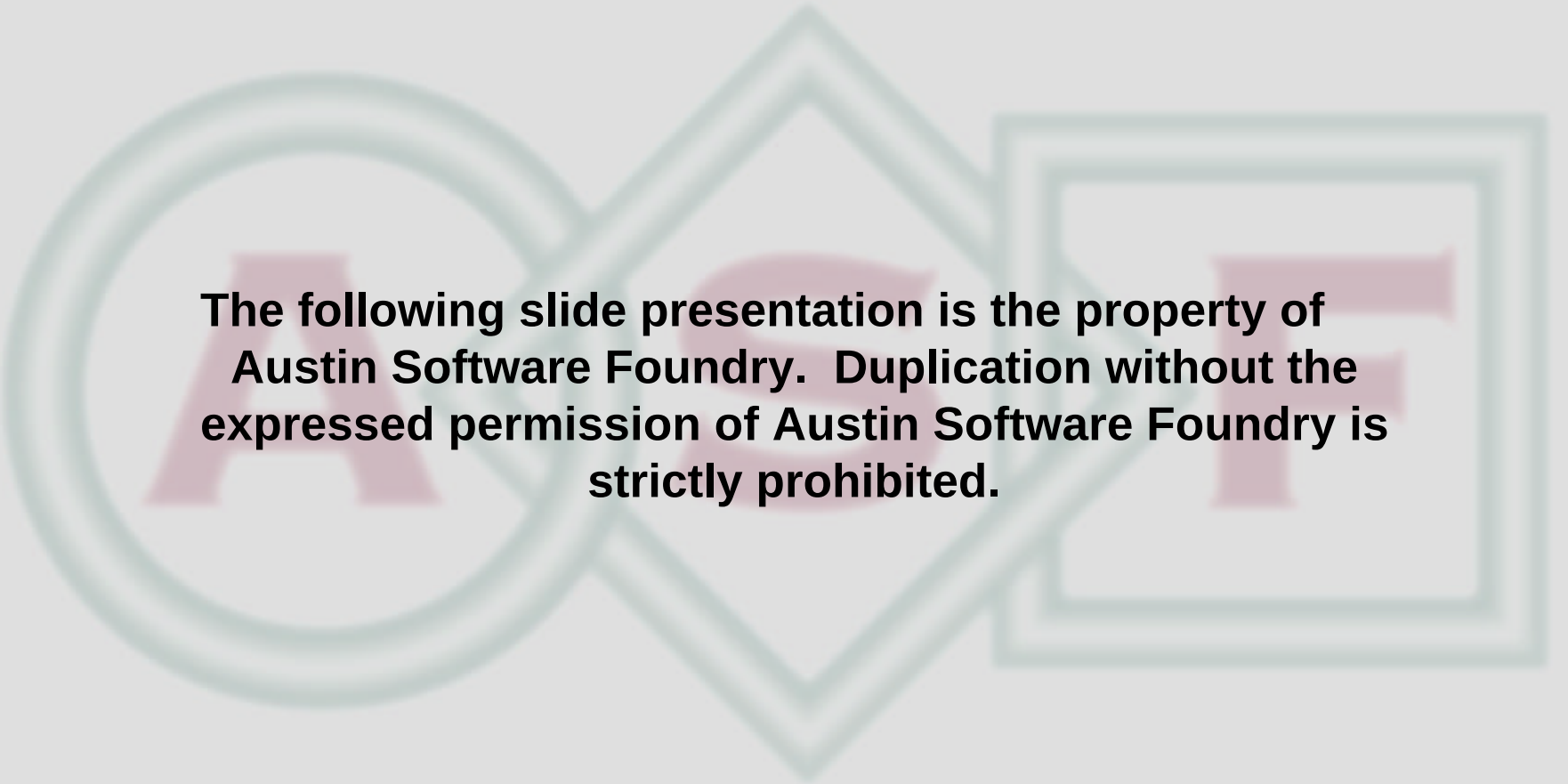


The following slide presentation is the property of Austin Software Foundry. Duplication without the expressed permission of Austin Software Foundry is strictly prohibited.

Industry Direction: A Software Component Industry

December 5, 1996

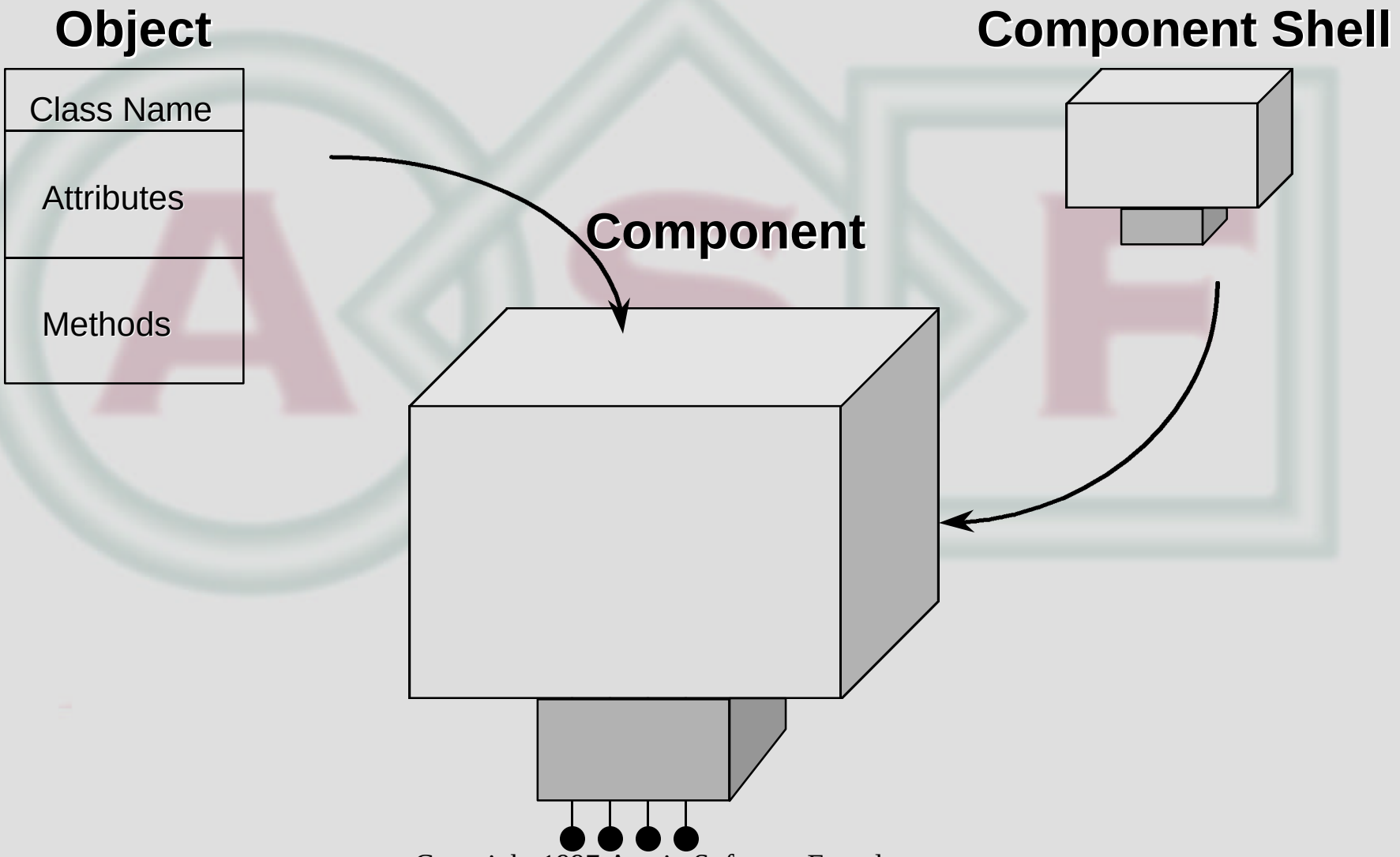
Overview

- **What is the Software Component Industry?**
- **Current Trends in the Software Component Industry**
- **Implications for Application Development**

What is the Software Component Industry? Objects vs. Components

- **Characteristics**
  - **Objects encapsulate data and behavior for a single entity**
  - **Components encapsulate, or package, objects**
- **Implementation**
  - **Objects are implemented using object-oriented programming languages, tools and techniques**
  - **Components are implemented, or manufactured, using interface definition and object binding tools**

What is the Software Component Industry?
Objects vs. Components



What is the Software Component Industry?
Software Components

Definition: A reusable, self-contained piece of software that is independent of any application.

