

DESENVOLVIMENTO DE SISTEMAS EMBEBIDOS

(MESTRADO EM INFORMÁTICA)

- SESSÃO 1: Introdução aos Sistemas Embebidos -

JOÃO MIGUEL FERNANDES

Email: miguel@di.uminho.pt

URL: <http://www.di.uminho.pt/~miguel>



UNIVERSIDADE DO MINHO
ESCOLA DE ENGENHARIA

2000/01



DEP. INFORMÁTICA

Sumário

1. Enquadramento
2. Sistemas Embebidos
3. Taxinomia
4. Motivação

© 2001 UMBEEDJINF

2

1. Enquadramento (1/3)

- **Apresentação pessoal**
 - docente do DI/EE/UM
 - coordenador C&T do IDITE-Minho
 - membro do SIG-ES do IFIP
- **Doutoramento** (Fev/2000)
 - **Título**
 - "MIDAS: Metodologia Orientada ao Objecto para Desenvolvimento de Sistemas Embebidos"
 - **Área científica**
 - Engenharia de Sistemas Baseados em Computador
 - **Área aplicacional**
 - Sistemas de Informação Industriais

© 2001 UMBEEDJINF

3

1. Enquadramento (2/3)

- **Objectivos desta sessão**
 - apresentar a disciplina
 - introduzir definições e terminologia a usar na disciplina
- **Audiência alvo**
 - licenciados (com ou sem formação na área das TSI) com responsabilidades e experiência comprovada (desejável!) na análise, concepção e implementação de sistemas baseados em software

© 2001 UMBEEDJINF

4

1. Enquadramento (3/3)

- **Bibliografia recomendada**
 - Wolf W. (2001). "Computers as Components: Principles of Embedded Computing System Design". Morgan Kaufman Publishers. ISBN 1-55860-541-X.
 - Gajski D., Vahid F., Narayan S., Gong J. (1994). "Specification and Design of Embedded Systems". Prentice-Hall. ISBN 0-13-150731-1.
 - Kumar S., Aylor J.H., Johnson B.W., Wulf, W.A. (1996). "The Codesign of Embedded Systems: A Unified Hardware / Software Representation". Kluwer Academic Publishers. ISBN 0-201-49834-0.
 - Morris D., Evans G., Green P., Theaker C. (1996). "Object-Oriented Computer Systems Engineering". Applied Computing. Springer-Verlag, Londres, Reino Unido. ISBN 3-540-76020-2.
 - Douglass B.P. (1998). "Real-Time UML: Developing Efficient Objects for Embedded Systems". Object Technology. Addison-Wesley. ISBN 0-201-32579-9.

© 2001 UMBEEDJINF

5

2. Sistemas Embebidos (1/13)

- **Definição de sistema computacional**
 - conjunto de elementos que são organizados para atingir um objectivo pré-definido, através do processamento automático de informação.
- Nesta disciplina, será tratada uma classe especial dos sistemas computacionais: os sistemas embebidos.
- As técnicas e os métodos a tratar nesta disciplina são, na sua maioria, genéricos a outros sistema computacionais (não são exclusivos de sistemas embebidos).

© 2001 UMBEEDJINF

6

2. Sistemas Embebidos (2/13)

- Em 1770, o barão van Kempelen apresentou um autómato capaz de jogar xadrez contra jogadores humanos.
- A máquina ficou conhecida como o Turco, pois efectuava os seus movimentos através dum braço mecânico duma marioneta, com bigode e turbante.
- A marioneta estava ligada a uma cabina onde, supostamente, estaria o "cérebro" da máquina.
- Antes e depois dos jogos, a cabina era aberta para mostrar à audiência que a máquina operava sozinha sem qualquer intervenção humana.
- Anos mais tarde, descobriu-se que dentro da máquina estava, de facto, escondido um xadrezista, normalmente de baixa estatura, para mais facilmente se esconder.

