Ferramenta de Suporte ao Projeto Automatizado de Sistemas Computacionais Embarcados

Rafael L. Cancian - DAS/UFSC
Marcelo R. Stemmer - DAS/UFSC
Alexandre Schulter - INE/UFSC
Antônio A. M. Fröhlich - INE/UFSC

Sumário

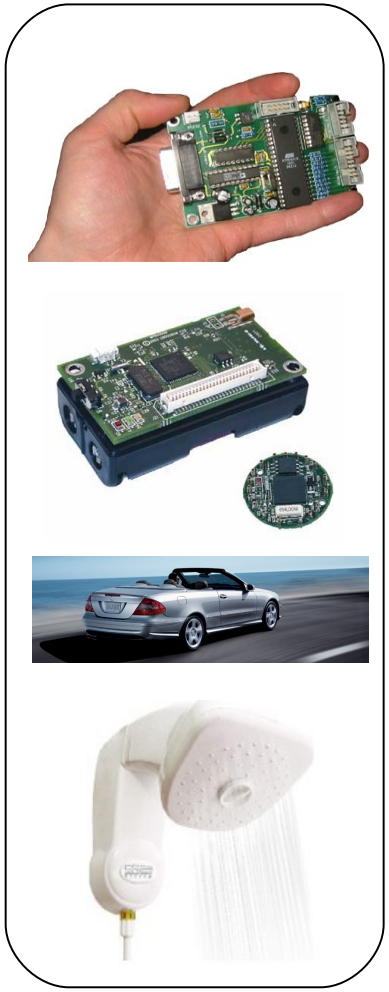
- Introdução
- Projeto de Sistemas Orientado à Aplicação
- EPOS
- O Problema
- A Ferramenta
- O Protótipo
- Estudo de Caso
- Considerações Finais

Introdução (1/3)

- Sistemas Embarcados (SE) são extensivamente usados em muitos setores:
  - Indústria;
  - Automóveis;
  - Utensílios domésticos;
  - Equipamentos pessoais;
  - Virtualmente qualquer dispositivo que inclui eletrônica.
- 99% dos microprocessadores produzidos atualmente são usados em Sistemas Embarcados (Pop, 2005).
- Os avanços na tecnologia têm permitido o aumento da complexidade dos Sistemas Embarcados.

Introdução (2/3)

- A abordagem de System-on-a-Chip (SoC)
  - É um compromisso entre complexidade e custo.
- Dispositivos Lógicos Programáveis (ex FPGA)
  - Permitem o desenvolvimento de projetos complexos em puco tempo;
  - Projeto é similar ao desenvolvimento de software.
- Fazer um SoC de uma FPGA não é fácil
  - Esforços têm focado em ferramentas para auxílio, seleção e configuração de componentes de hardware – Intellectual Property (IP) blocks.
- Entretanto, aplicações complexas precisam de suporte de execução (software), não apenas IPs (hardware).

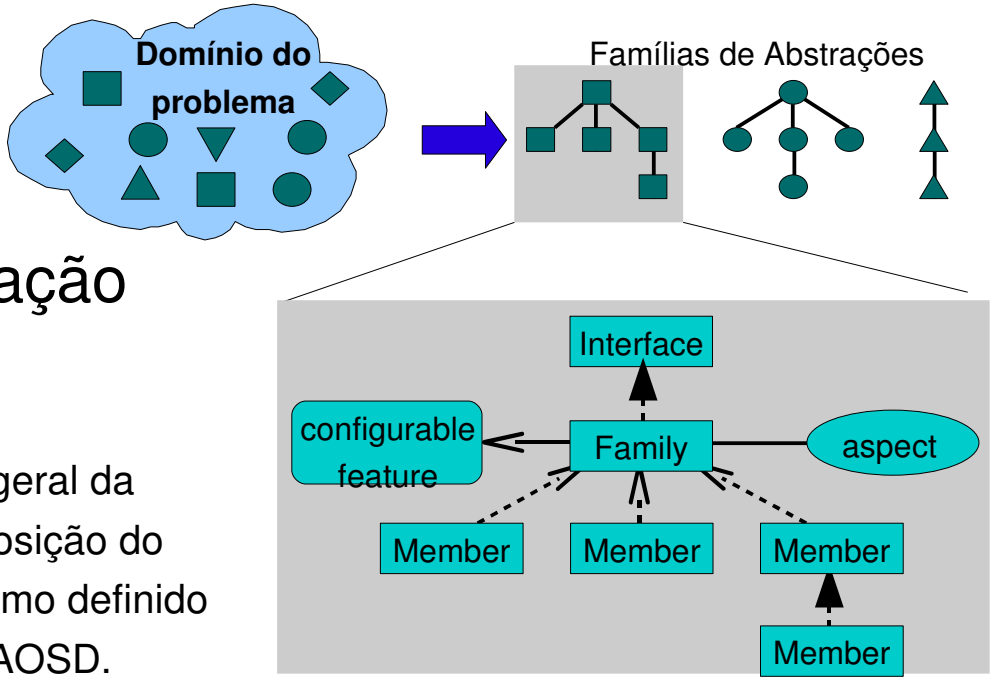


Introdução (3/3)

- Nosso Grupo de Pesquisa tem explorado metodologias e ferramentas para desenvolvimento eficiente de Sistemas Embarcados (SE)
  - **Objetivo**: suportar o desenvolvimento de SE como agregados de componentes reutilizáveis.
  - **Desafios:** prover transparência arquitetural para o suporte de execução e adequação à aplicação-alvo.
    - Aplicações podem executar em plataformas muito diferentes: hardware dedicado, microcontroladores de 8 bits a processadores de 32 bits.
    - Para permitir a portabilidade:
      - Os componentes que irão compor o SE devem apresentar a mesma interface em todos os casos;
      - Os componentes são os mesmos: interface e comportamento;
      - A arquitetura do software é que difere: Deve-se adaptar componentes a cenários de execução.