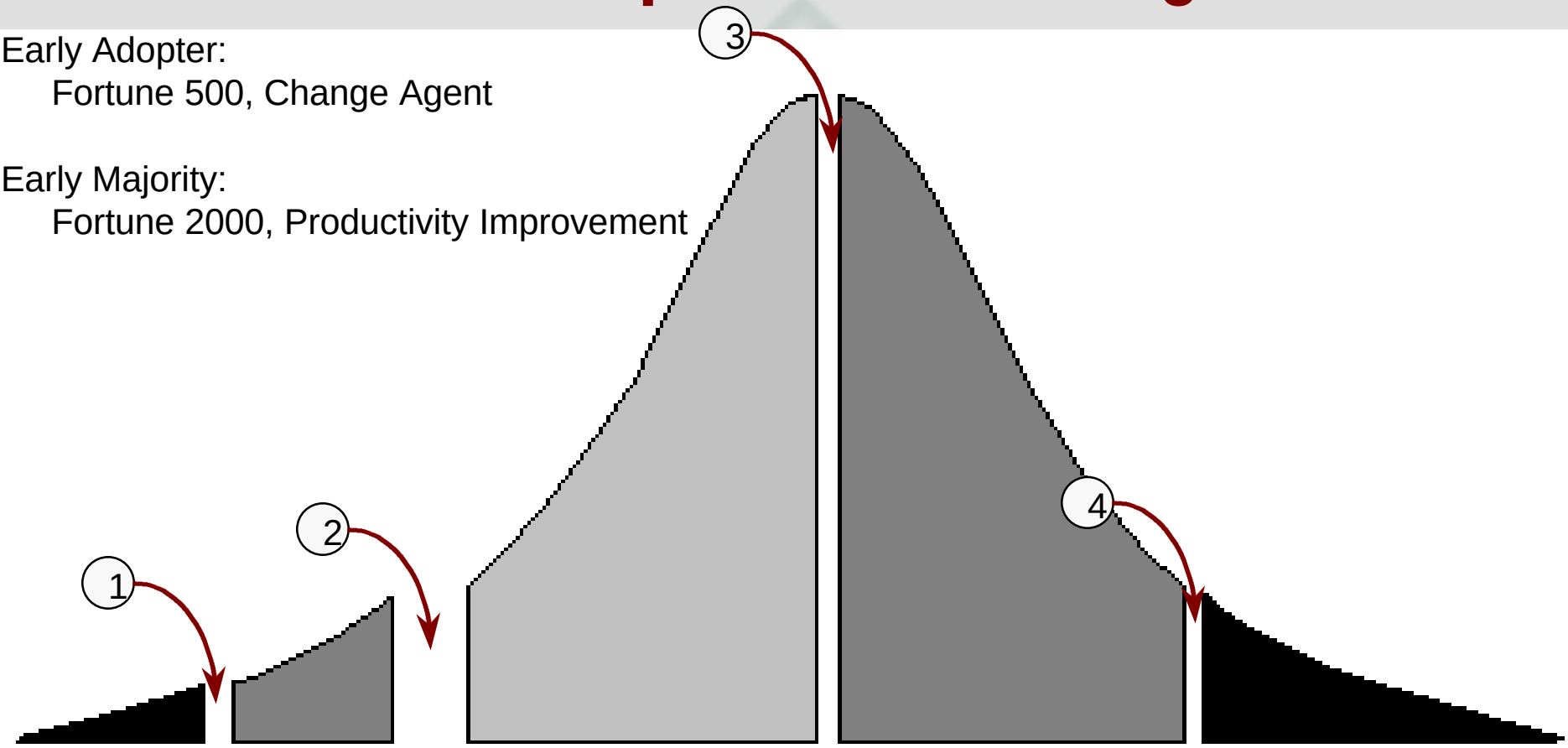
- **Form: binary, self-contained piece of software**
- **Size**
  - **Fine: individual C++, PB, Java object**
  - **Medium: GUI control, output service**
  - **Large: applet or whole application**
- **Usage: combined and used in various ways locally or across networks**
- **Interface: manipulated only through its published interface**
- **Extension: can be extended through inheritance, aggregation, polymorphism**

What is the Software Component Industry?

Components Crossing the Chasm

Early Adopter:
Fortune 500, Change Agent

Early Majority:
Fortune 2000, Productivity Improvement

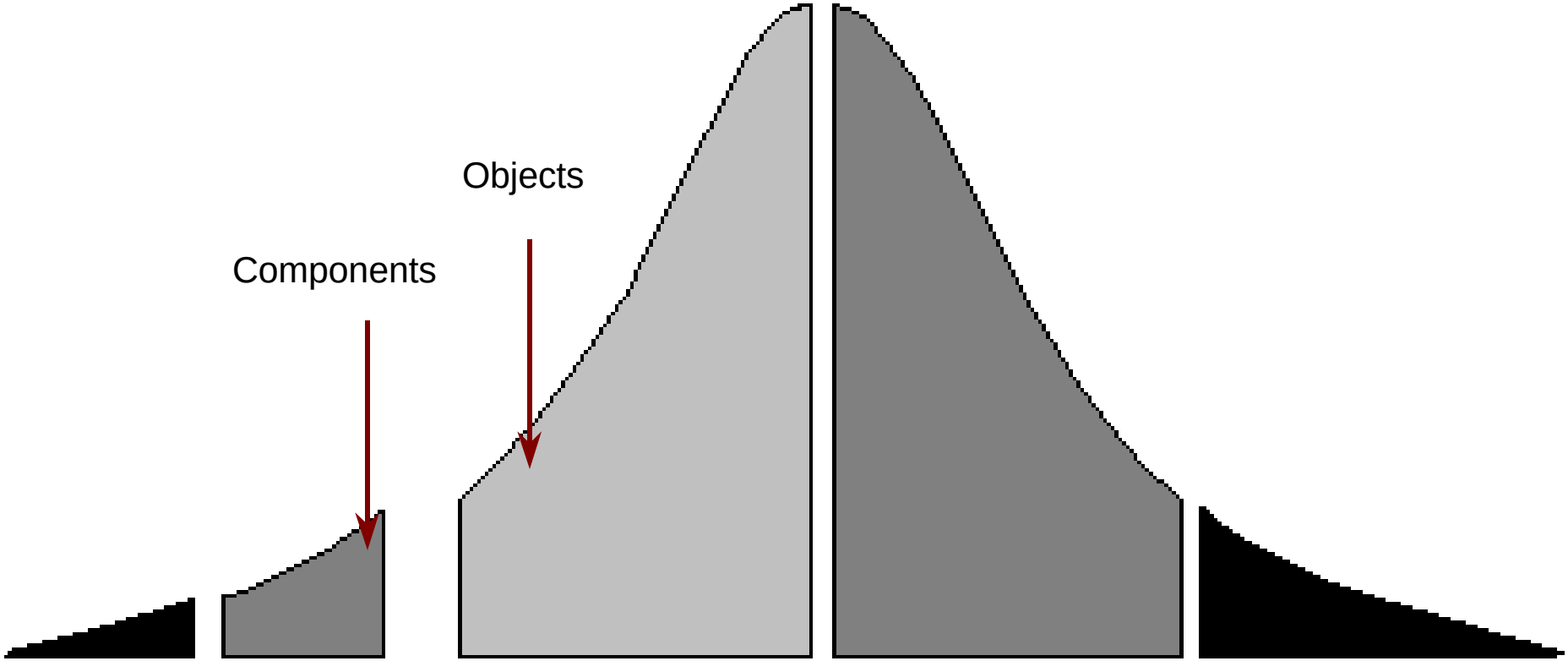


Innovator | Early Adopter | Early Majority | Late Adopter | Laggards

What is the Software Component Industry?
Components Crossing the Chasm

- (1) Hot technology cannot be translated into a major new benefit
- (2) Product doesn't have an extensive reference account base and large support industry
- (3) Product cannot be made easy enough for the average user
- (4) Product cannot be turned into a pure commodity and produced cheaply enough

