Mineração de Componentes para a Revitalização de Softwares Embutidos

Marcelo A. Ramos

Centro de Desenvolvimento de Aplicações

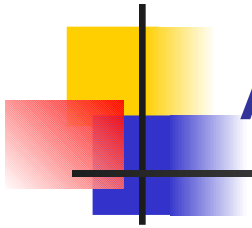
VeriFone do Brasil

Rosângela A. D. Penteado

Departamento de Computação

Universidade Federal de São Carlos

SBCARS - 2007



Agenda

- Fundamentos
- Mineração de Componentes
- Revitalização de Softwares Embutidos
- Estudo de Caso
- Considerações Finais

Fundamentos

“Sistemas embutidos são sistemas computacionais especializados, hardware e software, que integram sistemas com funcionalidade mais ampla, realizando tarefas específicas e pré-definidas ”

[Vahid e Givargis]

Fundamentos

- Desenvolvimento de Software Embutido
  - Técnicas Individualizadas;
  - Conjunto de Requisitos Específicos;
    - Limitação de Memória;
    - Consumo de Energia;
    - Variações do Hardware etc.
  - Amplo domínio de aplicações.

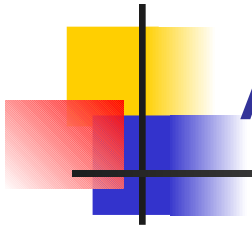
Apenas 2% dos processadores produzidos atualmente são empregados em computadores propriamente ditos, como PC's etc. Os 98% restantes equipam sistemas embutidos.

[TENNENHOUSE]

Fundamentos

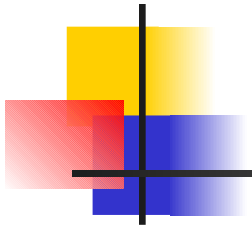
- Desafios para o Desenvolvimento de Novos Produtos
  - Reduzir Custos
  - Minimizar Riscos
  - Antecipar Prazos
  - Otimizar o Uso de Recursos
  - Aprimorar a Qualidade

- Combinação de Técnicas e Tecnologias Apropriadas
  - Manutenção e Reengenharia ..... [Revitalização]
  - Reuso ..... [Eficiência]
  - Linha de Produtos de Software ..... [Produtividade]



Agenda

- Fundamentos
- Mineração de Componentes
- Revitalização de Softwares Embutidos
- Estudo de caso
- Considerações finais



Mineração de Componentes

- Recuperação de Ativos em 4 passos:
  - Reunir as informações preliminares;
  - Decidir pela mineração e definir estratégia;
  - Obter entendimento técnico detalhado dos ativos existentes;
  - Recuperar os ativos.

