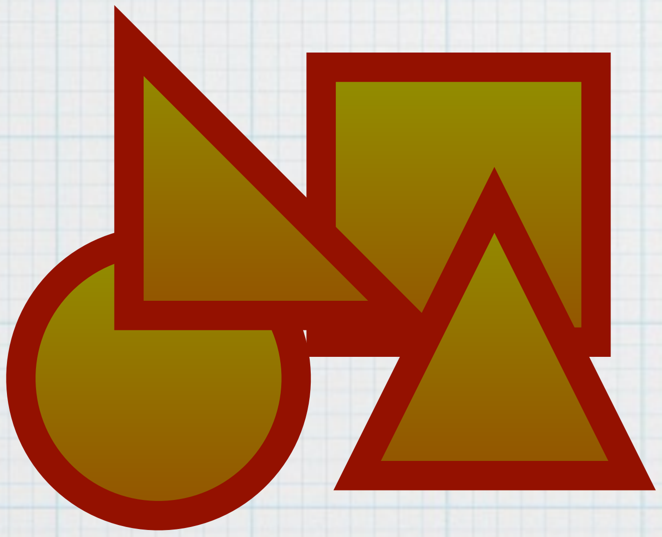
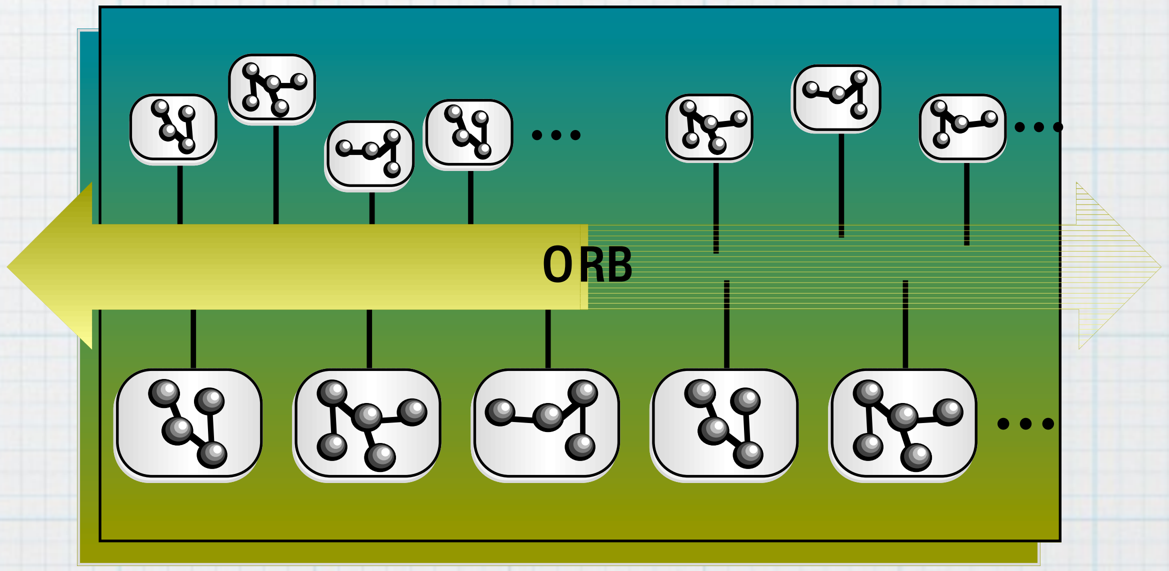