What is the Software Component Industry?
Components Crossing the Chasm



What is the Software Component Industry?
Categories of Component Products

- Container Components representing “virtual places” (today - PowerBuilder and VB)
- Object Components representing “people and things” (today - VBXs, OCXs, PB Objects)
- Component suites representing related groups of Container and Object Components
  - Preassembled Suites - prepackaged for a particular industry or application (today - Word, Excel)
  - Built-to-Order Suites - custom suites created from an existing catalog and assembled, tested, then shipped (today - PeopleSoft, PowerCerv)

What is the Software Component Industry?
Players

- Component Producers
  - Corporate IT "Object Technology Centers"
  - Specialized Component Software Companies
  - Packaged Application Software Companies' "Component Departments"
  - Software Development Tool Vendors' "Component Divisions"
- Component Consumers
  - All of the Component Producers
  - Corporate IT Departments
  - Line-of-Business IT Departments
  - Packaged Application Software Developers
  - Software Development Tool Vendors

What is the Software Component Industry?
Structure

- **Standards Groups (CI Labs, OMG, ActiveX Working Group)**
  - **Specify and document standards for industry specific vertical containers (places) and suites**
  - **Certification Testing**
  - **Evangelizing**
- **Market Infrastructure**
  - **Component distribution and sales channels**
  - **Support**
  - **Assembly and Integration**
  - **Active partnerships and cross-licensing among vendors**
  - **Consulting, Education and Tools**

Overview

- **What is the Software Component Industry?**
- **Current Trends in the Software Component Industry**
- **Implications for Application Development**

Current Trends in the Software Component Industry

- **Unification and widespread use of OOA&D methods**
- **Substantive progress on business objects**
- **Focus on larger-grained reusable artifacts**
- **Tool and technique support for iterative development processes**
- **Deployable distributed object architectures**

Unified Object Modeling

- **OOPSLA in Austin, Texas - October, 1995**
  - **Rational Software announces that James Rumbaugh and Ivar Jacobson have become Rational Fellows**
  - **Rational Software announces the merging of OMT, Objectory and the Booch Method into the Unified Modeling Language**
- **The Unified Modeling Language includes**
  - **OMT as the foundation**
  - **Objectory Uses Cases**
  - **Booch Packages**
  - **Wirfs-Brock Stereotypes**
  - **Coad/Yourdon Abstract Classes**
  - **Meyer Contracts**
  - **etc.**

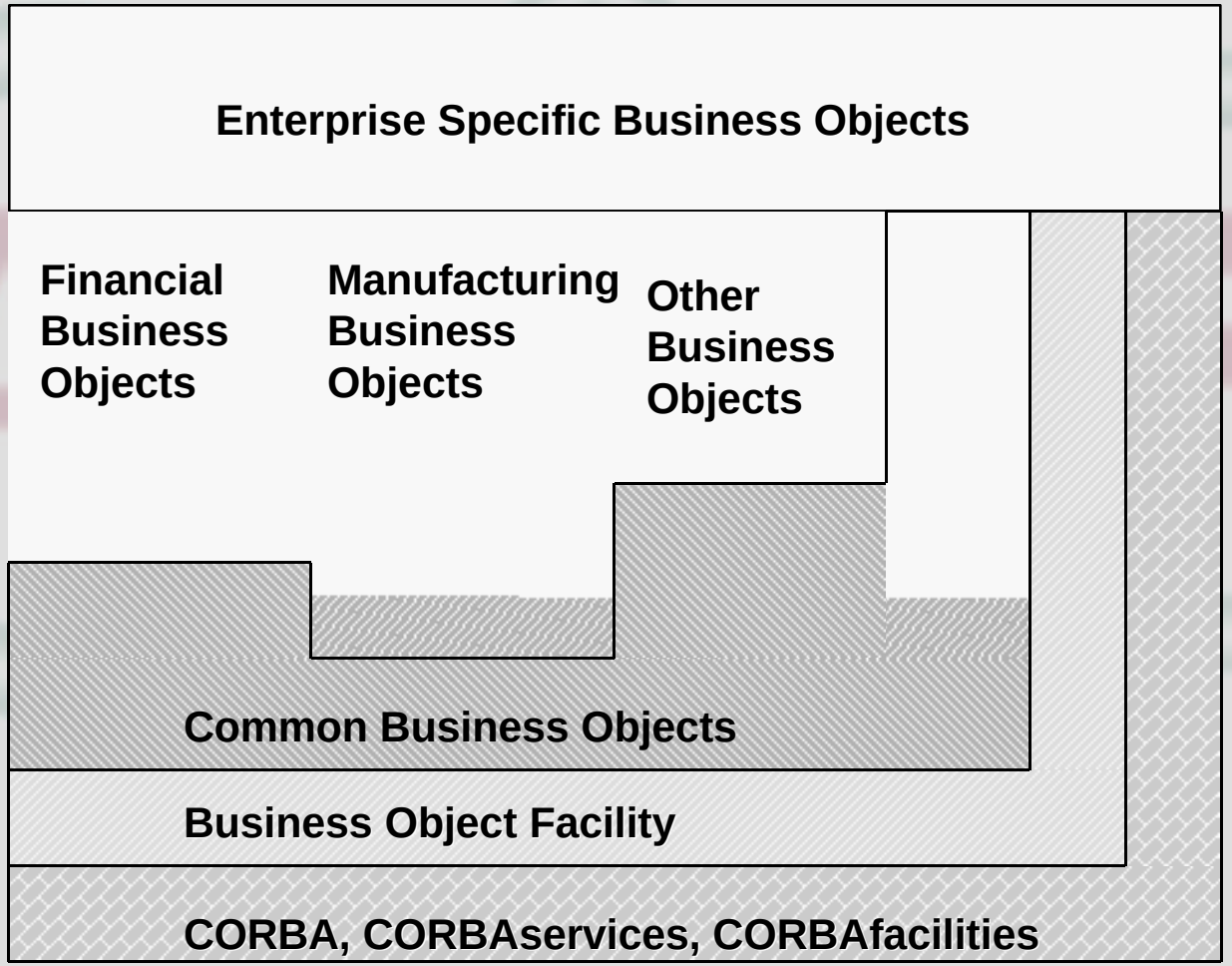
Current Trends in the Software Component Industry

- Unification and widespread use of OOA&D methods
- **Substantive progress on business objects**
- Focus on larger-grained reusable artifacts
- Tool and technique support for iterative development processes
- Deployable distributed object architectures

OMG Business Object Domain Task Force

- **Container Standards: Business Object Facility**
  - **Common infrastructure for business objects**
  - **Horizontal focus**
  - **Interface to the underlying CORBA services**
- **Object Standards: Common Business Objects**
  - **Common objects used in all businesses such as “legal entity”, “resource”, “transaction”**
  - **Standards for isolating user interface and data storage from business policies and rules**

OMG Business Object Domain Task Force



Current Trends in the Software Component Industry

- Unification and widespread use of OOA&D methods
- Substantive progress on business objects
- **Focus on larger-grained reusable artifacts**
- Tool and technique support for iterative development processes
- Deployable distributed object architectures

Object-Oriented Design and Programming: Too Confusing?

- **Flow of control is decentralized** into many collaborations, or messages, between many classes
- Fundamental building blocks are higher level **abstractions**
- Structural **relationships can seem complicated**, i.e. inheritance, aggregation, association
- Architectural relationships can be **complex to implement**, i.e. application partitioning
- Learning curve is **long and steep**
- "Any sufficiently advanced technology is indistinguishable from **magic.**" *-- Arthur C. Clarke's 3rd Law of Technology, "Profiles of the Future: An Inquiry into the Limits of the Possible"*

Corporate Reuse Libraries: Too Complex?