Application-Oriented System Design

- Propõe estratégias para definir componentes que representam entidades significantes em diferentes domínios.
- Os principais conceitos da AOSD são
  - Famílias de abstrações independentes de cenário de execução;
  - Adaptadores de cenário;
  - Interfaces infladas.
- Guia o desenvolvedor, do projeto à implementação final do sistema.



Visão geral da decomposição do domínio como definido pela AOSD.

EPOS

- EPOS é um SO concebido através da AOSD
  - Cada instância do EPOS é gerada com base numa aplicação-alvo, e possui unicamente os componentes necessários e suficientes para suportar tal aplicação.
    - Ou seja, é um Sistema operacional orientado à aplicação
  - Pode ser usado em diferentes ambientes:
    - possui transparência arquitetural.
  - Combina os conceitos de:
    - Family-Based Design;
    - Aspect-Oriented Programming;
    - Object-Oriented Design;
    - Static Meta Programming.

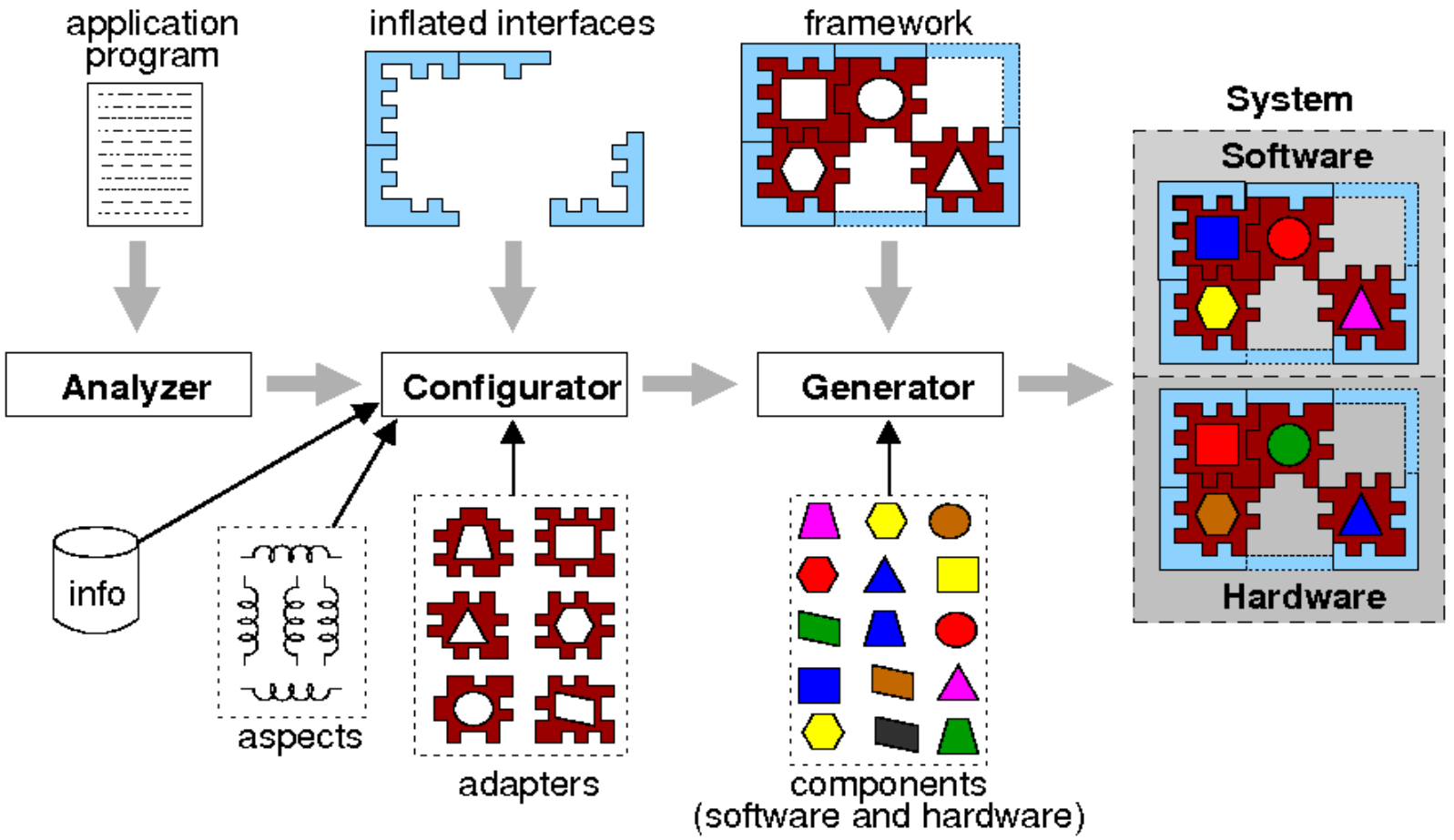
O Problema

- Suporte de execução e sistemas operacionais de propósito geral não são adequados para aplicações embarcadas. (AOSD é uma solução útil para esse problema)
