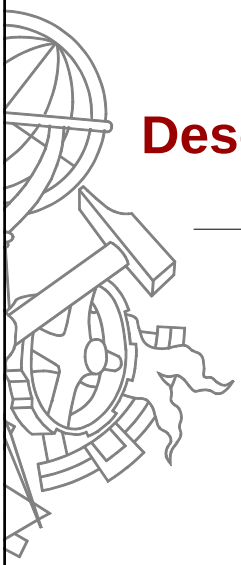


Ambientes de Desenvolvimento Avançados

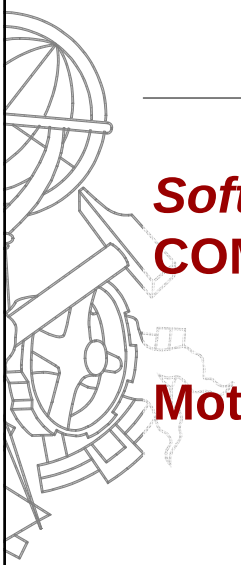
http://www.dei.isep.ipp.pt/~jtavares/ADAV/ADAV.htm

Aula 3

Engenharia Informática

2005/2006

José António Tavares
jrt@isep.ipp.pt



Software baseado em COMPONENTES

Motivação

O que é um componente?

- **"A software package which offers *service* through *interfaces*"**
  [Peter Herzum and Oliver Sims, "Business Components Factory: A Comprehensive Overview of Component-Based Development for the Enterprise", John Wiley & Sons, Incorporated, 1999].
- **"A coherent package of software artifacts that can be independently developed and delivered as a unit and that can be composed, unchanged, with other components to build something larger"**
  [D.F. D'Souza and A.C. Wills, "Objects, Components, And Frameworks with UML – The Catalysis Approach" Addison-Wesley, 1998].
- **"A component is a unit of composition with contractually specified interfaces and *explicit context dependencies* only. A software component can be deployed independently and is subject to composition by third parties."**
  [C. Szyperski, "Component Software: Beyond Object-Oriented Programming" Addison-Wesley, 1998].

O que não é um componente?

Component isn't an object, not in sense of simply being an object in a Java or C++ program, although it is true at runtime.

Motivação - Agenda

- Componentes são para composição
- Componentes – *software* à medida versus *standard*
- Inevitabilidade dos componentes
- Natureza do *software* e entidades '*deployable*'
- Componentes são unidades de '*deployment*'
- O que o passado nos ensina

Do capítulo **1** do livro de C. Szyperski, "Component Software: Beyond Object-Oriented Programming".

Motivação para os componentes

- O desenvolvimento da WWW e *Internet*
  - Sistemas de serviços poucos coordenados
- Técnicas de projecto e linguagens Orientadas aos Objectos
- Movimento da computação baseada em *Mainframes* para computação do tipo Cliente-Servidor
- Ritmo rápido das mudanças tecnológicas
- Necessidades económicas de maximizar a reutilização

Componentes são para composição

- Componentes de *software* permitem a reutilização de partes. Composição permite que partes "pré-fabricadas" possam ser reutilizadas pela sua composição em novas partes compostas;
- A composição de componentes melhora a qualidade, permite uma maior rapidez de desenvolvimento e aumenta a flexibilidade;
- Espera-se que o *software* baseado em componentes seja o 'paradigma' do desenvolvimento de *software* nos próximos anos
- . . . mas é isto o que se tem dito desde 1968
- . . . no entanto, os recentes desenvolvimentos trazem uma renovada esperança.

Re-utilização de *Software*

- *Reutilizar* é um termo muito abrangente que cobre o conceito geral de um recurso reutilizável
- A reutilização pode acontecer em vários níveis de abstracção dos sistemas:
  - Padrão (*System Pattern*)
  - Especificação
  - Projecto
  - Codificação

Software baseado em componentes é um conceito novo?

- Subrotinas
  - Turing, 1949, *Checking a Large Routine*
- Programação estruturada
  - Dijkstra, 1968
- Bibliotecas (*Libraries*)
  - NAG, 1971
- Information Hiding
  - Parnas, 1972

Então o que é novo?