- Requires **high overhead** to create it, and to maintain standards/quality
- **Difficult to search** through large, fine-grained libraries of classes
- Significant architectural **design work still remains** to combine individual classes from the library into applications
- Searches are usually done at coding time, and this is often **too late** to introduce reuse into a project
- **Incentives often don't exist** for contribution to, and reuse from the library

Alternative to Confusion and Complexity
Simplicity through Abstractions

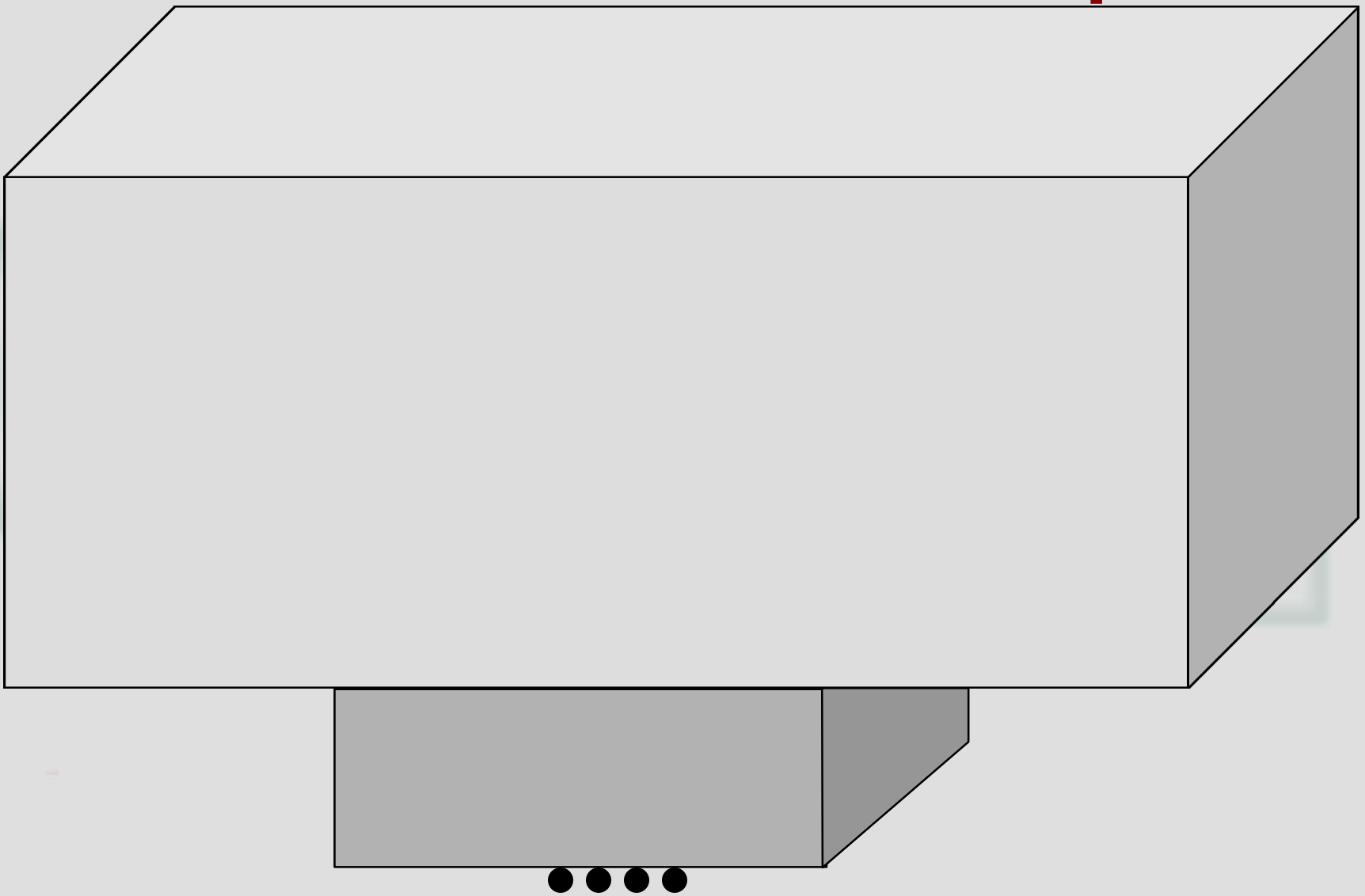
Physical Elements

- Application
- Framework
- Component
- Object

Logical Elements

- Architectural Pattern
- Meta-Pattern
- Pattern
- Class

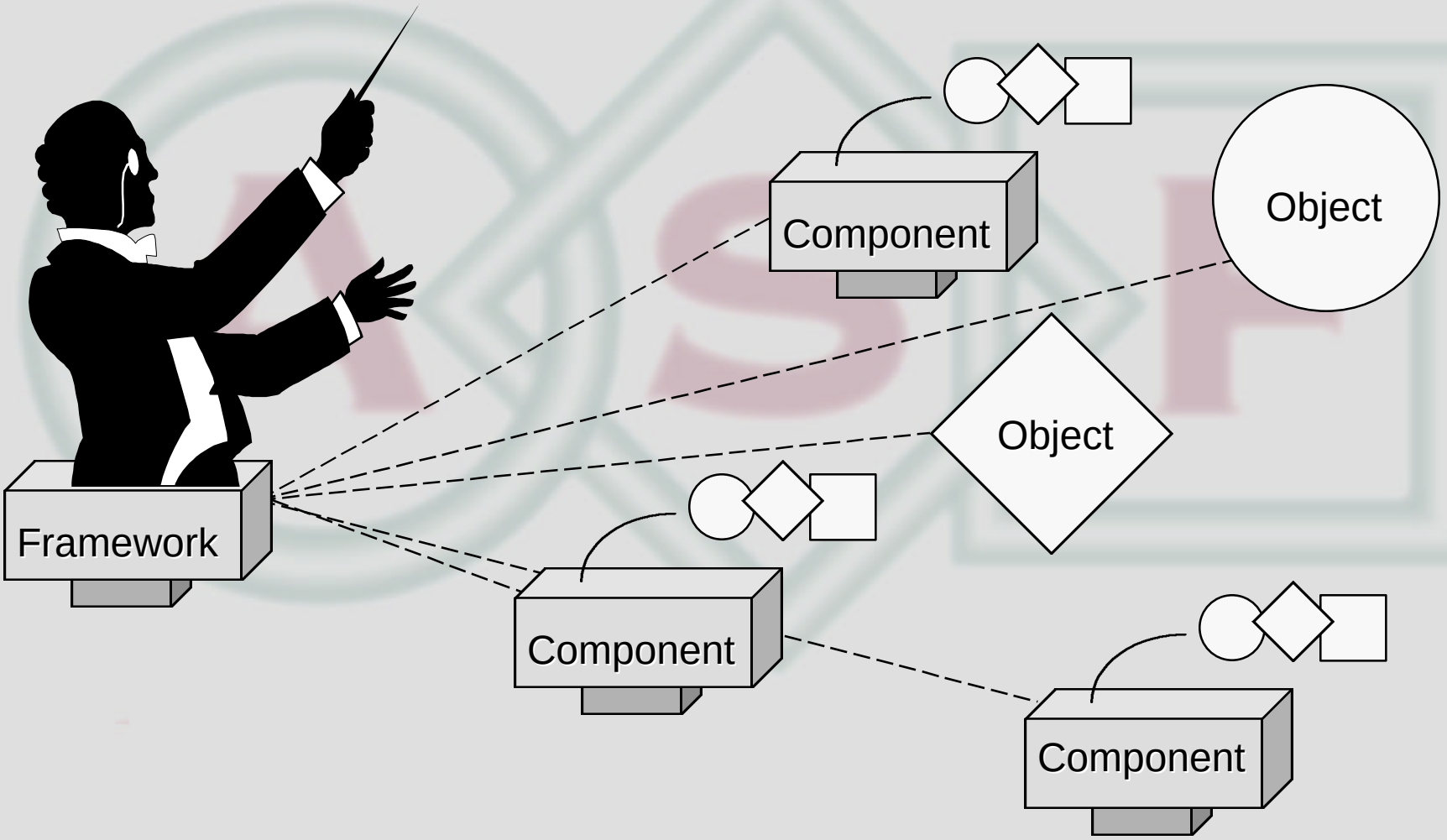
Patterns and Components



Published Patterns

- Domain Patterns
  - Business Context/Entity/Rule
  - Input/Activity/Output
  - Transaction
- Design Patterns
  - Partitions
  - Layers
  - Publisher/Subscriber
  - Model/View/Controller

Meta-Patterns and Frameworks



Current Trends in the Software Component Industry

- Unification and widespread use of OOA&D methods
- Substantive progress on business objects
- Focus on larger-grained reusable artifacts
- **Tool and technique support for iterative development processes**
- Deployable distributed object architectures

Iterative Systems Development

- Rational Software
  - Rational Rose/PB 4.0
  - Round-Trip Engineering
- DSDM Consortium
  - User Centric RAD
  - Incremental Prototyping
  - Standards and Principles for control
- Hybrid Development Processes
  - ASF ASAP
  - Select Software Select Perspective