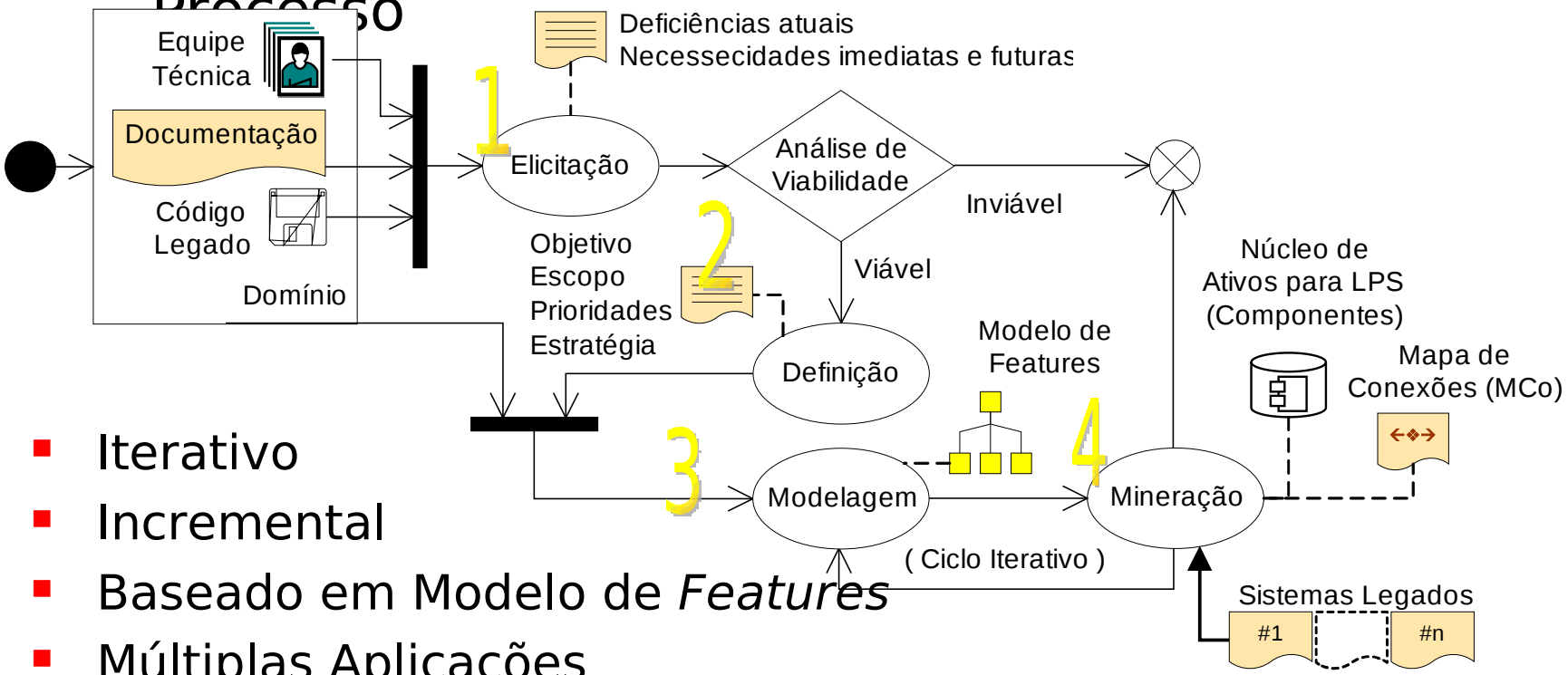
(Bergey, O'Brien e Smith, 2000)

"É essencial a criação de um Modelo de *Features* para apoiar a realização dessa tarefa"

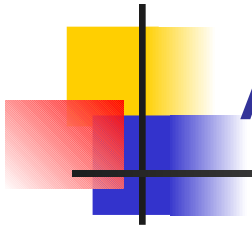
(Bosch e Ran, 2000)

Mineração de Componentes

- Modelo de Processo



- Iterativo
- Incremental
- Baseado em Modelo de *Features*
- Múltiplas Aplicações Concomitantemente



Agenda

- Fundamentos
- Mineração de Componentes
- Revitalização de Softwares Embutidos
- Estudo de caso
- Considerações finais

Revitalização de Softwares Embutidos

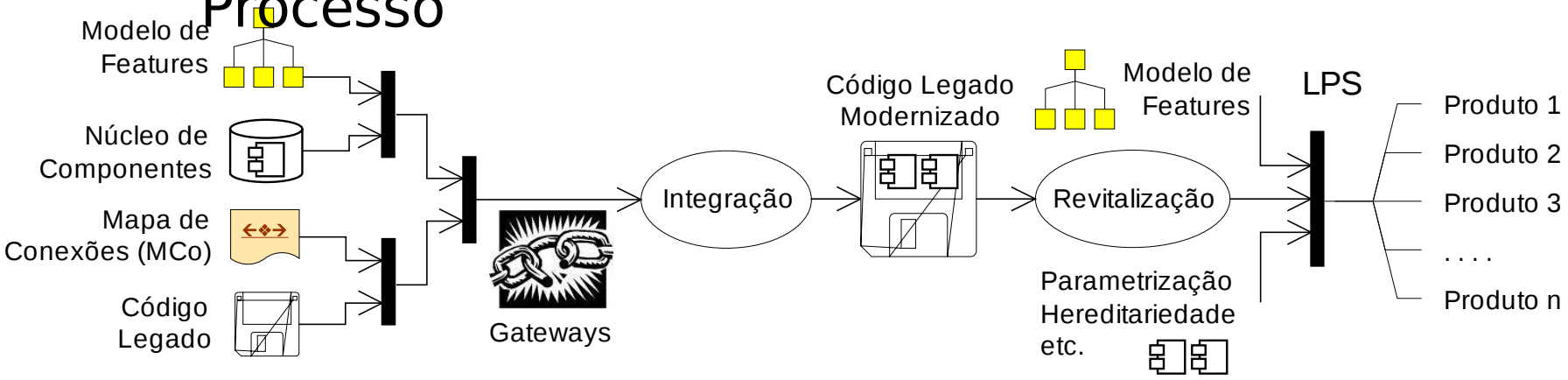
Classificações do reuso de software (Bosch, 2000)