- Modularisation of software is *not* new;
- What we want of a component is that
  - It may be used by other program elements (*clients*)
    (encapsulation and low coupling – good strategies for any modular design)
  - The clients and their authors do not need to be known to the component's authors
    This is a little bit new, and only works if all the respective authors work to a common standard

Engenharia de componentes

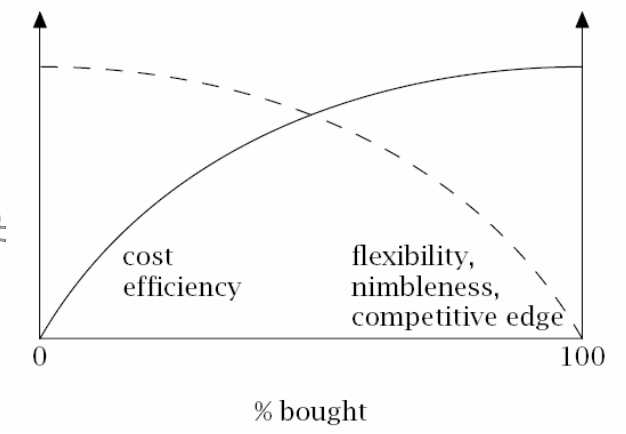
- use of components is axiomatic in any mature engineering discipline;
- by analogy, should be in software engineering too;
- 1968: software crisis ⇒ software integrated circuits;
- sometimes said that software is too *flexible* for components;
- . . . but this just indicates immaturity in the discipline;
- claim: *software engineering now entering maturity*.

Componentes – *software* à medida versus *standard*

- one extreme: project from scratch, using only programming tools and libraries;
- optimally adapted, but expensive, technically suboptimal, usually late, and requires maintenance;
- hence major trend towards other extreme: project outsourced, COTS (*Component of the Shelf*) software bought and parameterized until close enough;
- may involve business reorganization, cannot confer any competitive advantage, slow to adapt to changing needs;
- component software is a 'third way': customized assembly of standard components; compromise between cost efficiency and flexibility.

Componentes – *software* à medida versus *standard*

Espectro entre o fazer tudo e o comprar tudo



Inevitabilidade dos componentes

- To speed up development process;
- To reduce development cost;
- However, compared with make-all systems, component-based systems may have some lost in flexibility and competitive edge if they are not well organised with most suitable components.

Inevitabilidade dos componentes

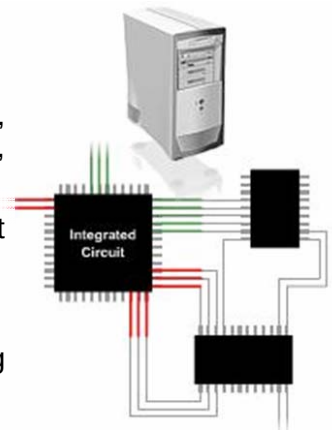
- technical superiority of component technology is not enough;
- *critical mass* also required: variety and quality, apparent benefit, strong or numerous sources;
- then use of components becomes inevitable, and proprietary solutions become unsustainable;
- if such a vortex is plausible, *preparedness* for an emerging component market is essential;
- preparation involves component-friendly (modular) approaches, which has its own advantages anyway.

Natureza do *software*

- software components initially considered analogous to hardware components

  software IC, software bus

- also analogy with hi-fi components, mechanical engineering (gears, nuts, bolts), Lego;
- but software is different: deliverable is not product, but *blueprint* for product
- software is a meta-product;
- computer is *factory*, instantiating blueprint;
- *by default, repeated instantiation is easy.*



Entidades '*Deployable*'

Definição

Entidades '***deployable***' :

São entidade de instalação e distribuição binária.

Entidades '*Deployable*'

- as important to distinguish between software and instances as between plan and building;
- confusion between classes and objects;
- understandable: in software, both plan and building representable by same kind of thing (eg object, bits);
- plans can be parameterized, applied recursively, scaled, repeatedly instantiated, whereas instances cannot;
- software is a *generic metaproduct*;
- maths a good model of logical structure of software, but not of engineering and market aspects
- moral: software engineering is a blend of physical and logical engineering

Unidades de '*Deployment*'

- software component is what is actually deployed in a component-based approach;
- objects are almost never bought, sold or deployed independently;
- unit of deployment is typically more static: class, or framework of classes;
- objects forming part of 'component instance' are instantiated from classes deployed within the component;
- thus components themselves are not normally instantiated; indeed, component may not have OO implementation at all;
- object technology is mostly used to construct monolithic applications, not components.