© 2001 UMEEED/DJMF

7

2. Sistemas Embebidos (3/13)



© 2001 UMEEED/DJMF



8

2. Sistemas Embebidos (4/13)

- Apesar do Turco ter um "processador" humano, vamos usá-lo para ilustrar as características essenciais que os sistemas embebidos apresentam actualmente.
- O autómato inclui hardware e software.
- O hardware, usado para interactuar com o ambiente, é composto por sensores (detecção das peças do adversário) e actuadores (braço mecânico).
- O software (o intelecto do xadrezista) controla as acções do hardware.
- O sistema responde às jogadas do adversário dentro dum intervalo de tempo aceitável em jogos de xadrez.

© 2001 UMEEED/DJMF

9

2. Sistemas Embebidos (5/13)

- Tipos de elementos usados um sistema computacional
 - Software: Programas e estruturas de dados que realizam a funcionalidade requerida.
 - Hardware: Dispositivos electrónicos que disponibilizam a capacidade computacional e os dispositivos electro-mecânicos que providenciam o contacto com o exterior.
 - Pessoas: Utilizadores e operadores que usam o hw e o sw.
 - Base de dados: Repositório de informação usado via software.
 - Documentação: Manuais, relatórios e outros documentos que descrevem o uso e/ou a operação do sistema.
 - Procedimentos: Passos que definem o uso específico de cada elemento do sistema ou o contexto no qual este reside.

© 2001 UMEEED/DJMF

10

2. Sistemas Embebidos (6/13)

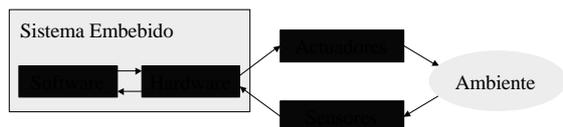
- Sistema embebido é um sistema computacional
 - Um sistema embebido é concebido para uma aplicação específica, contendo componentes de hardware e de software.
 - Um sistema embebido faz parte, como o próprio nome indicia, dum sistema mais complexo e nele se encontra incorporado (embutido, embebido, embarcado).
 - O termo "embebido", foi popularizado pelo DoD e refere-se ao facto de esses sistemas estarem incluídos em sistemas maiores, cuja principal função não é a computação.
- Definição: um sistema embebido é qualquer dispositivo que inclua um computador programável mas que não tenha por finalidade a computação genérica.

© 2001 UMEEED/DJMF

11

2. Sistemas Embebidos (7/13)

- Estrutura típica dum sistema embebido



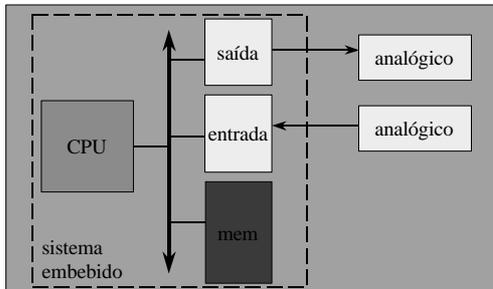
© 2001 UMEEED/DJMF

- Exemplos de sistemas embebidos
 - PDAs, impressoras, telemóveis, televisões, electrodomésticos
 - Sistemas de controlo (indústria, medicina, militar, doméstica)
 - Sistemas de monitorização de processos industriais
 - Sistemas de aquisição de dados (laboratórios)

12

2. Sistemas Embebidos (8/13)

▪ Estrutura típica dum sistema embebido



© 2001 UMFEED/JMIF

13

2. Sistemas Embebidos (9/13)

▪ Propriedades relevantes dum sistema embebido

- É desenvolvido para realizar uma função específica para uma dada aplicação (muitas vezes, será apenas posto a funcionar um único sistema).
- Deve apresentar um funcionamento em contínuo.
- Deve manter uma interacção permanente com o respectivo ambiente. Deve responder continuamente a diferentes eventos provenientes do exterior e cuja ordem e tempo de ocorrência não são previsíveis.
- Tem de estar correctamente especificado (e implementado), pois realiza tarefas críticas em termos de fiabilidade e segurança.
- Deve obedecer a imposições ou restrições temporais, pelo que questões de tempo-real têm que ser equacionadas.
- É digital.