- Application-Oriented Operating Systems (AOOS) são desenvolvidos para uma aplicação-alvo
  - Compostos apenas dos componentes selecionados para satisfazer exatamente os requisitos da aplicação.
- Como suportar o desenvolvimento de AOOS? Como automatizar parte do processo para obter maior produtividade?
  - Com uma ferramenta que auxilie o desenvolvedor a compor o sistema embarcado, o que inclui identificar, selecionar, configurar, adaptar e compor esses componentes.

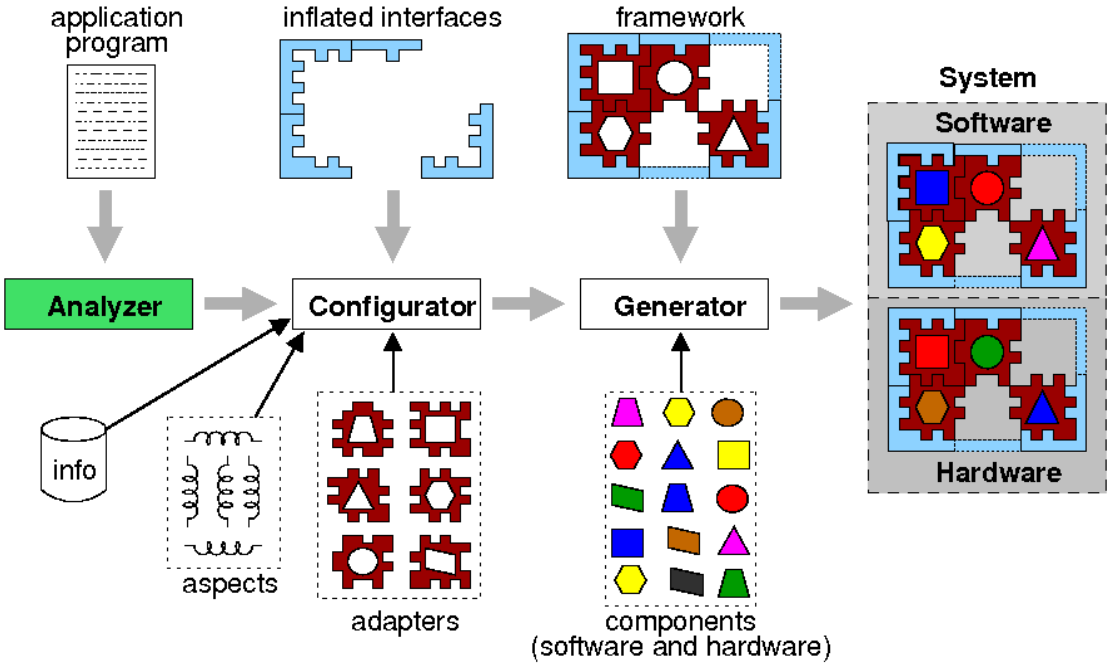
A Ferramenta

- Funcionalidades da Ferramenta
  - Análise dos requisitos da aplicação (automático);
  - Sugestões de componentes (automático);
  - Seleção de componentes (semi-automático);
  - Configuração e composição de componentes (semi-automático);
  - Generação do sistema - software e hardware (automático);
  - Estimação de custo (automático);
  - Validação da configuração (semi-automático).
- Outros Requisitos da Ferramenta
  - Possuir interface gráfica;
  - Fornecer feedback para o desenvolvedor;
  - Permitir a automação do processo.