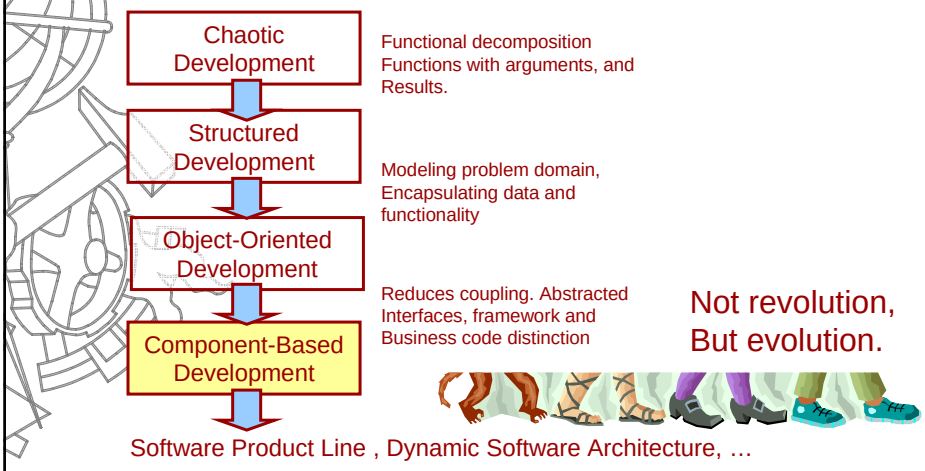
O que o passado nos ensina? (1/2)

- For many years, the most popular has been Microsoft's Visual Basic;
- Later the JAVA (Enterprise JavaBeans – EJB) and COM+;
- Oldest component success stories:
  - modern OSs (applications composed in environment, communicating via filestore or pipes)
  - plug-in architectures (eg Netscape Navigator, PhotoShop)
  - MacOS extensions, DOS TSRs

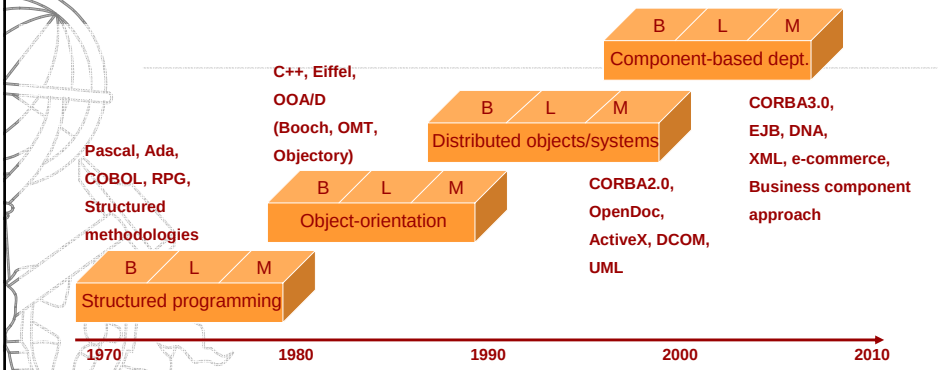
O que o passado nos ensina? (2/2)

- infrastructure providing *rich foundational functionality*
- components purchased from *independent providers*, *deployed by clients*, with *direct meaning to client*
- different roles for component *construction* and *assembly*
- large enough for reimplementation not to be cost effective
- multiple components from different sources coexist
- . . . but arbitrary combinations may conflict (composability need only be highly likely, not guaranteed)

História do desenvolvimento do S/W



Evolução para os Componentes na Industria



- **B**, Bleeding edge is a phase in which a few visionary organizations start to work out certain concepts and apply them, having to create new technologies from scratch or use existing technologies in a very unusual.
- **L**, Leading edge is a phase in which there is industry support for a new technology, but the right way to use this technology is still known to only a few organization.
- **M**, Mature is the phase in which the knowledge of a new approach is largely available and the supporting technologies are stable.

Questões

?