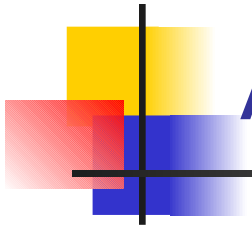
Top-down ou *Bottom-up* e Oportunista ou Planejado

- *Bottom-up* - Componentes reutilizáveis, uma vez desenvolvidos ou minerados, são adicionados a uma coleção, possivelmente grande, de artefatos de software, da qual podem ser extraídos para a solução de problemas recorrentes.
- Planejado - Artefatos reutilizáveis são produzidos mediante esforço para prover abstrações corretas e níveis corretos de variabilidades para um produto planejado.

Revitalização de Softwares Embutidos

- Modelo de Processo





Agenda

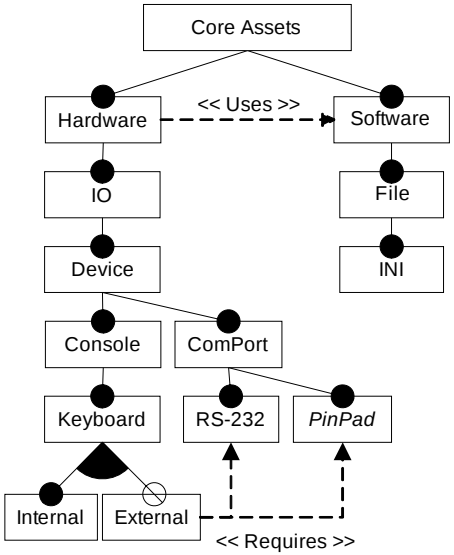
- Motivação
- Fundamentos
- Reengenharia Iterativa Orientada a Características
- Estudo de caso
- Considerações finais

Estudo de Caso

Terminais POS (***P**oint **O**f **S**ale*)

São equipamentos pequenos, leves, portáteis e versáteis, com um conjunto reduzido de componentes periféricos padronizados, controlado por um sistema embutido microprocessado. Tais características direcionam seu uso para a realização de Transferências Eletrônicas de Fundos (TEF) com cartões de pagamento.