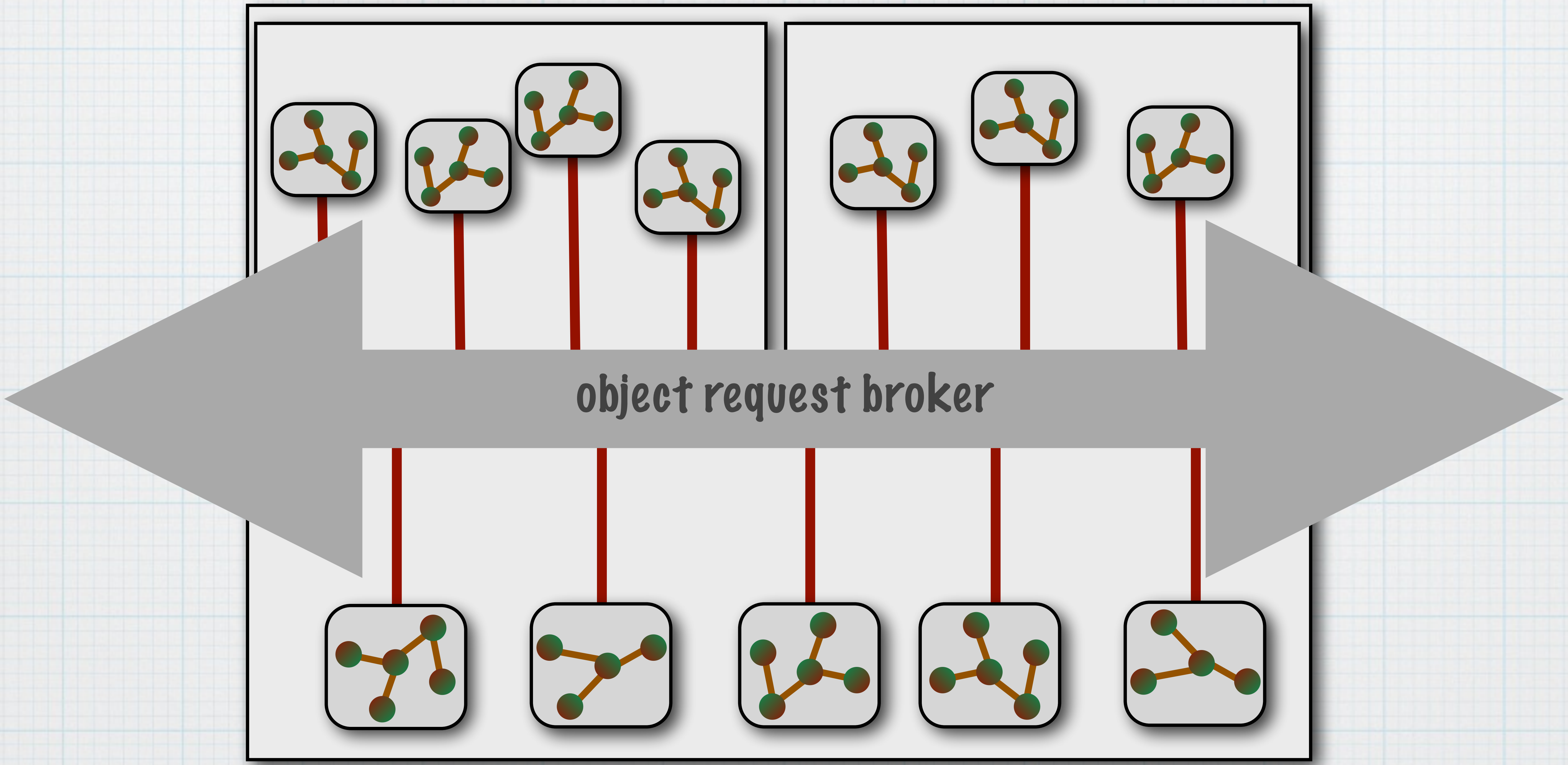
CS630
Component Architecture