A Ferramenta – Visão Geral

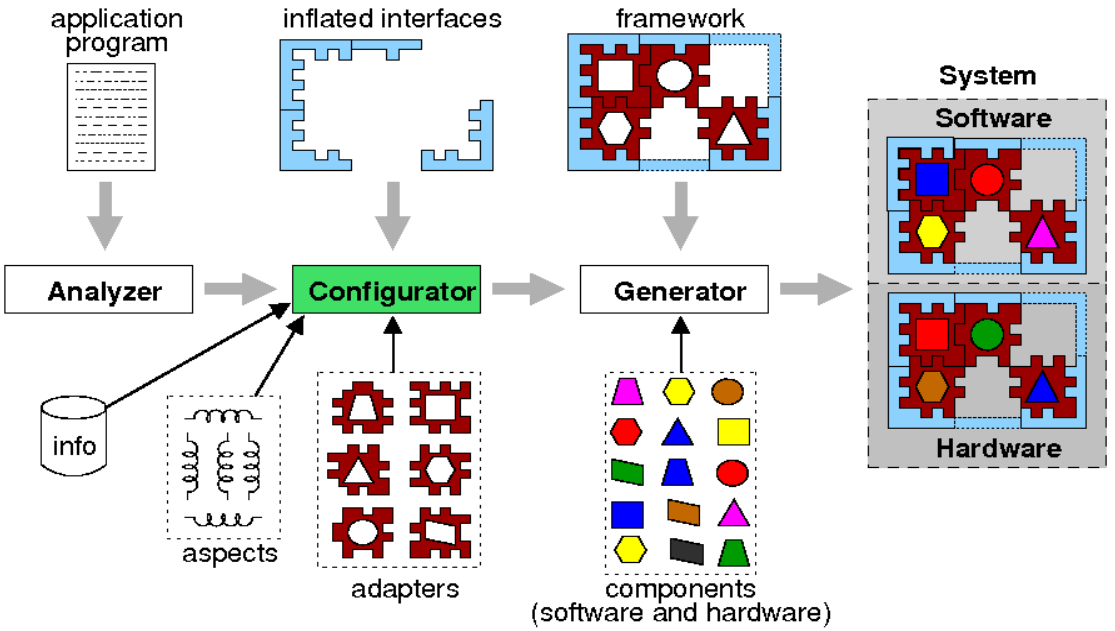


A Ferramenta - Analisador



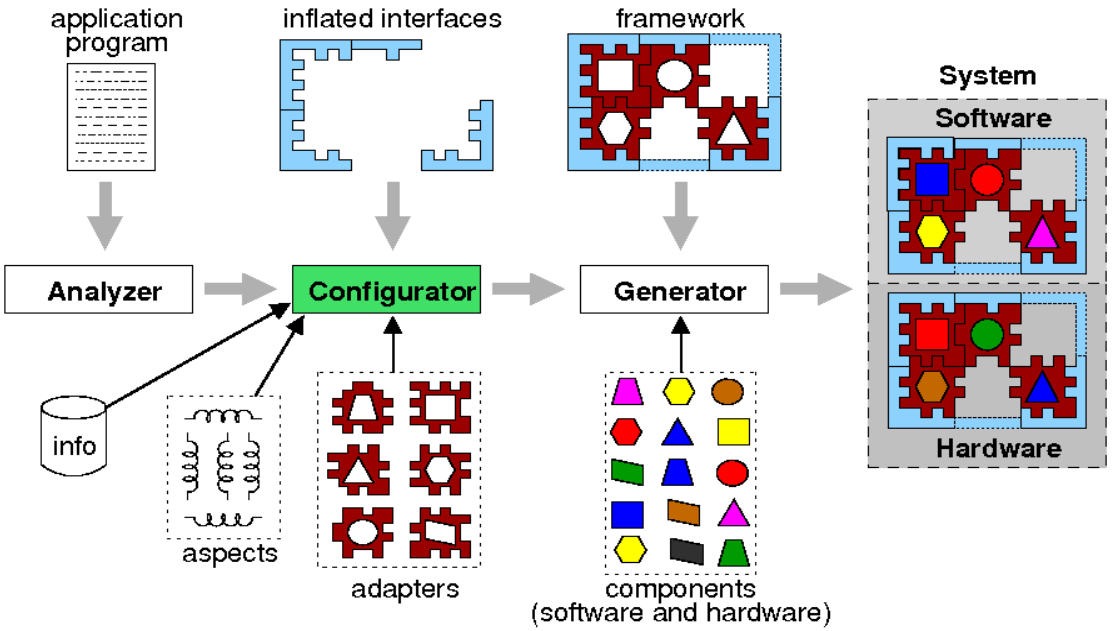
- O desenvolvedor escreve uma aplicação baseada na API do EPOS (interfaces dos componentes) e submete o código-fonte ao Analisador;
- O analisador cria uma especificação de requisitos que inclui métodos, tipos e constantes usadas pela aplicação;
- O Analisador seleciona componentes e/ou apresenta opções ao desenvolvedor, bem como estimativas de custos.
- O Analisador repassa uma série de dependências da aplicação ao Configurador.

A Ferramenta - Configurador (1/2)



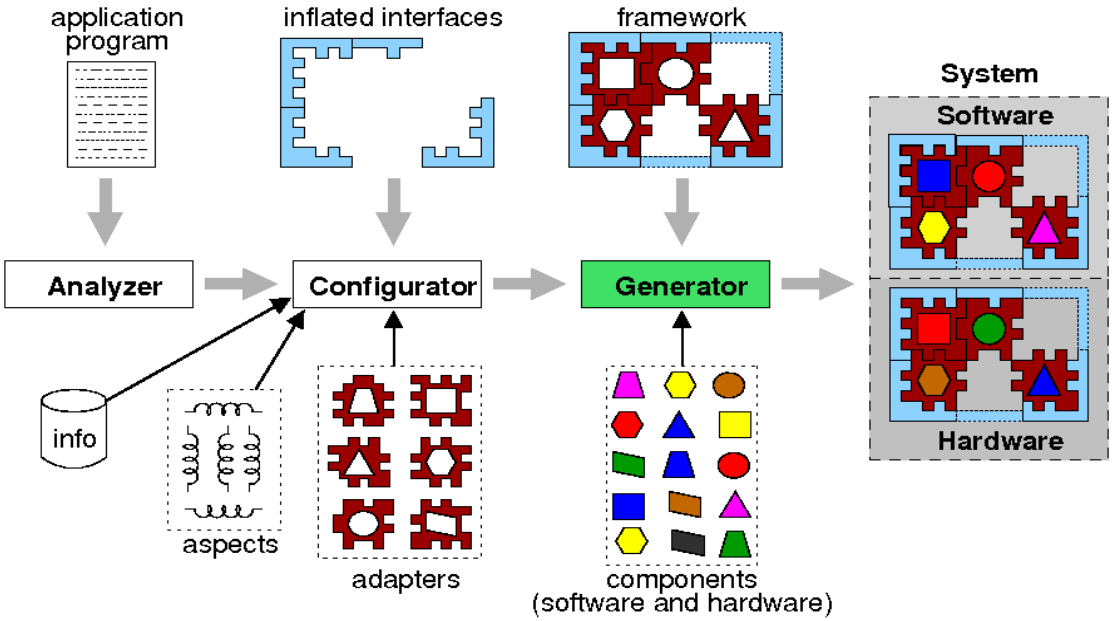
- Os componentes escolhidos com o Analisador são adicionados a uma configuração;
- O Configurador constrói uma árvore de dependências e mantém informações sobre:
  - Dependências da aplicação;
  - Dependências entre componentes;
  - Regras de composição de componentes;