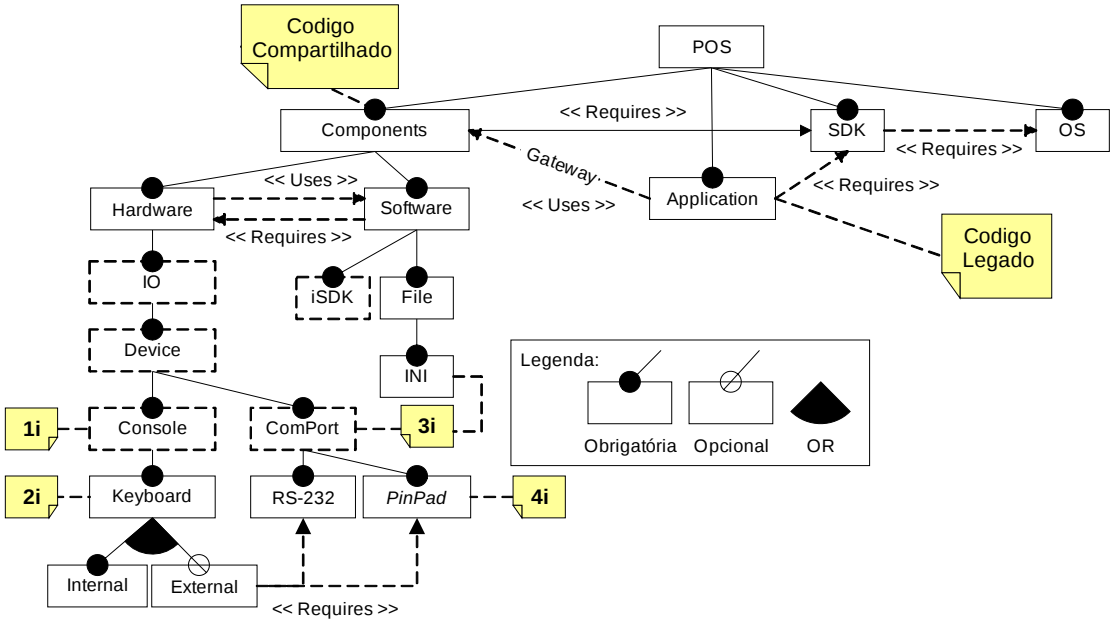
Estudo de Caso



- Terminais POS (*Point of Sale*);
- Três Sistemas Legados Similares;
- Apoio às Variabilidades do Hardware;
- Decomposição Estrutural;
- Construção do Modelo de Características;
- Construção do Mapa de Conexões (MCo);
- Mapeamento Característica-Componente;
- Construção do *Gateway;*
- Reconstrução das Aplicações Legadas;

Estudo de Caso

- Modelo de *Features*



1i O SO redireciona as leituras do Console para o Teclado e as escritas para a Tela do POS.

2i Deve permitir escritas e leituras

3i Deve permitir a configuração de parâmetros externamente à aplicação

4i A porta física de E/S para *PinPad* Externo e Interno é única (compartilhada), mas pode possuir endereços lógicos diferentes (e.g. COM2, COM5, etc). Quando os endereços lógicos coincidem é necessário um procedimento especial para a selecão correta do dispositivo de E/S desejado. Caso contrário, essa seleção é feita automaticamente pelo S.O. com base no endereço lógico utilizado.

Estudo de Caso

Característica:		**IO → Device → Console → Keyboard**			
Função	**Grupo**	**Parâmetros**	**Retorno**	**Comentários**	**Aplicações**
KBHIT	1	Nenhum	Status: bool	Retorna TRUE se houver tecla pressionada.	#1, #2, #3
get_char	2	Nenhum	Tecla : int	Aguarda o pressionamento de uma tecla e retorna seu valor	#1, #2, #3

[Limitações]

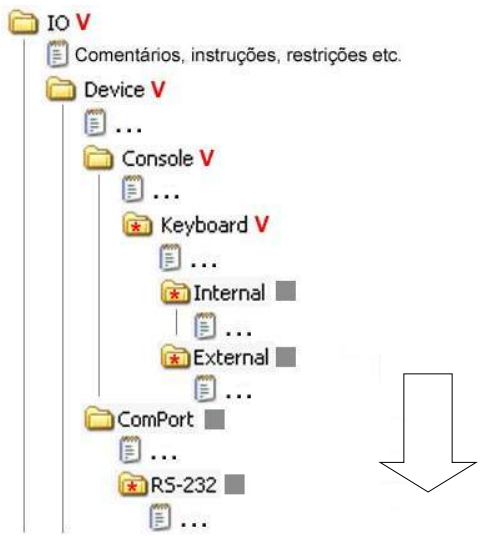
- O sistema operacional não permite escritas no *buffer* do teclado interno;
- O teclado externo não possui um buffer de teclas provido pelo sistema operacional.

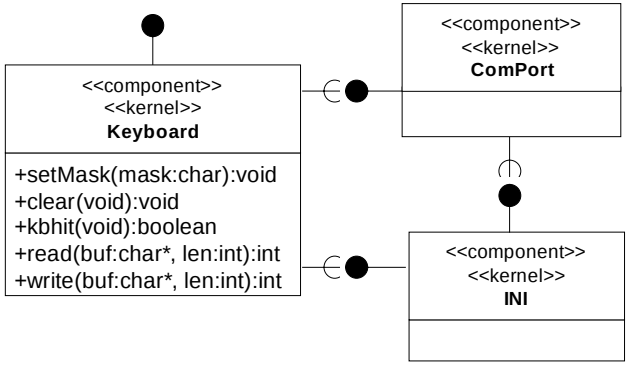
[Novos Requisitos]