Les Waguespack, Ph.D.





Component A, B, C's



Component

- "a software component is a physical packaging of executable software with a well-defined and published interface." Hopkins - 2000
  - software
    physical package » executable
    well-defined
    interface

Component

- "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'Souza - 1999
  - coherent
    software artifacts
    independently developed » independently delivered
    composable unchanged » unit of construction

Component

- "a software 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 a third party." Szyperski - 1998
  - contractually specified interfaces » explicit context
    deployed
    composition by third party

Engineering Drivers

- Reuse
  - "the ability to reuse existing components to create a more complex system."
- Evolution
  - "by creating a system that is highly componentized, the system is easier to maintain. ... changes will be localized ... with little of no effect on the remaining components."

Component "World"

* available components to reuse
  * in-house or third party supply
* a component model supporting assembly and interaction
  * a standard "backplane" for component communication
* • a process and architectures to support component based development
  * component development tools, frameworks, and environments

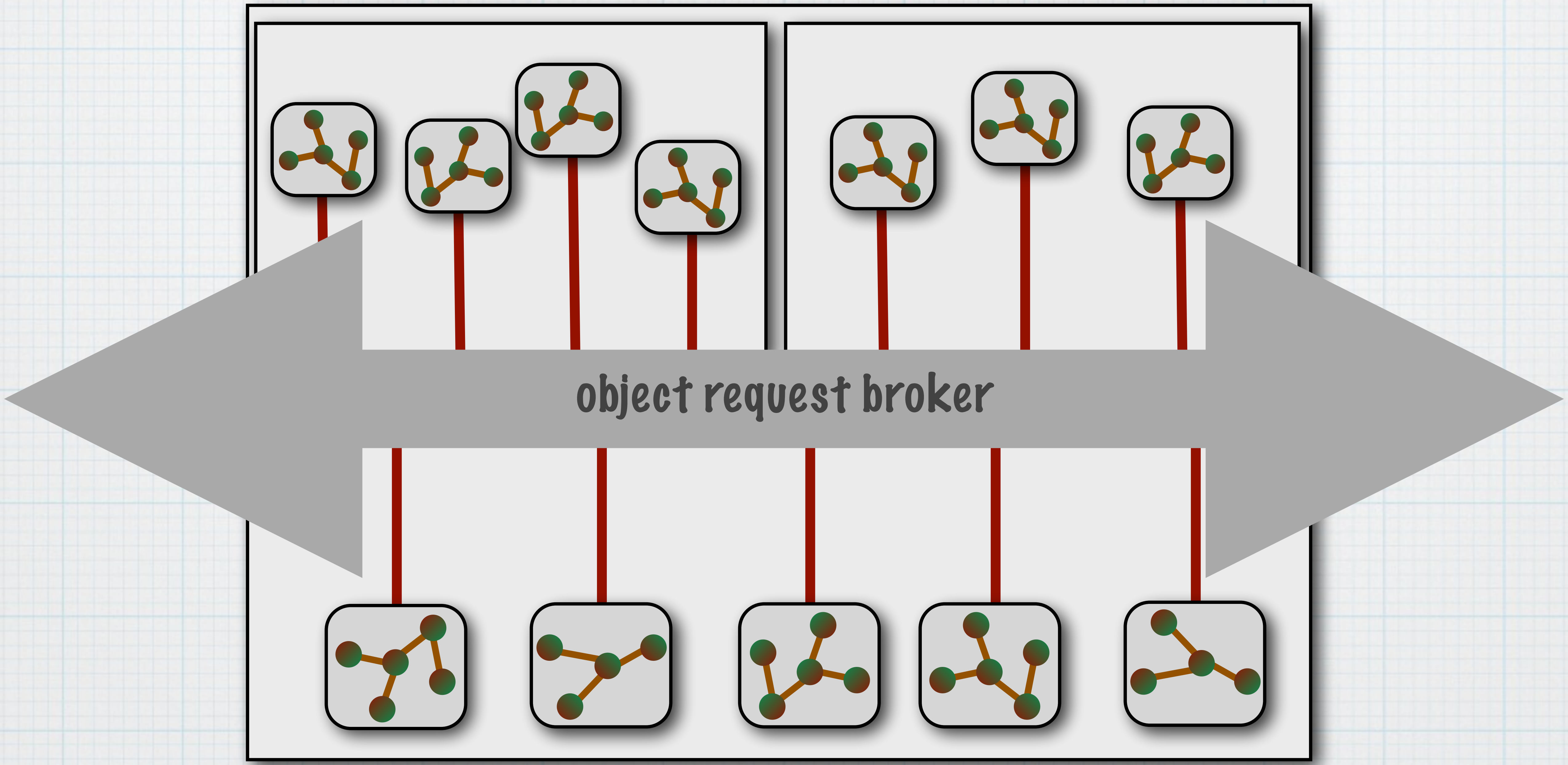
Component = functionary

* Components are components because of how they interact rather than because of how they are constructed
    * "a natural extension of the object model"
    * may not be created using OO tools or languages
    * interact through carefully defined interfaces and "messages"