A Ferramenta - Configurador (2/2)



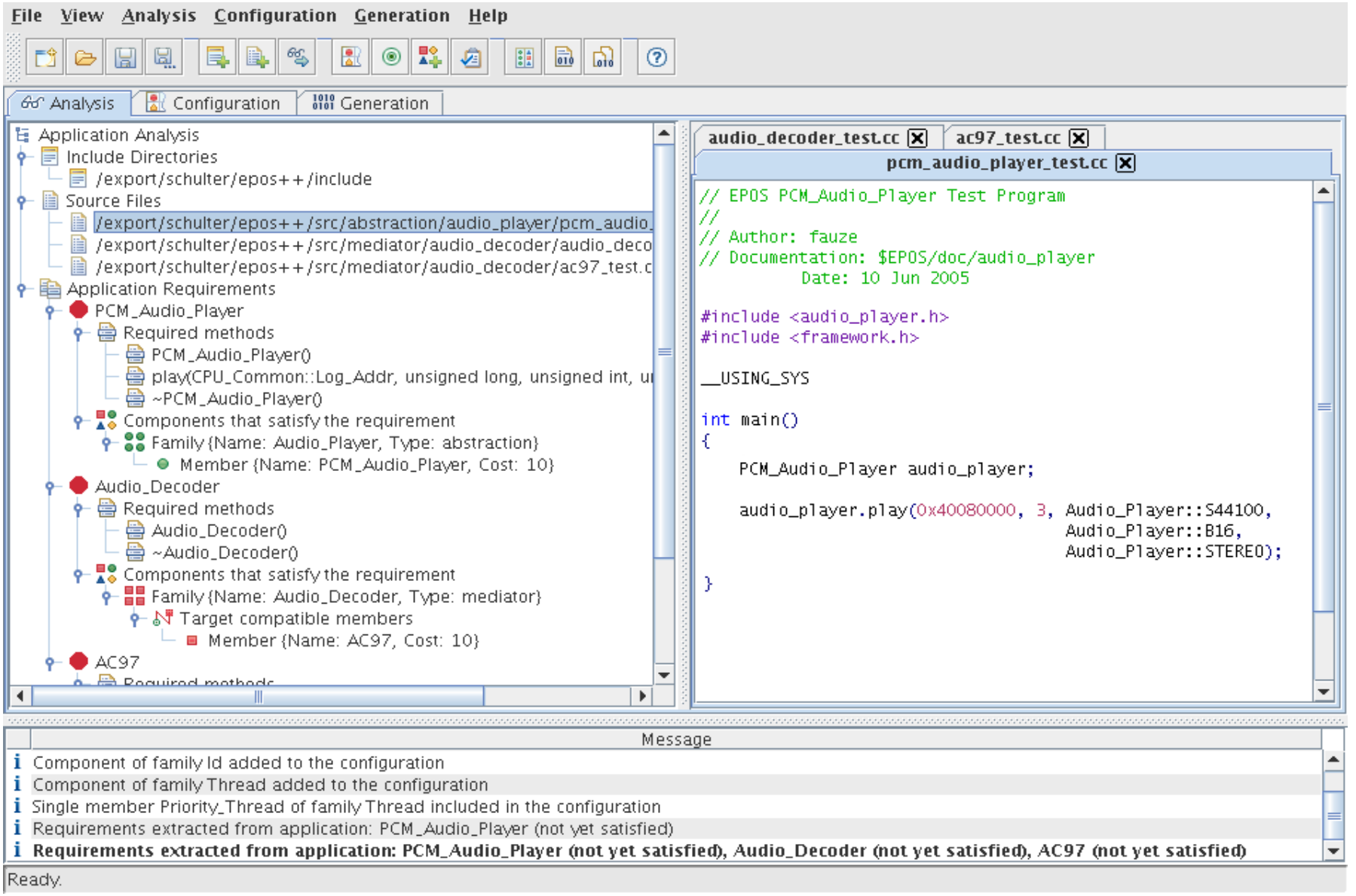
- Cria um conjunto de especificações que representam a configuração;
- Liga a interface usada pela aplicação a componentes específicos;
- Ativa adaptadores de cenário que satisfazem restrições impostas por:
  - Aplicação-alvo;
  - Cenário de execução configurado.
- Liga os mediadores de hardware aos *IP blocks* associados a eles.