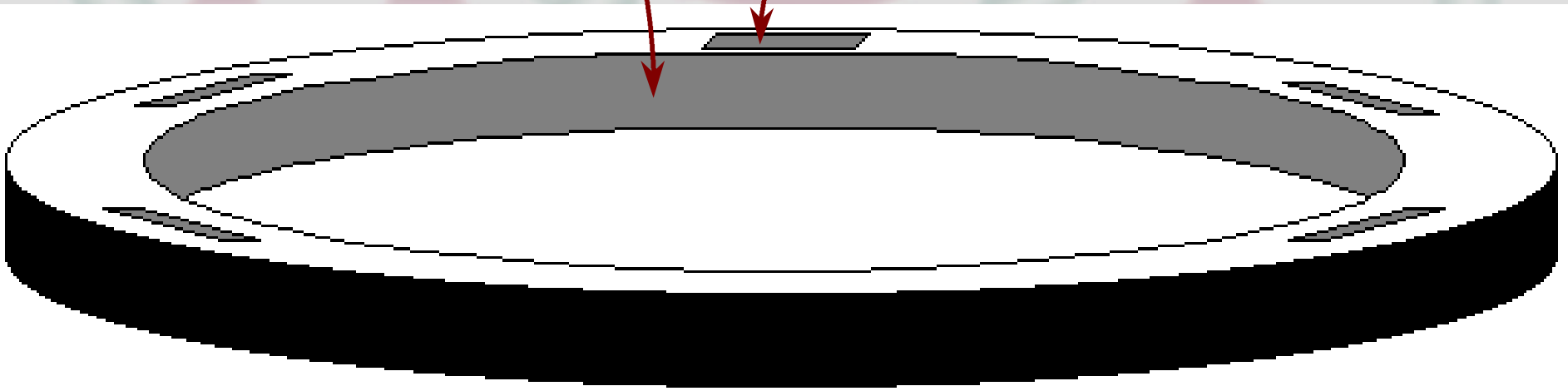
Current Trends in the Software Component Industry

- Unification and widespread use of OOA&D methods
- Substantive progress on business objects
- Focus on larger-grained reusable artifacts
- Tool and technique support for iterative development processes
- **Deployable distributed object architectures**

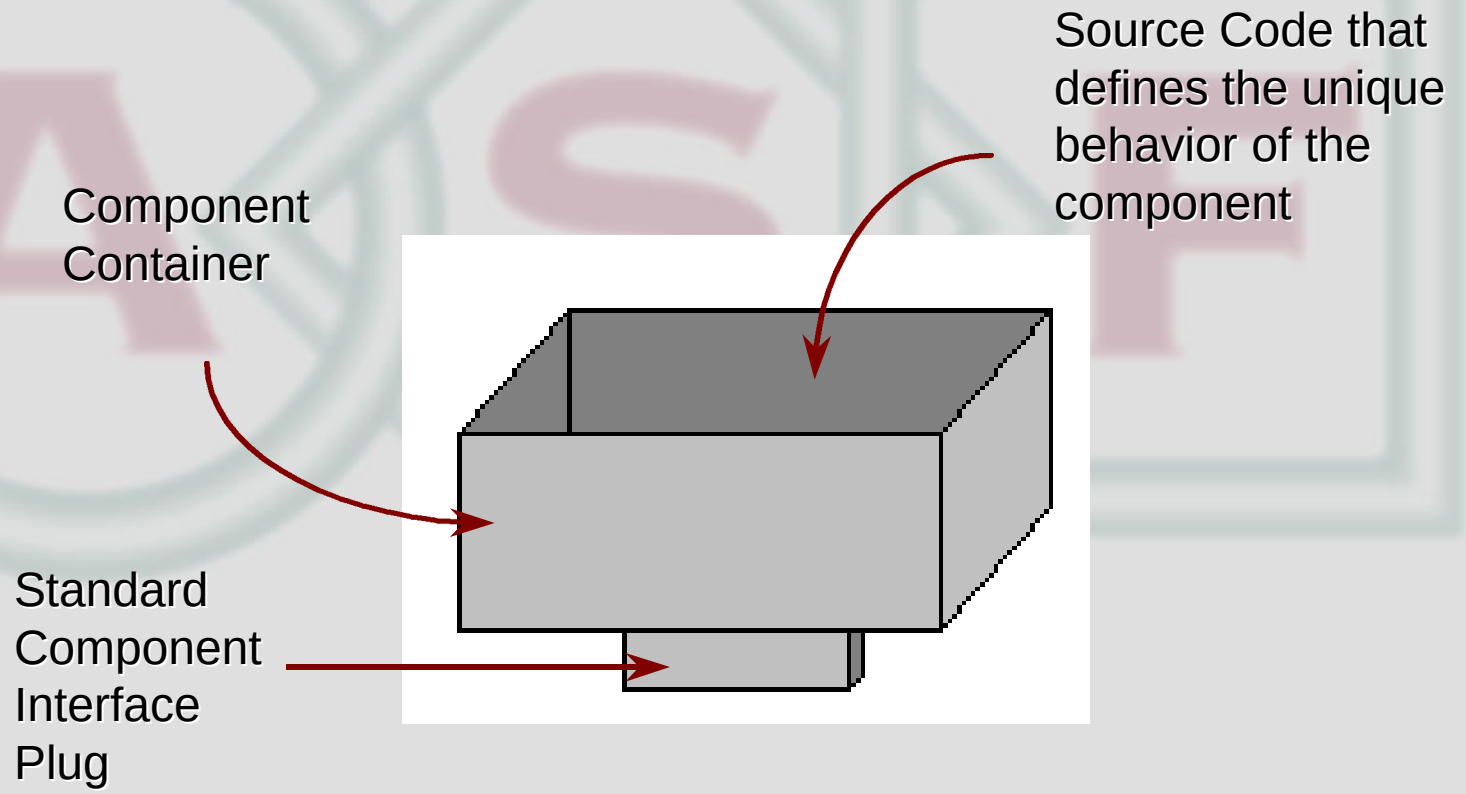
Distributed Components
The Communications Infrastructure

Standard Communication Services
Standard System Services:
- Trans. Mgmt.
- Event Mgmt.
- Security
- etc.