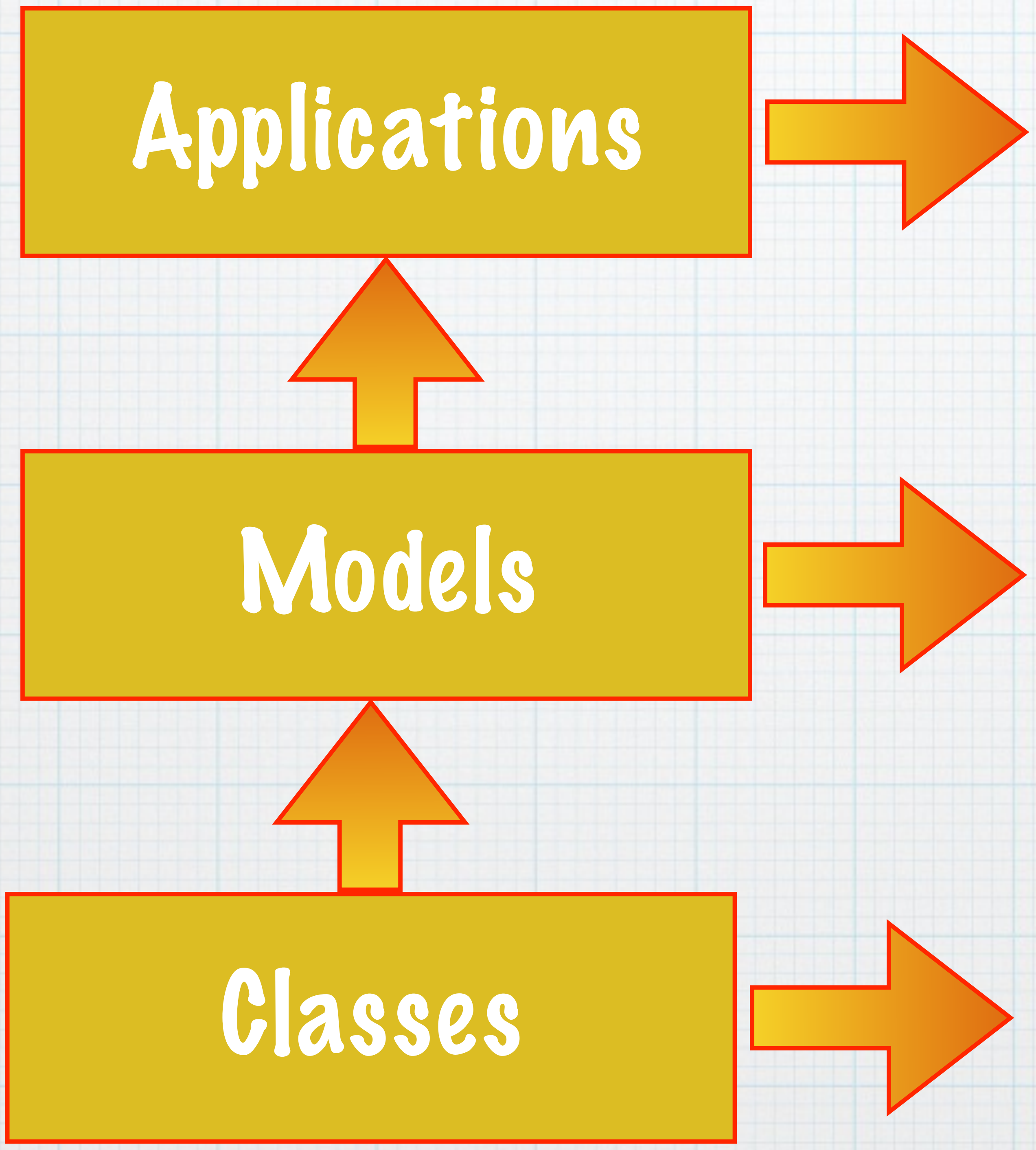
Component Interaction

- • they must find each other
  - the component model must support a "reference model" with "registration"
- • they must converse with messages
  - the component model allows components of different implementation technologies to publish their interfaces, send messages and pass data.
- • component model standards
  - DCOM - distributed component object model (Microsoft)
  - CORBA - common object request broker architecture (OMG)
  - EJB - enterprise Java beans (Oracle / Sun Microsystems)

There must be a "backplane"