A Ferramenta - Gerador

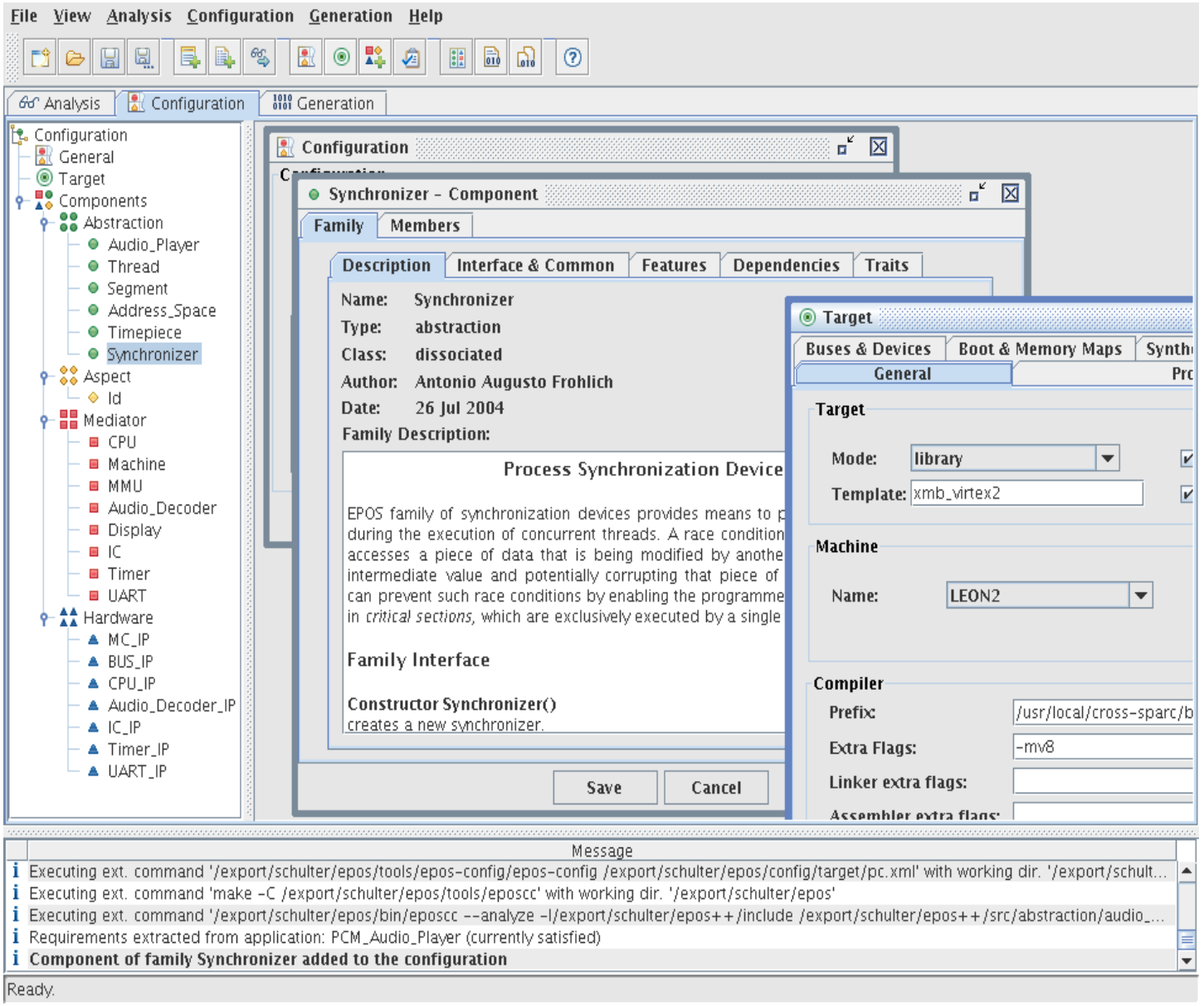


- Traduz as especificações recebidas do Configurador em parâmetros para um repositório de componentes meta-programados;
- Invoca a compilação da instância do sistema gerado (software);
- Quando hardware também precisa ser sintetizado,
  - Produz os arquivos de configuração da síntese para uma ferramenta de terceiros (ex. Xilinx, Altera);
  - Invoca o sintetizador externo.

