEPOS. O mini-curso é dividido em duas partes: uma teórica e outra prática. A parte teórica apresenta os principais conceitos relacionados com o OpenEPOS. A parte prática foca em exercícios onde são implementadas aplicações utilizando o sistema operacional. As atividades práticas cobrem o desenvolvimento de aplicações multi-thread, uso da infraestrutura de mediadores de hardware e da pilha de comunicação, incluindo TCP, UDP e IP.*

1. Introdução

O projeto e a implementação de sistemas embarcados atualmente apresentam enormes desafios. As aplicações embarcados estão se tornando cada vez mais complexas conforme o avanço da indústria de semi-condutores, que tornou possível o uso de recursos computacionais que antes só eram encontrados em sistemas de computação de propósito geral. Se por um lado as aplicações estão mais complexas, do outro ainda existe a pressão do mercado para a entrega do produto de forma rápida. Neste contexto, a utilização de um sistema operacional embarcado e baseado em componentes ajuda na aceleração do projeto e desenvolvimento de aplicações embarcadas, bem como na reutilização dos componentes, tanto de hardware como de software. Este mini-curso apresenta a construção de sistemas embarcados sob a perspectiva do sistema operacional EPOS.

2. Embedded Parallel Operating System

O EPOS (*Embedded Parallel Operating System*)[1] [Fröhlich 2001, Marcondes et al. 2006] é um arcabouço baseado em componentes para a geração de ambientes dedicados de suporte de tempo de execução. O arcabouço do EPOS permite que programadores desenvolvam aplicações independentes de plataforma e ferramentas de análise permitem que componentes sejam adaptados automaticamente para atender aos requisitos destas aplicações particulares. Por definição, uma instância do sistema agrega todo suporte necessário para a sua aplicação dedicada, e nada mais.

O projeto modular do EPOS foi guiado pela metodologia Projeto de Sistemas Embarcados Guiado pela Aplicação (ADESD - *Application-Driven Embedded System Design*). A ADESD baseia-se nas conhecidas estratégias de decomposição de domínio por trás do Projeto Baseado em Famílias (FBD - *Family-Based Design*) e Orientação a Objetos (OO) como, por exemplo, análise de atributos comuns e variabilidade, para adicionar o conceito de identificação e separação de aspectos ainda nos estágios iniciais

[1]EPOS está disponível para download no endereço `http://epos.lisha.ufsc.br`.

do projeto [Fröhlich 2001]. Deste modo, a ADESD guia a engenharia de domínio para famílias de componentes, das quais dependências de cenários de execução são fatoradas na forma de aspectos e relacionamentos externos são capturados em um arcabouço de componentes. Esta estratégia de engenharia de domínio trata consistentemente algumas das questões mais relevantes do desenvolvimento de software baseado em componentes: reúso, gerenciamento da complexidade e composição.

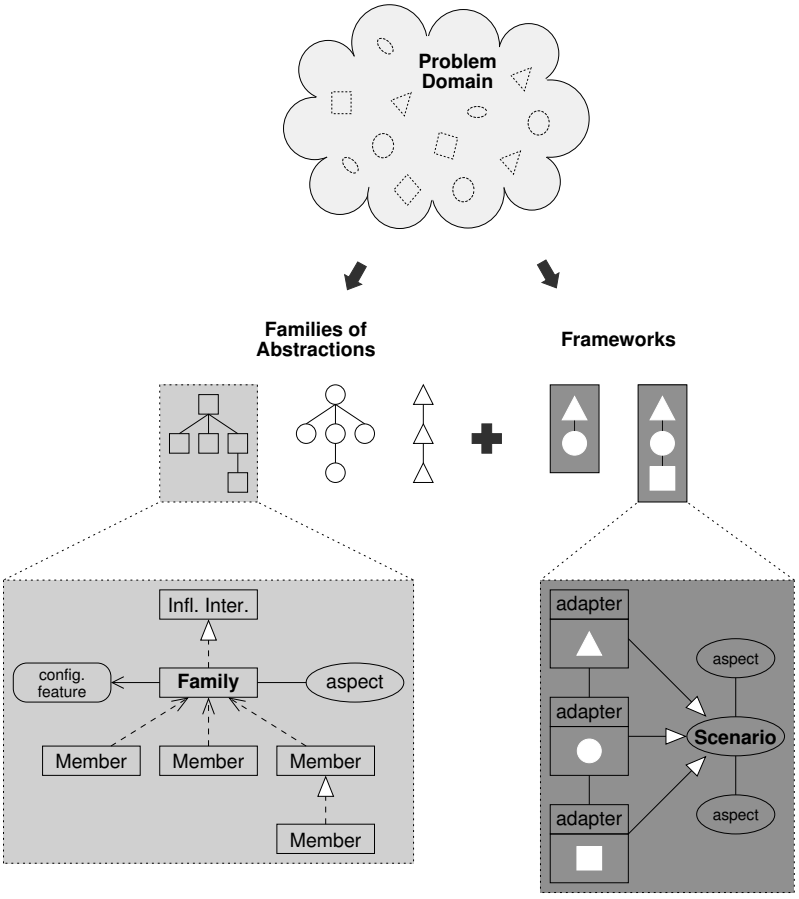

Figura 1. Processo de decomposição de domínio da ADESD.