© 2001 UMFEED/JMIF

14

2. Sistemas Embebidos (10/13)

▪ Características dum sistema embebido

- Para além dos requisitos funcionais, também têm de ser considerados os requisitos não-funcionais.
 - Peso, tamanho.
 - Custo.
 - Desempenho (débito ou tempo)
 - Requisitos temporais.
 - Tolerância a faltas.
 - Quantidade de memória.
 - Distribuição dos nós de computação.
 - Estilo da família de produtos.
 - Duração da bateria.
 - ...

© 2001 UMFEED/JMIF

15

2. Sistemas Embebidos (11/13)

▪ Sistema reactivo

- Um sistema reactivo é aquele que mantém uma interacção permanente ou frequente com o seu ambiente e que responde continuamente, em função do seu estado interno, aos estímulos externos a que está sujeito.
- Os sistemas reactivos não podem ser descritos apenas pela especificação dos sinais de saída em função dos sinais de entrada, mas, antes, por especificações que relacionam as entradas e as saídas ao longo do tempo.
- Tipicamente, as descrições de sistemas reactivos incluem sequências de eventos, acções, condições e fluxos de informação, combinadas com restrições temporais, para definir o comportamento global do sistema.

© 2001 UMFEED/JMIF

16

2. Sistemas Embebidos (12/13)

▪ Sistema de monitorização (de controlo)

- Os sistemas de monitorização e os sistemas de controlo são responsáveis pela supervisão dum dado ambiente não "inteligente", informando constantemente o seu utilizador (humano ou não) do estado daquele e actuando em situações críticas.
- Por exemplo, numa instalação industrial, quando um dado gás atinge uma determinada concentração (pré-indicada), o respectivo sistema de controlo deve imediatamente accionar um alarme.

© 2001 UMFEED/JMIF

17

2. Sistemas Embebidos (13/13)

▪ Sistema de tempo-real

- Um sistema de tempo-real é um sistema reactivo, cujo comportamento deve respeitar, além da funcionalidade pretendida, um conjunto de restrições temporais externamente definidas.
- No desenvolvimento dum sistema deste tipo, além de ser necessário satisfazer a sua correcção em termos funcionais, devem identificar-se os requisitos temporais e deve assegurar-se que estes são cumpridos durante o desempenho do sistema.
- As restrições temporais a que os sistemas podem estar sujeitos, dividem-se em três categorias principais, correspondendo cada uma delas a um tipo distinto de sistemas (*hard real-time*; *firm real-time*; e *soft real-time*).

© 2001 UMFEED/JMIF

18

3. Taxinomia (1/15)

▪ Hardware e Software

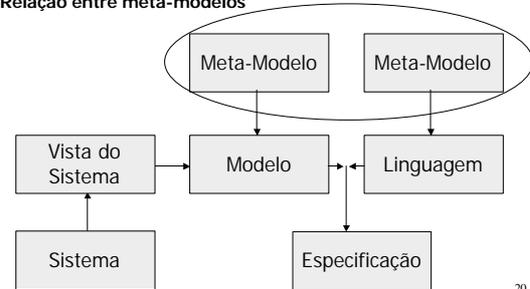
- Os circuitos electrónicos dum sistema computacional, bem como a memória e os dispositivos de entrada/saída formam o hardware do sistema.
- O hardware dum sistema é constituído por objectos tangíveis (circuitos integrados, chips, placas de circuito impresso, cabos, fontes de alimentação, memórias, terminais) e não por ideias abstractas, algoritmos ou instruções.
- O software dum sistema consiste em algoritmos (instruções que indicam como fazer algo) e respectivas representações, nomeadamente os programas.
- Um programa pode ser representado em diversos suportes (cartão perfurado, fita magnética, disquete, disco compacto), mas a sua essência reside no conjunto de instruções que o constitui e não no meio físico no qual é armazenado.