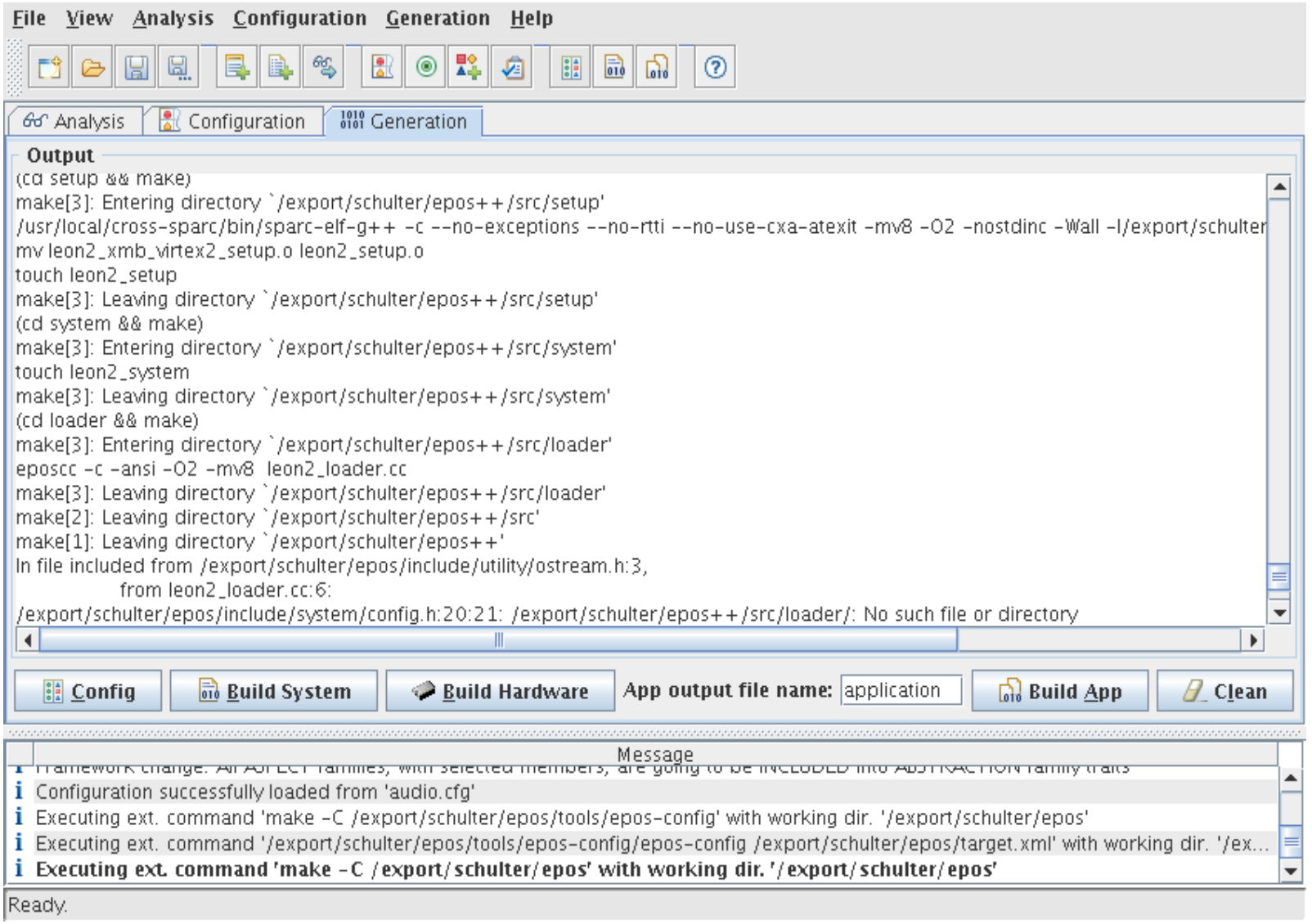
Protótipo - Analisador



Protótipo - Configurador

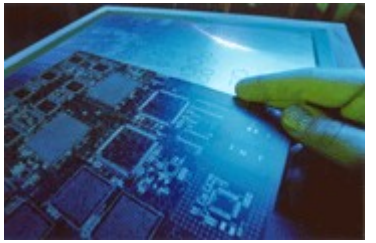


Protótipo - Gerador



Estudos de Caso

- Diferentes tipos de aplicações embarcadas já foram desenvolvidas e testadas:
  - Multimídia;
  - Controle automotivo;
  - Redes de sensores em fio;
  - Sistemas móveis de baixo consumo de energia.



- Para diferentes plataformas/arquiteturas:
  - AVR;
  - MIPS;
  - SparcV8;
  - PPC405;
  - IA32.

Estudo de Caso – Um *Audio Player*

- Realizado com o Sistema Operacional EPOS;
- Placa de prototipação da Xilinx como plataforma-alvo:
  - LEON2 soft-core processor
    - Sintetizado com outros *IP blocks* para criar um *System-on-a-Chip*;
  - Diversos componentes:
    - Memory Management Unit (MMU);
    - Floating Point Unit (FPU);
    - Network Interface Card (NIC);
    - Audio Decoder;
    - Video Decoder;
    - Display Controller, e outros.