A Figura 1 mostra o processo de decomposição de domínio no projeto de sistemas embarcados guiado pela aplicação. Abstrações são identificadas a partir do domínio do problema e organizadas em famílias, de acordo com suas características comuns. Dependências de cenário são modeladas como aspectos que podem ser aplicados através de adaptadores de cenário. Famílias de abstrações são visíveis para aplicações através de interfaces infladas, que exportam seus membros como um único "super-componente". As arquiteturas de sistema são capturadas em arcabouços de componentes, que são definidos em termos de aspectos de cenário.

Famílias de abstrações no EPOS representam abstrações tradicionais de sistemas operacionais e implementam serviços como gerenciamento de memória e de processos, sincronização de processos, gerenciamento de tempo e comunicação. Todas as funções dependentes de arquitetura são abstraídas através de mediadores de hardware, que exportam para as abstrações as funcionalidades necessárias através de interfaces independentes de plataformas [Polpeta and Fröhlich 2004]. Mediadores de hardware são funcionalmente

equivalentes aos *drivers* de dispositivos em SOs baseados em UNIX, mas não apresentam uma camada de abstração de hardware (HAL - *Hardware Abstraction Layer*) tradicional. Mediadores provêm uma interface entre os componentes do SO e o hardware através de técnicas de metaprogramação estática que "diluem" o código do mediador nos componentes em tempo de execução. Consequentemente, o código gerado não possui chamadas a métodos, nem camadas ou mensagens, atingindo uma maior portabilidade e reúso em comparação com as HALs tradicionais

No EPOS, processos são gerenciados pelas abstrações `Thread` e `Task`. Cada thread armazena seu contexto em sua própria pilha. O mediador de hardware `CPU` define o conjunto de dados que precisa ser armazenado para um fluxo de execução e, deste modo, cada arquitetura define seu próprio contexto. Já o gerenciamento de memória é realizado pela família de mediadores `MMU` (*Memory Management Unit*). A abstração `Address Space` é um contêiner para "fatias" de memória física chamados de `Segments`. O membro `Flat Address Space` define um modelo de memória no qual endereços lógicos e físicos coincidem, eliminando a necessidade de hardware para `MMU`. Em plataformas que não dispõem de `MMU`, um mediador para `MMU` simplesmente mantém o contrato de interface com o `Flat Address Space`, realizando implementações vazias de métodos quando necessário.

Tempo no EPOS é tratado pela família de abstrações `Timepiece`, composta pelas abstrações `Clock`, `Alarm` e `Chronometer`. A abstração `Clock` é responsável por armazenar o tempo corrente e está disponível apenas em sistemas que possuem dispositivos de relógios de tempo-real (`RTC`). A abstração `Chronometer` é utilizada para realizar medições de tempo com alta precisão e a abstração `Alarm` é utilizada para gerar eventos, acordar uma Tarefa/Thread ou chamar uma função após um tempo definido. A família de abstrações `Timepiece` é suportada pelos mediadores `Timer`, `TimeStamp Counter (TSC)` e `Real-Time Clock (RTC)` [Fröhlich et al. 2011].

A família de abstrações `Synchronizer` provê mecanismos que garantem a consistência de dados em ambientes com processos concorrentes. O membro `Mutex` implementa um mecanismo de exclusão mútua que entrega duas operações atômicas: `lock` e `unlock`. O membro `Semaphore` implementa, como o próprio nome diz, um semáforo, que é uma variável inteira cujo valor apenas pode ser manipulado indiretamente através das operações atômicas `p` e `v`. O membro `Condition` realiza uma abstração de sistema inspirada no conceito de variável de condição, que permite a uma thread esperar que um predicado se torne válido.

Controle de entrada e saída (I/O) de dispositivos periféricos é disponibilizado no EPOS pelo mediador de hardware correspondente. O mediador `Machine` armazena as regiões de I/O e trata o registro dinâmico de interrupções. O mediador `IC` (*Interrupt Controller*) trata a ativação ou desativação de interrupções individuais. Para lidar com as diferentes interrupções existentes em diferentes plataformas e contextos, EPOS atribui um nome e uma sintaxe independente de plataforma a interrupções pertinentes ao sistema (e.g., interrupção de timer, interrupção de conversão completa no ADC).

3. Conclusão

Este documento brevemente discute os principais conceitos e componentes que fazem parte do sistema operacional EPOS. O foco do mini-curso está no desenvolvimento de

sistemas embarcados sob a perspectiva do EPOS.

Com a utilização de um sistema operacional orientado à aplicação, é possível construir sistemas embarcados que possuem somente o suporte necessário para a sua aplicação dedicada, sem nenhum sobrecusto adicional. Além disso, o mini-curso é dedicado a demonstrar a utilização e a eficiência do uso de técnicas avanças de engenharia de software na construção de sistemas dedicados.

Referências

Fröhlich, A. A. (2001). *Application-Oriented Operating Systems*. Number 17 in GMD Research Series. GMD - Forschungszentrum Informationstechnik, Sankt Augustin.

Fröhlich, A. A., Gracioli, G., and Santos, J. F. (2011). Periodic timers revisited: The real-time embedded system perspective. *Computers Electrical Engineering*, 37(3):365–375.

Marcondes, H., Hoeller, A. S., Wanner, L. F., and Fröhlich, A. A. (2006). Operating systems portability: 8 bits and beyond. In *ETFA '06: IEEE Conference on Emerging Technologies and Factory Automation*, pages 124 –130.

Polpeta, F. V. and Fröhlich, A. A. (2004). Hardware mediators: a portability artifact for component-based systems. In *In Proceedings of the International Conference on Embedded and Ubiquitous Computing*, volume 3207 of *Lecture Notes in Computer Science*, pages 45–53, Aizu, Japan. Springer Berlin / Heidelberg.