Components Infrastructures

Manuel Oriol
December 14th, 2005
Component?

“A component is a reusable unit of deployment and composition which is accessed through an interface.”

http://www.idt.mdh.se/cbse-book/overviews/1Part.htm

“In object-oriented programming and distributed object technology, a component is a reusable program building block that can be combined with other components in the same or other computers in a distributed network to form an application.”

http://searchvb.techtarget.com/sDefinition/0,,sid8_gci211826,00.html
Some Components Infrastructures

- Classes
- COM, DCOM
- Corba
- Java Beans, EJB
- Agents (not in the sense of Eiffel agents...)
- A bunch of custom infrastructures (e.g. LuckyJ)
Classes

What we do here since the beginning of the semester.

Are components because they answer to a clear interface, is reusable and is used as a building block.

Argued by others: “they are not self contained”
Component Object Model, DCOM

- Based on RPC
- Spec in binary
- Interface defined using the Microsoft IDL
- By querying the interface: pointer to the interface
- Containment: accepts calls and forward them to another object
- Aggregation: expose the interface of an object as its own.

- DCOM: Distributed COM
Common Object Request Broker Architecture (CORBA)

- Encapsulate some binary components and provide the stubs
- Based on RPC
- Stubs defined by the OMG’s IDL
- Method invocations are available and implemented on skeletons

- Naturally distributed
JavaBeans

- Visual components
- Define properties
- Event-Based Communication
- Can use a Component Assembler

Enterprise Java Beans:
- Non-visual
- Session Beans
- Entity Beans
- No plumbing issues (security, atomicity...)

Software Engineering
Agents

- Intelligent?
- Resident
- Autonomous
- Analyze and report
- On an agent platform

- Mobile
Custom Component infrastructures (e.g. LuckyJ)

Sample Presentation

http://se.inf.ethz.ch/people/moriol/www/Research/ETAPS2003_USE
Challenge of Modeling the Components

- Components are generally less numerous than classes so it is understandable

- Explain interactions between components

- Possibly, code the interactions between components

- Prove things?
UML 2

http://www.agilemodeling.com/artifacts/componentDiagram.htm
Petri Nets

http://www.jfsowa.com/figs/clock.gif

http://www.fmi.uni-stuttgart.de/szs/research/resources/pom/net.gif
Ad Hoc solution: Reuse Frequency (Bay, 2005)

Graph is static
Links are fixed
Ad Hoc solution: LuckyJ (Oriol, 2005)
Problems with Graphical Representations

http://www.fmi.uni-stuttgart.de/szs/tools/mckit/dekker1_ll_net.gif
Conclusions

- A lot of components infrastructures
- A lot of ways of modeling components

How does this help us to build trust in the component-based application?