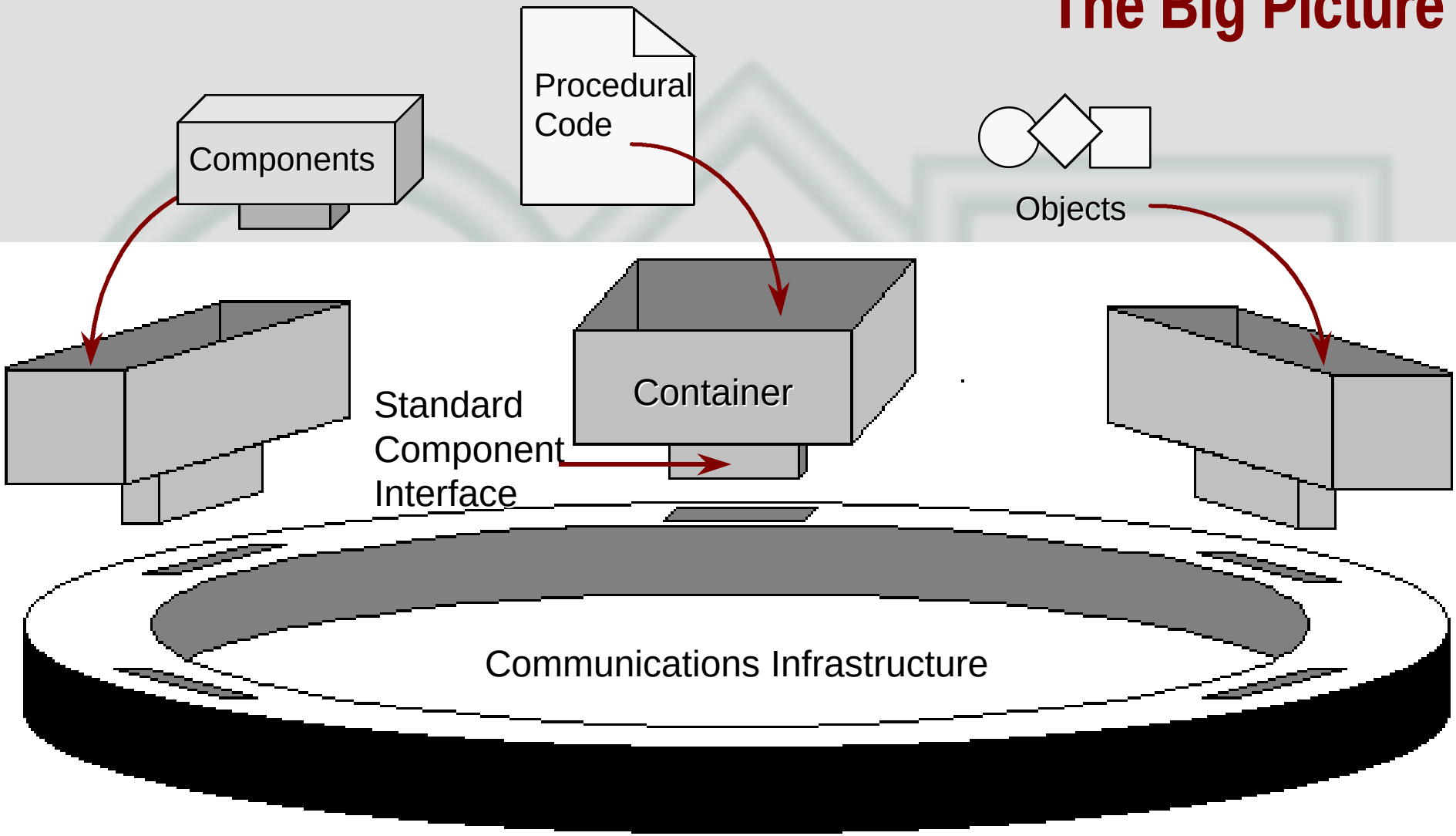
Standard
Component
Interface
Socket



Distributed Components
The Component Infrastructure



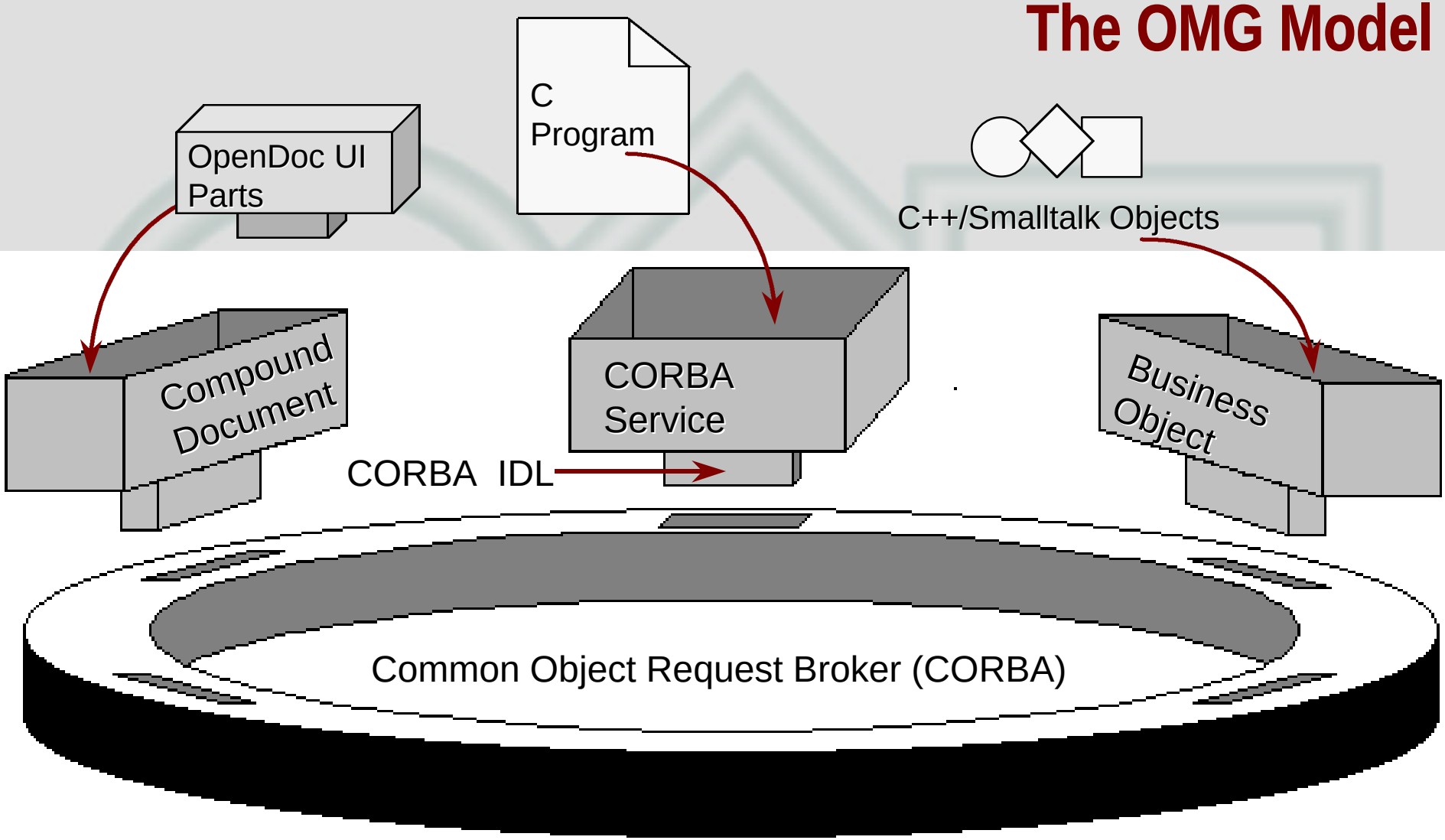
Distributed Components
The Big Picture



Distributed Components
The Standard Component Interface

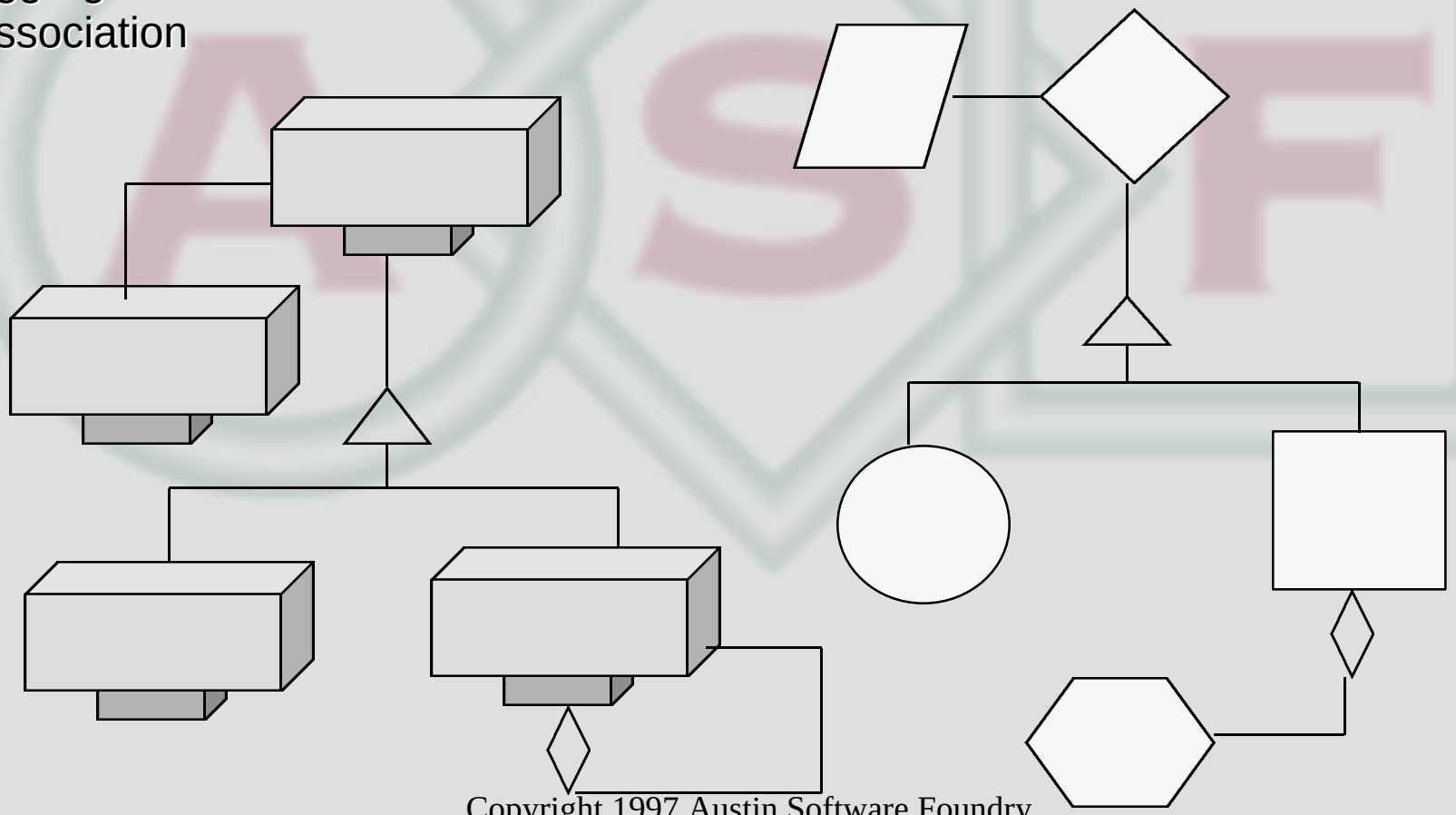
- The Standard Component Interface defines a binary standard for heterogeneous components to send messages to each other, and requires:
- Object Definition (compiled object interfaces) Database
  - CORBA = Interface Repository
  - COM/DCOM = Type Library (.TLB)
- Runtime Object Reference (pointer and type) Database
  - CORBA = Implementation Repository
  - COM/DCOM = OLE Registry (.REG)