Estudo de Caso – Um *Audio Player*

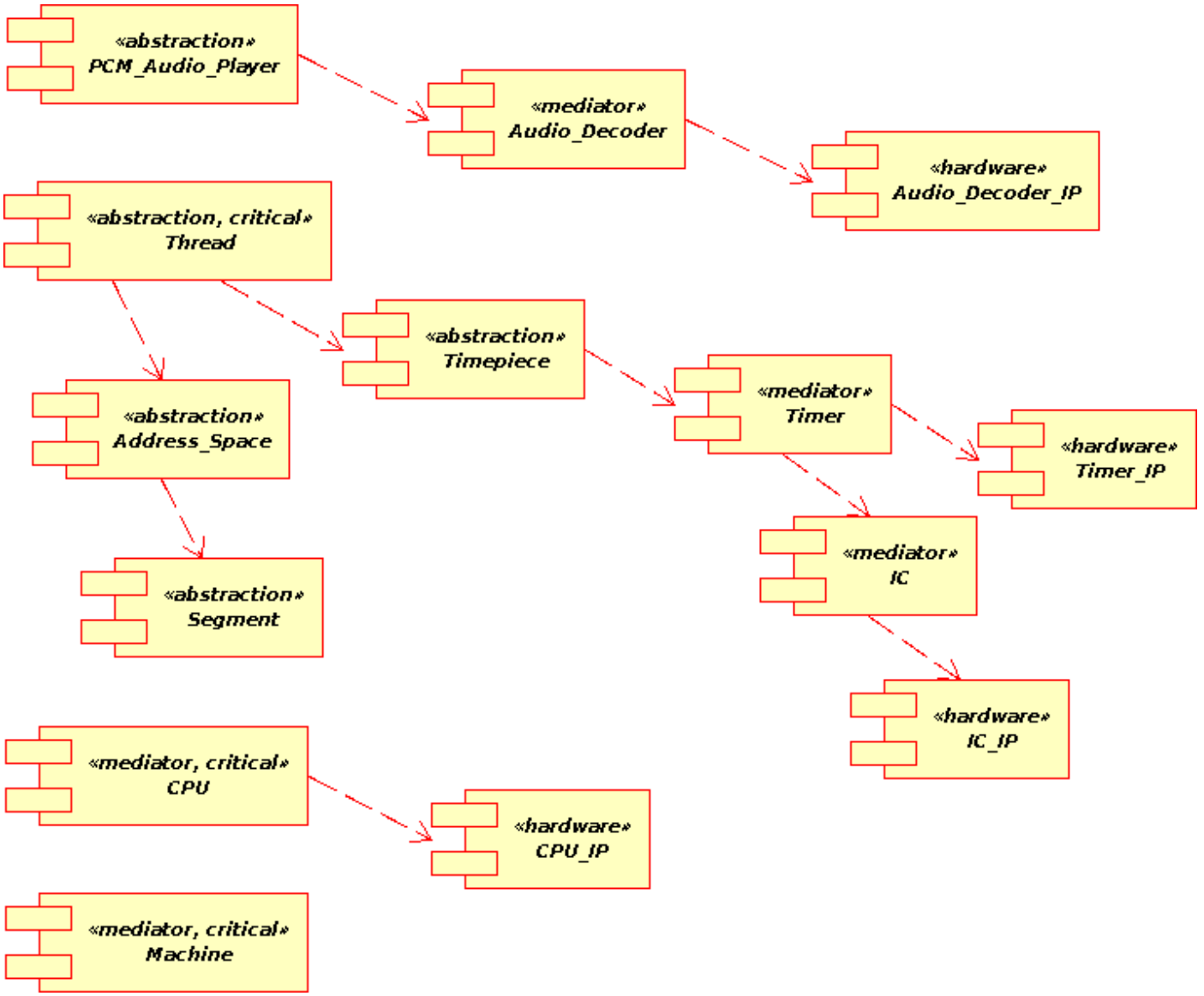
```
// PCM_Audio_Player Test Program

#include <audio_player.h>
#include <framework.h>

__USING_SYS

int main()
{
    PCM_Audio_Player audio_player;
    audio_player.play(0x40080000, 3, Audio_Player::S44100,
                      Audio_Player::B16, Audio_Player::STEREO);
    return 0;
}
```

Estudo de Caso – Um *Audio Player*



Considerações Finais (1/2)

- Consideramos que as soluções existentes não tratam ou resolvem adequadamente
  - Transparência arquitetural;
  - Desempenho;
  - Geração de sistemas embarcados adaptados à aplicação.
- Nossa ferramenta auxilia desenvolvedores na configuração e geração de software e hardware para Sistemas Embarcados.
  - Baseados numa coleção de componentes reutilizáveis (AOSD).

Considerações Finais (2/2)

- O protótipo atual da ferramenta
  - Provê automação parcial;
  - Desenvolvedor pode concentrar-se na aplicação, e não no suporte necessário, permitindo:
    - Processo de desenvolvimento mais rápido e confiável;
    - Menor *time-to-market* para produtos comerciais.
  - Geração fácil de sistemas operacionais otimizados resulta em
    - Ganhos de desempenho;
    - Otimização na utilização de recursos.
  - Outros sistemas além do EPOS poderiam ser configurados e gerados com essa ferramenta.

Trabalhos Futuros

- Automação do processo pode ser melhorada
  - Automação completa pode ser conseguida por algumas técnicas de otimizaçao – exige melhorias no modelo de custo.
- Incluir a detecção de componentes redundantes ou não adequados (incluídos manualmente pelo projetista) na configuração.
- Incluir não apenas requisitos da aplicação, mas requisitos de projeto
  - Área de silício, frequência de operação, consumo de energia, etc
- Incluir projeto conjunto de hardware e software e seu particionamento.

Contatos

Laboratório de Integração de Software e Hardware

Prof. Dr. Antônio Augusto Fröhlich (head)
M.Sc. Alexandre Schulter (former-member)
M.Sc. Rafael L. Cancian (member)

UFSC/CTC/LISHA
Caixa Postal 476
Campus Universitário - Trindade
88040-900 - Florianópolis - SC - Brazil
+55 (48) 3721-9516
lisha@lisha.ufsc.br

http://www.lisha.ufsc.br