Components and Architecture



End Users

Business Units

I. S.

Rapid prototyping provides fast, effective applications

Operational models meet specialized needs

Standardized data and procedures protect corporate assets

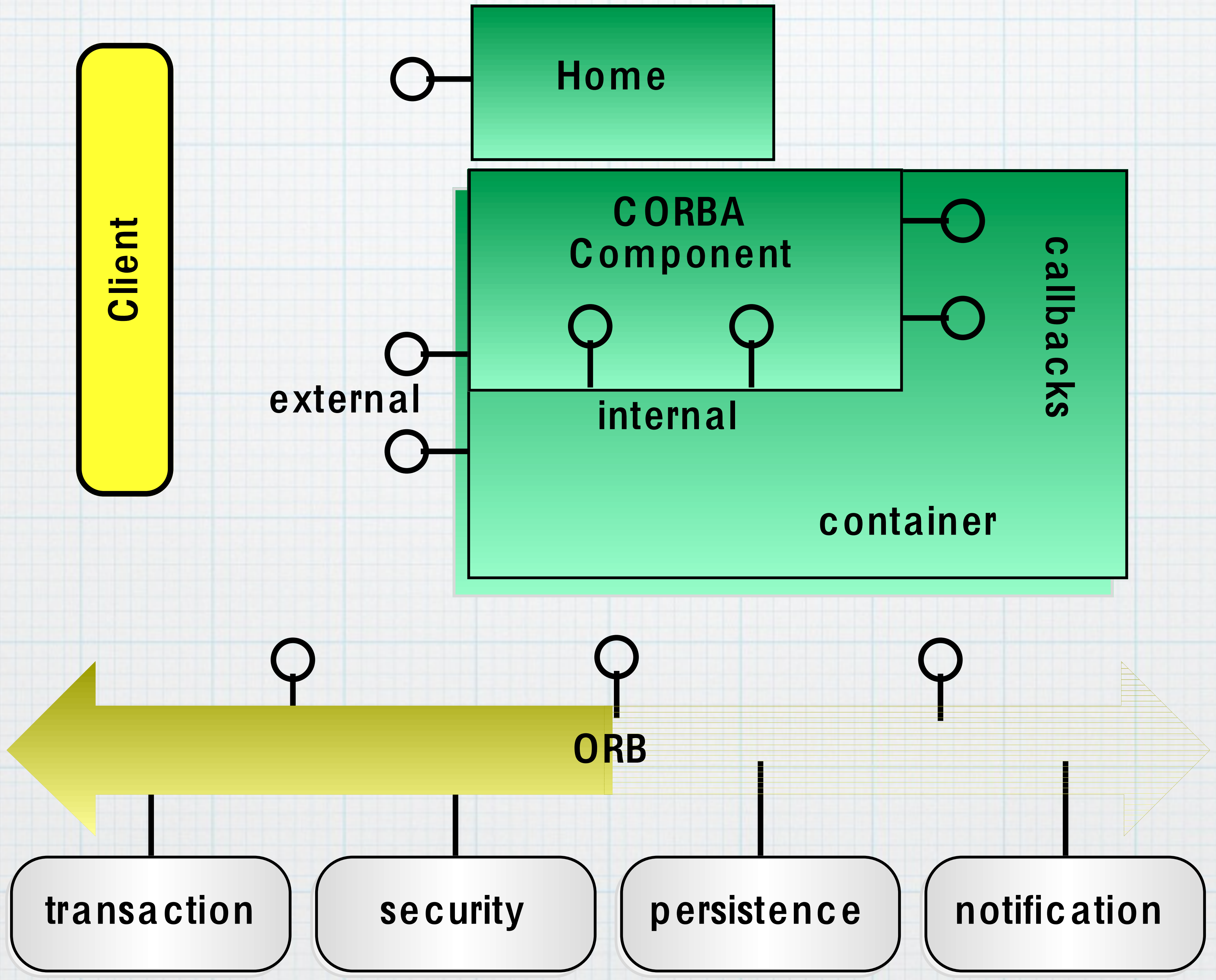
Component Modeling

* UML Component metamodel
    * component view
      object packages building components » interface declaration
* Traceability
    * constituent object models
      extension points
      public and private interface definition
* XML standards for net-based systems
    * Extensible Markup Language
    * XML is a potential "Rosetta stone" for component interfaces
    * any component supporting an XML interface can interact with any other

What Component?!

- Components vs. Applications
  - building block vs. complete solution
  - re-target-able vs. tailor made
  - problem architecture derived vs. solution policy derived
  - "naturally occurring interface" vs. finely focused algorithmic definition
  - Which pieces should be included in a LegoTM or TinkertoyTM set???
- Core problem domain functionality
  - what distinguishes the domain?
    what unique expertise exists in it?
    what "service" in the domain can evolve with the same interface?