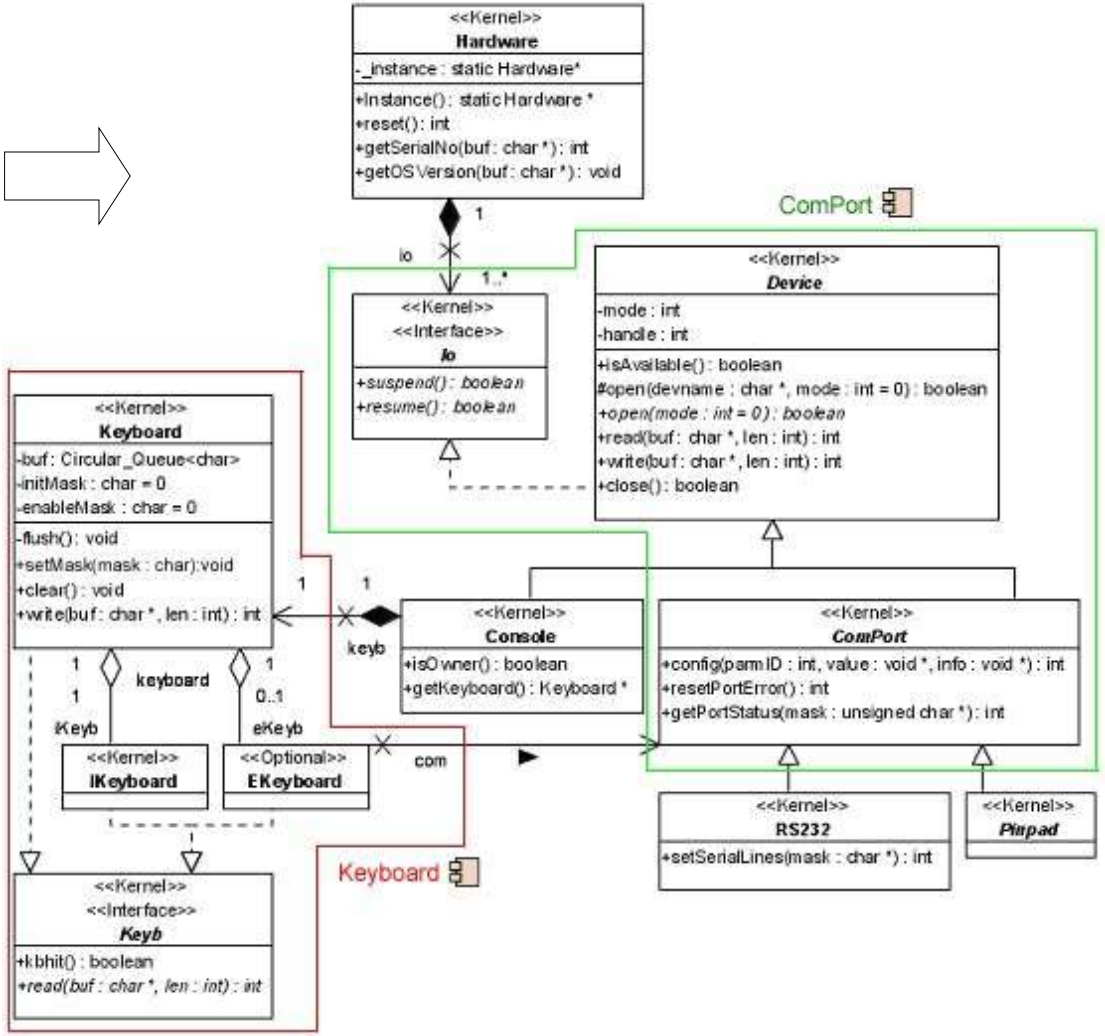
- Os teclados, interno e externo, devem compartilhar um *buffer* comum de teclas, provido pelo componente, o qual deve permitir operações de escrita e leitura;
- O componente deve incluir um controle dinâmico de liga/desliga para os teclados ativos;
- A fim de flexibilizar o seu uso, a conexão do teclado externo pode ser feita nas portas RS-232 ou *PinPad* do terminal POS, com parâmetros de comunicação configuráveis externamente à aplicação.