Distributed Components
The OMG Model

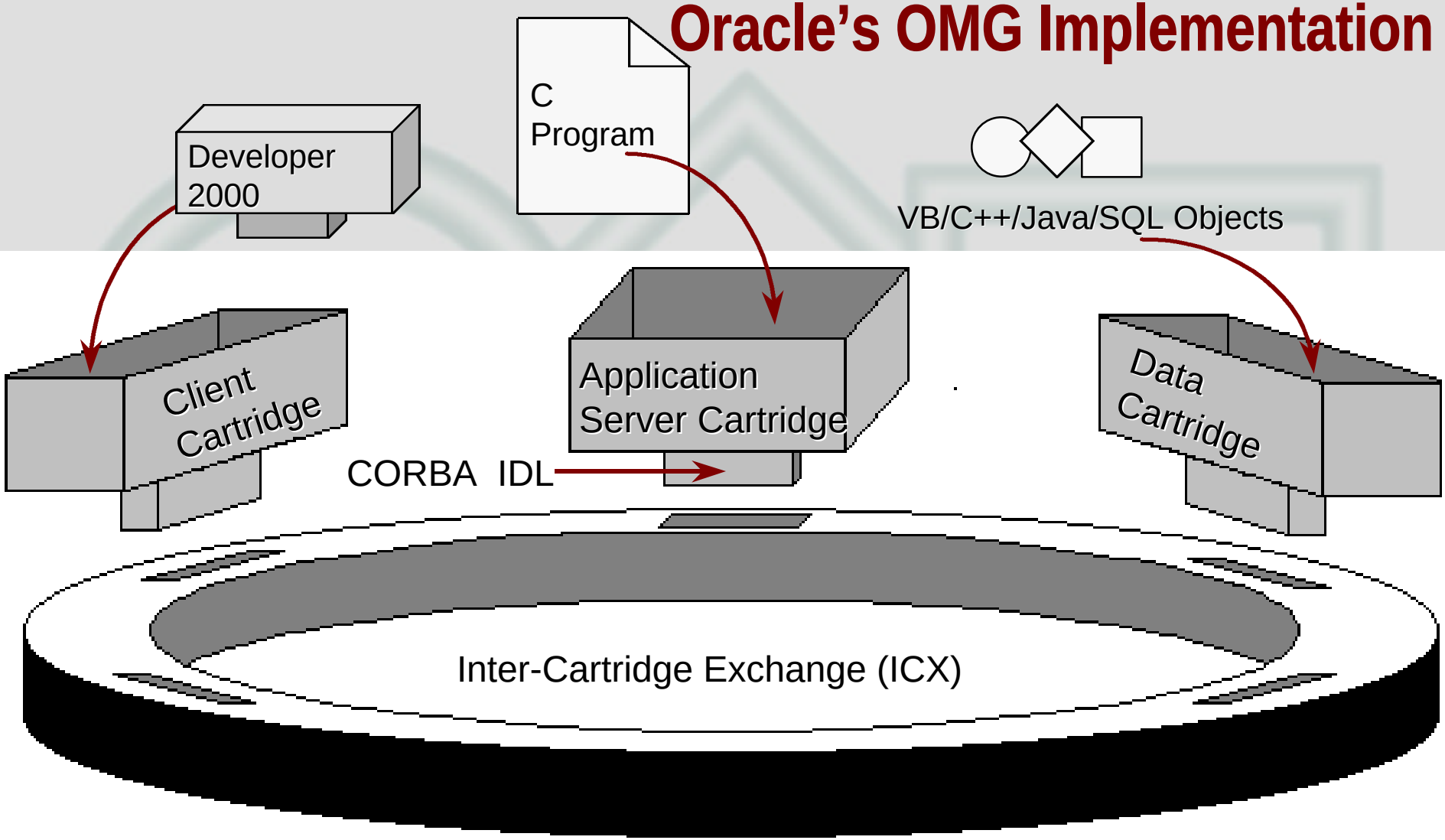


Distributed Components
The OMG Model

Implementation Inheritance
Interface Inheritance
Polymorphism
Aggregation
Association



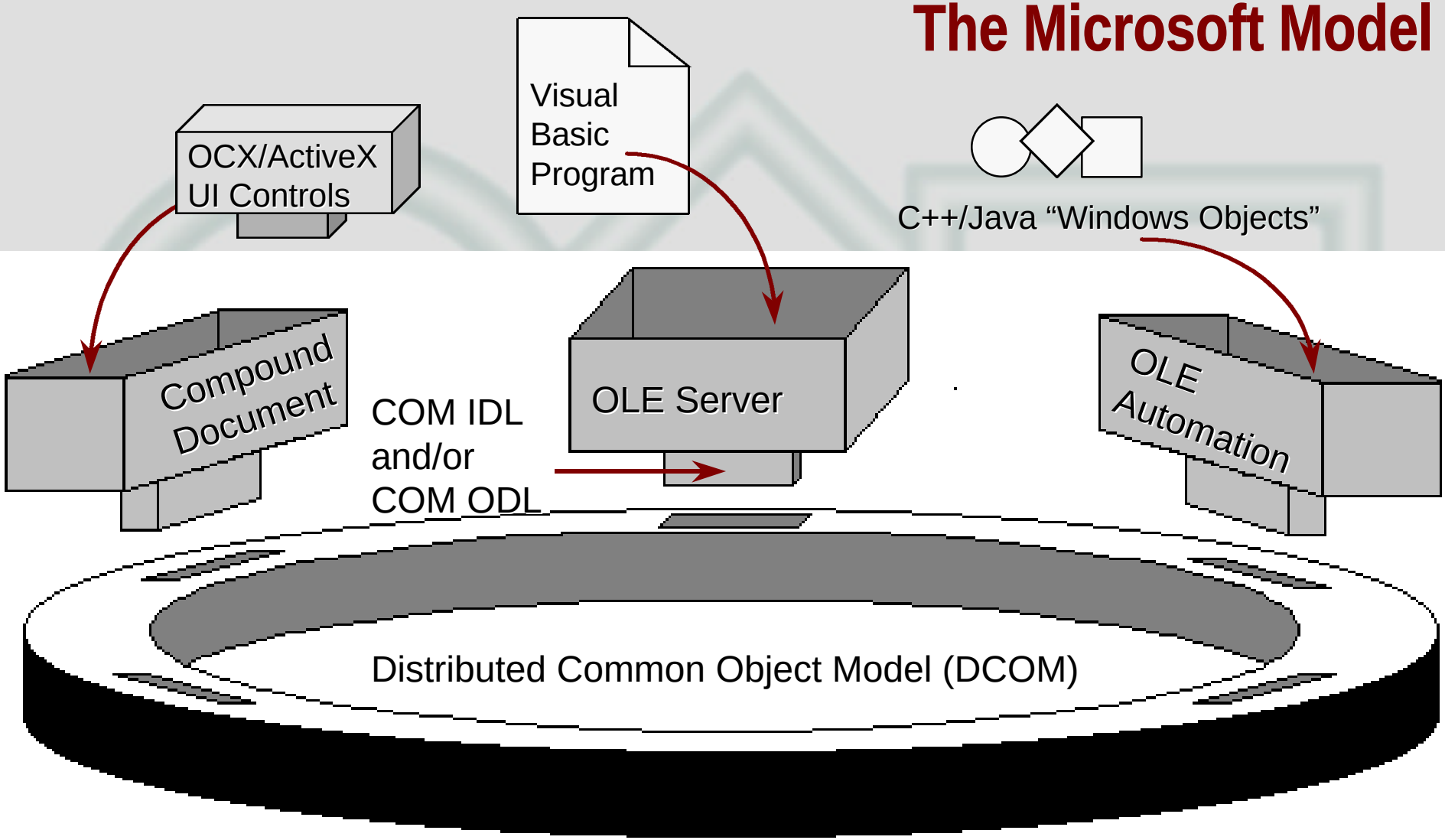
Distributed Components
Oracle's OMG Implementation