© 2001 UMFEED/UMF

19

3. Taxinomia (2/15)

▪ Relação entre meta-modelos



© 2001 UMFEED/UMF

20

3. Taxinomia (3/15)

▪ Sistema

- Um sistema pode ser definido como uma colecção de componentes inter-relacionados que actuam como um todo, para atingir um determinado objectivo.
- Esta definição permite que praticamente tudo o que existe no universo seja entendido como um sistema.
- O que interessa não é saber se algo é ou não um sistema, mas antes se esse "algo" pode ser visto como um sistema
- Quando tal se verifica, há um interesse em estudar as propriedades desse sistema como um todo.
- É o observador do sistema que define a fronteira deste com o seu ambiente, o que torna a definição de sistema não intrínseca a este, mas dependente do seu observador
- Como consequência, os componentes que, num determinado contexto, constituem um dado sistema, podem, noutro, ser apenas um sub-sistema dum sistema mais amplo.

© 2001 UMFEED/UMF

21

3. Taxinomia (4/15)

▪ Vista dum sistema

- Uma vista dum sistema é uma perspectiva (total ou parcial) do sistema que se pretende representar.
- Por exemplo, a descrição dum livro pode ser feita segundo duas vistas distintas.
- Uma hipótese é descrever o livro como sendo constituído por várias páginas; cada página é descrita como uma série de linhas, podendo eventualmente incluir figuras; cada linha, por sua vez, é composta por símbolos (letras, dígitos, pontuação).
- Outra vista, mais lógica, baseia-se na estrutura do conteúdo do livro. Um livro está dividido em capítulos; cada um dos capítulos tem um ou mais sub-capítulos; um sub-capítulo pode conter várias secções; finalmente, chega-se aos parágrafos, em que cada um é composto por diversas frases, que, por sua vez, se compõem de palavras. As palavras, por seu lado, são constituídas por letras.

© 2001 UMFEED/UMF

22

3. Taxinomia (5/15)

▪ Meta-modelo

- Um meta-modelo (modelo dum modelo) é um conjunto de elementos, funcionais ou estruturais, de composição e de regras de composição que permitem construir uma representação conceptual para o sistema.
- Exemplos: fluxogramas, FSMs, redes de Petri, grafos de fluxo de controlo e dados (CDFG).
- O meta-modelo deve ser formal (preciso, rigoroso), evitando ambiguidades na interpretação da representação do sistema, e deve também ser completo, permitindo a construção duma representação que descreva totalmente o sistema.
- Um meta-modelo formal designar-se por formalismo.

© 2001 UMFEED/UMF

23

3. Taxinomia (6/15)

▪ Modelo

- Um modelo é uma representação conceptual dum sistema, à luz dum determinado meta-modelo.
- Exemplos: FSM da unidade de controlo dum elevador ou o grafo de fluxo de dados e controlo desse elevador.
- O modelo manifesta todas as características do respectivo meta-modelo.
- Parte-se do pressuposto que o projectista explora e usa todas as características do meta-modelo na construção do modelo.
- Um modelo diz-se formal quando evita ambiguidades na sua interpretação e completo quando representa totalmente o sistema.

© 2001 UMFEED/UMF

24

3. Taxinomia (7/15)

▪ Linguagem

- Uma linguagem (ou notação) é o conjunto de todas as frases válidas que é possível construir utilizando a respectiva gramática.
- Exemplos: Pascal, C, Java, VHDL, assembly i8086.
- Também para linguagens se pode considerar a existência do respectivo meta-modelo.
- O meta-modelo duma linguagem é o modelo conceptual, seguido na definição da linguagem.
- Qualquer especificação escrita numa linguagem segue obrigatoriamente o meta-modelo por ela imposto.
- A linguagem manifesta todas as características do seu meta-modelo.

© 2001 UMFEED/JMFP

25

3. Taxinomia (8/15)

▪ Especificação

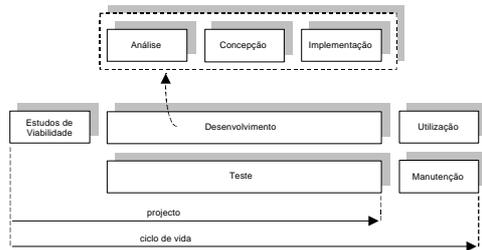
- Uma especificação é uma representação concreta (real) do modelo do sistema numa dada linguagem.
- Exemplos: FSMs da unidade de controlo do elevador escritas em VHDL ou em C.
- As características da especificação dependem simultaneamente das características da linguagem (i.e. meta-modelo da linguagem) e do modelo do sistema (i.e. meta-modelo do sistema).
- Em termos práticos, não há grandes diferenças, ao nível conceptual, entre uma especificação e um modelo dum dado sistema, pois ambos são representações deste.

