A Strategic Comparison of Component Standards

Prof. Dr. Wolfgang Pree
Department of Computer Science
cs.uni-salzburg.at

© Copyright Wolfgang Pree, All Rights Reserved
Contents

- What is a component?
- COM :: Java :: Corba
- Visions
What is a component?
Remember: What is missing in OO?

- visual/interactive configuration
- interoperability
What is a component?

- Not yet clearly defined

- Is everything a component?
  - macros, mixins, functions, procedures, modules, classes, etc.

- Conventional, heavy-weight components:
  - operating systems
  - database systems
Our definition of the term (software) component

A piece of software with a programming interface
Wiring standards (I)

Interoperability problem:

=> wiring standards
Wiring standards (II)

Product-driven definition

Microsoft’s **Component Object Model (COM)**
- evolutionary / incrementally
- originally targeted at the desktop
  => had to be extended for Internet/Intranet and *Enterprise Computing*
- carries some legacy
- de facto standardization through the market dominance of Microsoft
Wiring standards (III)

Consortium standardization (OMG)

CORBA
- slow progress (compared to COM and SunSoft’s JavaBeans)

JavaBeans
- based on 100% pure Java
- standards for integrating other components are under development (EJB, AECORBA)
CORBA model of distributed applications

**Diagram Description:**
- **main-Prg**: Central program
- **Proxy A**: Interface between main-Prg and Proxy B
- **Proxy B**: Interface between main-Prg and IR
- **IR**: Interface request
- **Impl. Rep.**: Implementation representation
- **Dyn. Inv. Int.**: Dynamic invocation interface
- **MOfA(p1)**: Method of A (1)
- **MOfB(p2)**: Method of B (2)

**Layers and Technologies:**
- **(D)SOM, Distributed COM**
- **RMI**
- **ORB Core, IBM DSOM ObjMgr**
- **OS**

**Client/Server:**
- **Client**: Interface to OS
- **Netz**: Network layer
- **Server**: Interface to OS
Characteristics of components

- **Information Hiding**
  - interface described in IDL
  - implementation in any language (Java, ST, C++, C, ...)

- components as binary units (machine-independent byte code is also OK)

- components can be made persistent
Component = Class?

Usually, a component (large-grained component) comprises a couple of classes (fine-grained components):
Beyond Wiring

- meta-level informationen
  - components can ask others about offered features
  - dynamic loading and linking
- semantic aspects

CORBA: wiring

JavaBeans: meta-level \((reflection)\), semantics;
  for pure Java wiring becomes irrelevant

COM: all three aspects
Characteristics of component standards
Component Object Model (I)

COM concepts:
- interfaces and components (= COM classes) have a unique (128-Bit) ID
- each COM-Objekt can be asked, which features are supported:

  interface IUnknown; method QueryInterface
A component can have any number of interfaces:

Extension by adding interfaces; existing interfaces remain untouched.
JavaBeans

- **Properties** (→ Setter/Getter methods) are defined interactively in a Beans environment:

- **Events** form the communication mechanism:
Commonalities and differences
Commonalities

- OO (*Information Hiding, late Binding, Subtyping*)
- *Compound Documents* (original meaning of OLE, idea of OpenDoc)
- component transfer mechanism
  - e.g. JAR files, COM Structured Storage
- coupling based on events
- meta-information
- persistence
Differences

- memory management
- binary standards
- development environments
- versioning
- application domains
- supported platforms and languages
Memory management

- COM: tedious reference counting; should be automated in COM+

- Java: garbage collection; distributed GC not compatible to Java-CORBA integration

- CORBA: no general solution
Binary standards

- core aspect of COM

- in Java: byte code; partially through Java Native Interface (JNI)

- CORBA provides no binary standard (compatibility based on language bindings)
Development environments

- COM: solid environments
- Java/JavaBeans: have to grow up
- CORBA: quite unsatisfying
Versioning

- COM: solved via freezing of interfaces
- Java: based on binary compatibility; tedious rules
- CORBA: not directly supported; unsatisfying version numbers
Applications

- COM: focus on the desktop
- Java: focus on the Web
- CORBA: focus on server/Enterprise Computing

- DCOM and EJB aim at server/Enterprise Computing
- ActiveX-components for Windows-Web-Clients
COM: Due to the binary standard, almost any language can be supported efficiently on any platform (DCOM):

Visual Basic, C, C++, C#, Java, Smalltalk, Object Pascal, Lightning Oberon, Object Cobol, ML, etc.

Java: binary standard based on Java byte code + platform independent (VM per platform)
– too much biased towards Java not well suited for Ada95, REXX, Oberon; impossible for C++
Languages and platforms (II)

- CORBA: ORB developers have to provide language bindings for particular languages

  Thus, only a few languages are supported: C++, (Smalltalk), Java
Visions
Filling the gap

Mega components (SAP, DB systems, operating systems)

only a few medium-sized components exist so far

very small components (GUI components, etc.)
Mechanistic view

Currently software components assembly requires exact matching of interfaces:
Adaptive architectures

**Alternative:** components configure themselves automatically through testing & fitting.

Sources of inspiration:
- Sun’s Jini, Microsoft’s .NET
- agent technology
- ontologies