Mapa de Conexões (MCo)

Estudo de Caso







Componentes-

Estudo de Caso

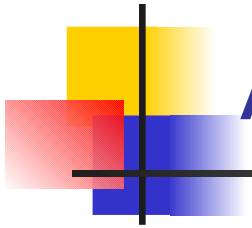
```
#ifndef _GATEWAY_C_
#define _GATEWAY_C_

#include <string>
using namespace std;

#include "Console.h"
#include "Keyboard.h"
#include "Sdk.h"
#include "Gateway.h"

namespace Hw
{   class Console   ;
    class Keyboard  ;
}
// Keyboard variability control through configuration file

Hw::Console*    _c = NULL ;
Hw::Keyboard*   _k = NULL ;

void initConsole(void)
{
        _c = new Hw::Console("HARDWARE.INI");
    _k = _c->getKeyboard();
}
int _G_kbhit (void)
{       if  (   !_k )initConsole( ) ;
    return  (int)_k->kbhit( )       ;
}
int _G_getchar(void)
{
    char    ch                      ;
    while(!_G_kbhit())              ;
                _k->read(&ch, 1)    ;
    return  (int)    ch             ;
}

#endif/*_GATEWAY_C_*/
```

```
[CONSOLE]
KEYBOARD=EXT-PINPAD
[EXT-RS232]
# <1 = COM1, 2 = COM2>
COM=1
# MODE: <protocol>, <baudrate>, <data length>, <parity>, <stop bits>, <stx_char>, <etx_char>
MODE=CHAR,115200,8,N,1
[EXT-PINPAD]
COM=2
MODE=CHAR,19200,7,E,1
```

Hardware.ini

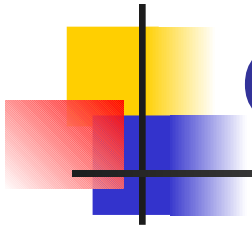
Gateway

Código Legado



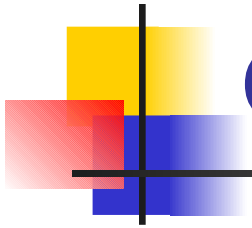
Agenda

- Motivação
- Fundamentos
- Reengenharia Iterativa Orientada a Características
- Estudo de caso
- Considerações finais



Considerações Finais

- Resultados
  - Revitalização incremental com baixo risco;
  - Implantação com boa aceitabilidade;
  - Simplicidade e Agilidade;
  - Melhoria do reuso;
  - Uso de paradigmas mais modernos;
  - Código legado parcialmente modernizado;
  - Documentação parcialmente recuperada;
  - Núcleo de componentes reutilizáveis;
  - Sistemas legados revitalizados concomitantemente;
  - Família de produtos.