© 2001 UMFEED/JMFP

26

3. Taxinomia (9/15)

▪ As fases do ciclo de vida dum sistema



© 2001 UMFEED/JMFP

27

3. Taxinomia (10/15)

▪ Ciclo de vida

- Em sistemas de software, usa-se o termo ciclo de vida (*life cycle*) para referir o conjunto de actividades que se inicia no momento em que o sistema é mentalmente conceptualizado até ao instante em que ele é retirado definitivamente de uso.
- O ciclo de vida representa todo o conjunto de actos válidos, realizados, durante a vida útil do sistema, com o objectivo de o idealizar, desenvolver e usar.

▪ Desenvolvimento

- O desenvolvimento inclui as fases do ciclo de vida responsáveis pela construção do sistema, (a análise, a concepção e a implementação).
- Excluem-se, por exemplo, os estudos de viabilidade económica, as tarefas de manutenção e a utilização efectiva do sistema.

© 2001 UMFEED/JMFP

28

3. Taxinomia (11/15)

▪ Projecto

- O termo projecto é usado para referir as actividades anteriores à utilização prática do respectivo sistema.
- Projectista (equipa de projecto) é usado para indicar os indivíduos responsáveis por qualquer das fases do projecto dum dado sistema (ou por uma parte deste).
- Podem usar-se termos como analista ou programador para referir um projectista especialista numa dada fase do projecto.
- Os termos utilizador e cliente são usados para indicar, respectivamente, o indivíduo que interage directamente com o sistema final e a pessoa que encomenda o desenvolvimento do sistema à equipa de projecto.

© 2001 UMFEED/JMFP

29

3. Taxinomia (12/15)

▪ Processo

- Chama-se processo à sequência de factos que conduzem a certo resultado (exemplos: processo de fabrico, processo químico, processo judicial), mais particularmente, à sequência de actos ordenados para a consecução de certa finalidade.
- Por exemplo, o processo industrial, numa dada fábrica, representa o conjunto de actos executados para produzir um determinado bem de consumo, desde a chegada das matérias primas até à embalagem para carregamento.
- No âmbito da construção metódica de sistemas, processo significa o conjunto de tarefas executadas ao longo do ciclo de vida do sistema.
- O termo processo de desenvolvimento tem um âmbito menos abrangente e restringe-se às tarefas realizadas durante o desenvolvimento do sistema.

© 2001 UMFEED/JMFP

30

3. Taxinomia (13/15)

- **Modelo do processo**
 - Um modelo de processo é um esquema que organiza, ordena e relaciona a forma como as várias fases e tarefas devem ser prosseguidas ao longo do ciclo de vida do sistema.
 - A função principal dum modelo de processo é determinar a ordem das fases envolvidas no desenvolvimento de sistemas e estabelecer os critérios de transição para progredir entre fases.
 - Quais as diferenças entre um processo e o respectivo modelo?
 - Um modelo de processo pode, por exemplo, considerar a existência de duas tarefas que, potencialmente, poderão ser executadas em paralelo.
 - No entanto, durante a real execução dum processo, que segue esse modelo, essas tarefas são executadas em sequência, enquanto que noutro processo são executadas em paralelo.

© 2001 UM/FEED/DJMF

31

3. Taxinomia (14/15)

- **Fases do ciclo de vida**
 - Uma fase é uma abstracção, ao longo do tempo, dum conjunto de actividades, ou seja, é um conceito útil para agregar actividades e relacioná-las temporalmente.
 - O ciclo de vida dum sistema complexo é dividido nas seguintes fases genéricas:
 - O propósito do estudo de viabilidade consiste na avaliação dos custos e benefícios do sistema proposto.
 - Na análise são levantados os requisitos do sistema e produz-se um modelo abstracto que descreve os aspectos fundamentais do domínio de aplicação e que permite captar a essência do sistema em causa.
 - Na concepção, com base no modelo obtido na análise, é criado um modelo que especifica os componentes que realizam uma determinada solução para o sistema.