CORBA™ Component Model



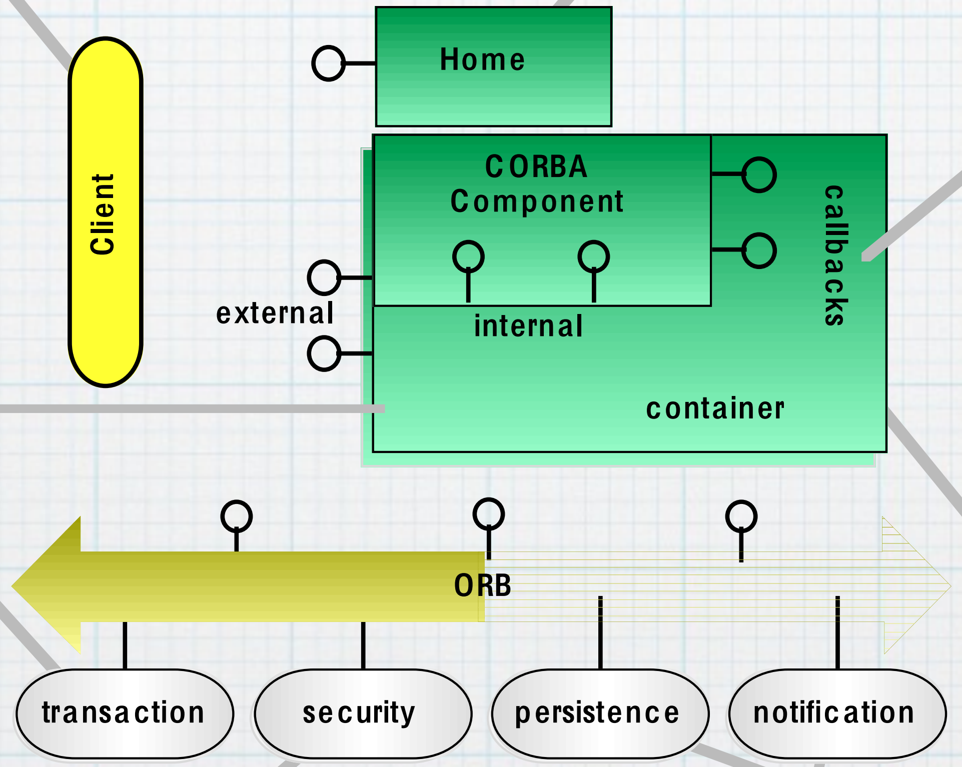
CORBA™ Component Model Interfaces and Services

client side functionality acting as the component's agent

"factory" operations (create, destroy) for entity components, the "housecleaning" interface

functions responding to servant requests for internal component and server resources.

provides access to transaction processing for begin/end trans and recoverability.



provides access to PSDL to store and recover the persistent state of a component instance.

provides component security access/privileges via configuration file

events managed by the container

CORBA™ Component Services

- Transaction
  - defines component instances and protocol for client transaction management
- Security
  - access, deployment, permissions
- Events
  - notification of defined component and transaction events
- Naming
  - support component finding components
- Persistence
  - container-managed persistence, saving and restoring component state from persistent state

CORBA™ Component Types

- • Service
  - used for a single service call, a self- contained function with simple result
- Session
  - defines an ongoing relationship with client during system up-time, yet transitory
- Process
  - a reliably persistent object possible aligned to a transaction
- Entity
  - represents truly persistent item such as customers, account, etc. closely aligned to database & transactions

Component Issues

- Platforms
    - transportable vs. reproducible component
- Architecture
    - framework dependency (DCOM,CORBA..)
- Specificity
    - what size should a component be?
- Versioning
    - inter-component compatibility and support
- Quality
    - immutability vs. extensibility, side-effects, documentation, testing

The Next Logical OO Step

* focuses on reuse of existing software rather than software development
* extends the OO paradigm benefits of reuse and modeling to net-based
* decentralizes the construction of complex, distributed systems
* extends the promise of "software-ic" to the distributed enterprise
* enables exploration/exploitation of connectivity (lan, wan, web, net)
* creates a new software industry segment and consulting arena