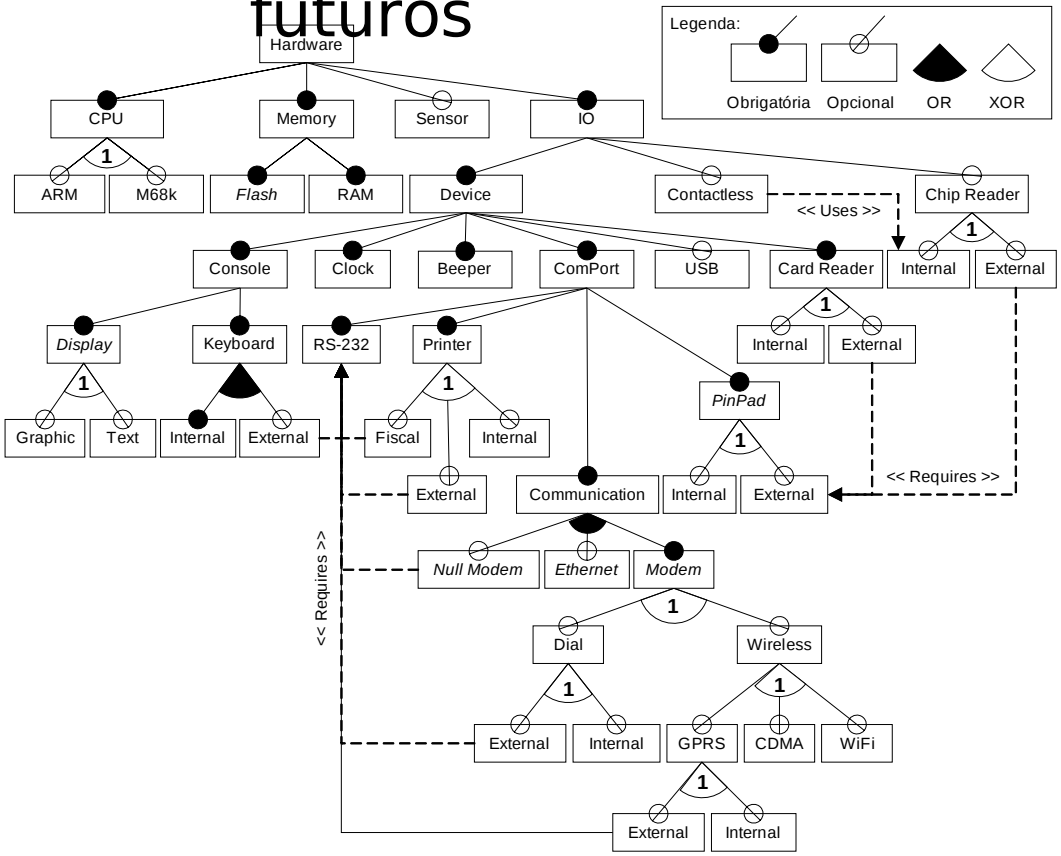


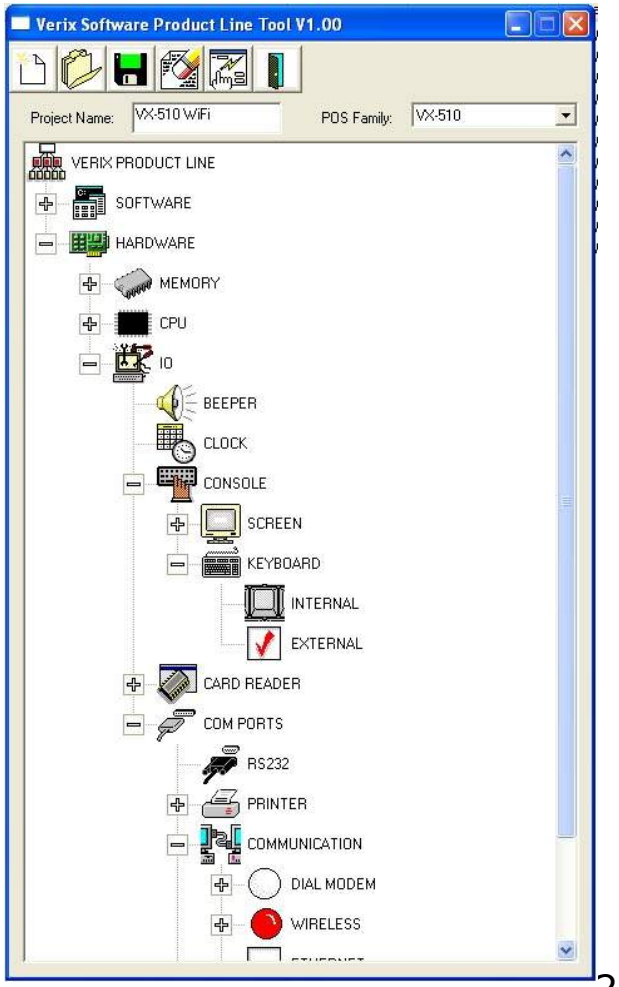
Considerações Finais

- Limitações
  - Único domínio (Terminais POS);
  - Único tipo de código (Procedimental);
  - Variabilidades do Hardware;
  - Sem ferramentas de apoio;
  - Sem gerência de configuração;
  - Sem especificação das atividades de teste.

Considerações Finais

- Trabalhos futuros





Fim

- Perguntas
- Comentários