© 2001 UM/FEED/DJMF

32

3. Taxinomia (15/15)

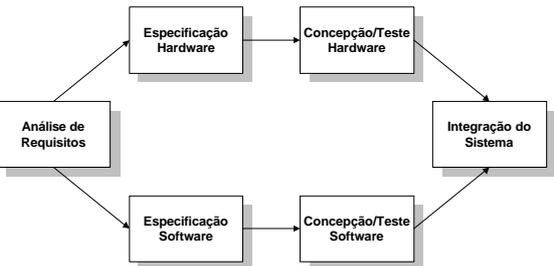
- **Fases do ciclo de vida (cont.)**
 - O ciclo de vida dum sistema complexo é dividido nas seguintes fases genéricas
 - Na implementação, realiza-se uma solução, descrita numa dada linguagem de programação, transformando o modelo de concepção.
 - Durante o desenvolvimento do sistema, o teste decorre em paralelo, a fim de se tentar encontrar todas as falhas que o sistema possa conter.
 - A utilização ocorre quando o sistema é posto a funcionar em ambiente real.
 - A manutenção procura corrigir todas as anomalias que não foi possível detectar durante o desenvolvimento e pretende também fazer evoluir o sistema de modo a que continue a ser útil aos seus utilizadores.

© 2001 UM/FEED/DJMF

33

4. Motivação (1/5)

- **O modelo de processo tradicional**



© 2001 UM/FEED/DJMF

34

4. Motivação (2/5)

- **Algumas crenças**
 - A interligação é fácil.
 - O software é maleável (i.e. fácil de alterar), pelo que deficiências encontradas no hardware podem rectificar-se no software.
 - O desenvolvimento das 2 componentes faz-se em paralelo.
- **Na prática**
 - Abordagem baseada no software.
 - Abordagem baseada no hardware (a mais usual)

© 2001 UM/FEED/DJMF

35

4. Motivação (3/5)

- **Típica abordagem baseada no hardware**
 - Escolha dum microprocessador (ou microcontrolador) comercial. Esta opção é ditada pelo conhecimento específico dos projectistas no referido processador.
 - Desenvolvimento duma versão *bread boarded* do hardware a projectar, sob o qual o software irá ser, posteriormente, desenvolvido.
 - Projecto do hardware e produção de software (decorrem paralela e independentemente). Surge uma primeira versão do hardware final.
 - Integração do software e do hardware desenvolvidos. Esta fase produz muitos problemas e nela são detectadas inúmeras falhas, de que resultam alterações profundas, no hardware e no software.
 - Finalmente, após várias iterações e reformulações, é possível construir um protótipo do sistema em desenvolvimento.

© 2001 UM/FEED/DJMF

36

4. Motivação (4/5)

- Problemas da abordagem baseada no hardware
 - O tempo de desenvolvimento é muito prolongado.
 - O produto final não responde aos requisitos do utilizador. (análise do problema muito superficial).
 - A documentação dos diversos elementos do projecto é precária.
 - A codificação é pouco metódica (código esparguete).
 - A integração do software no hardware é muito “dolorosa”.
 - A detecção de erros não é facilitada.
 - É impossível alterar o hardware (passar funcionalidades do software para o hardware, depois da placa estar feita).
 - A abordagem não produz um sistema optimizado (quaisquer que sejam as métricas usadas).
 - A manutenção é muito difícil.

4. Motivação (5/5)

- O que é preciso para desenvolver sistemas embebidos
 - Modelos de especificação multi-vista (UML).
 - Processo de software controlável e repetível.
 - Métodos de projecto.
 - Abordagem orientada aos objectos.
 - Mecanismos para atacar a complexidade dos sistemas.
- Nesta disciplina, apresentam-se as necessidades metodológicas para desenvolver sistemas embebidos de grande complexidade, pondo especial ênfase nas questões associadas à fase de análise (não esquecendo a concepção e a implementação).