Distributed Components
Other OMG Implementations

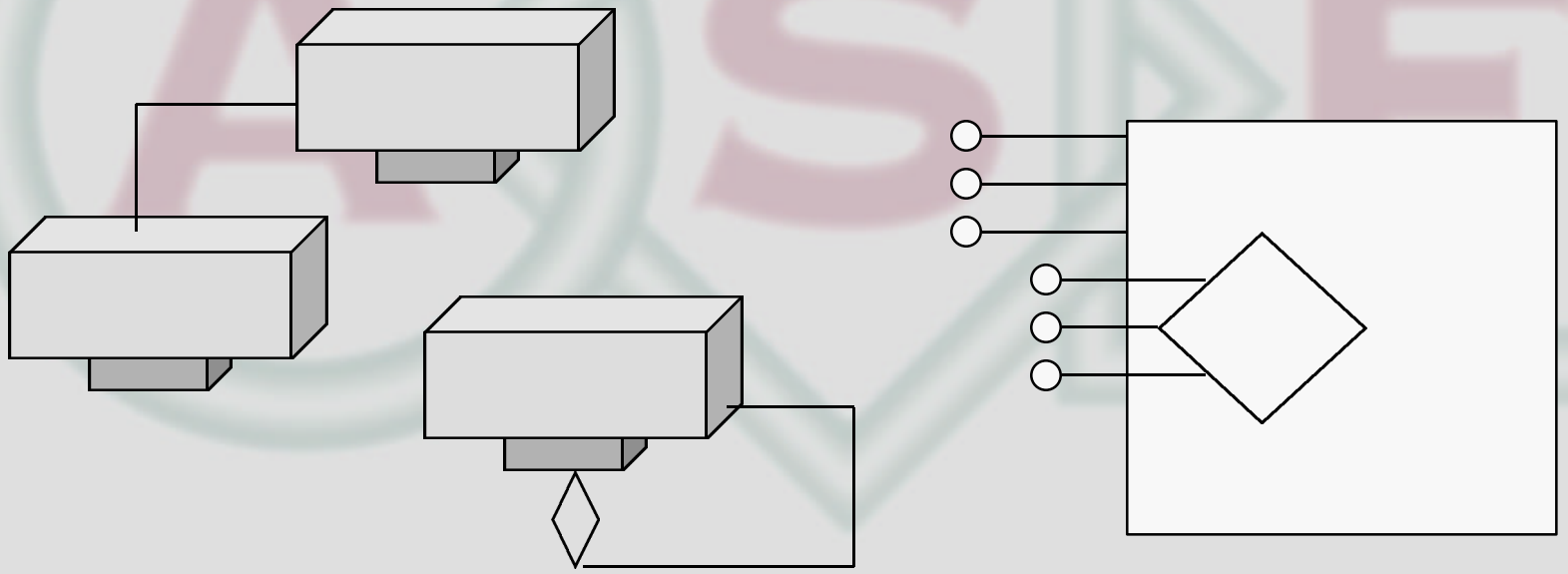
- HP: ORBPlus
- IBM: SOM/DSOM
- Iona: ORBIX
- DEC: ObjectBroker
- SUN: DOE
- And the list goes on...

Distributed Components
The Microsoft Model



Distributed Components
The Microsoft Model

Interface “Inheritance”
Aggregation
Association

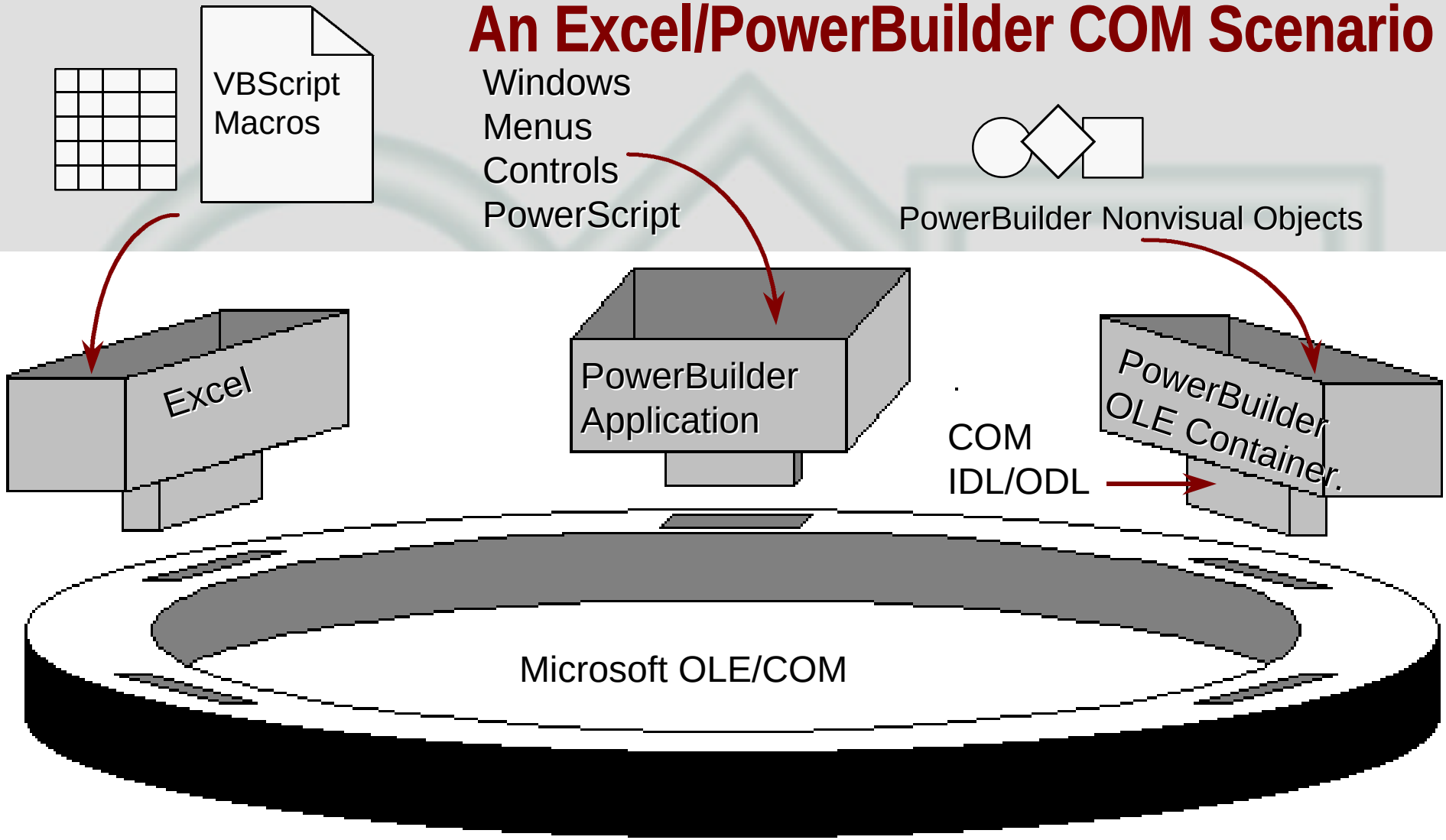


Distributed Components
Other Microsoft Implementations

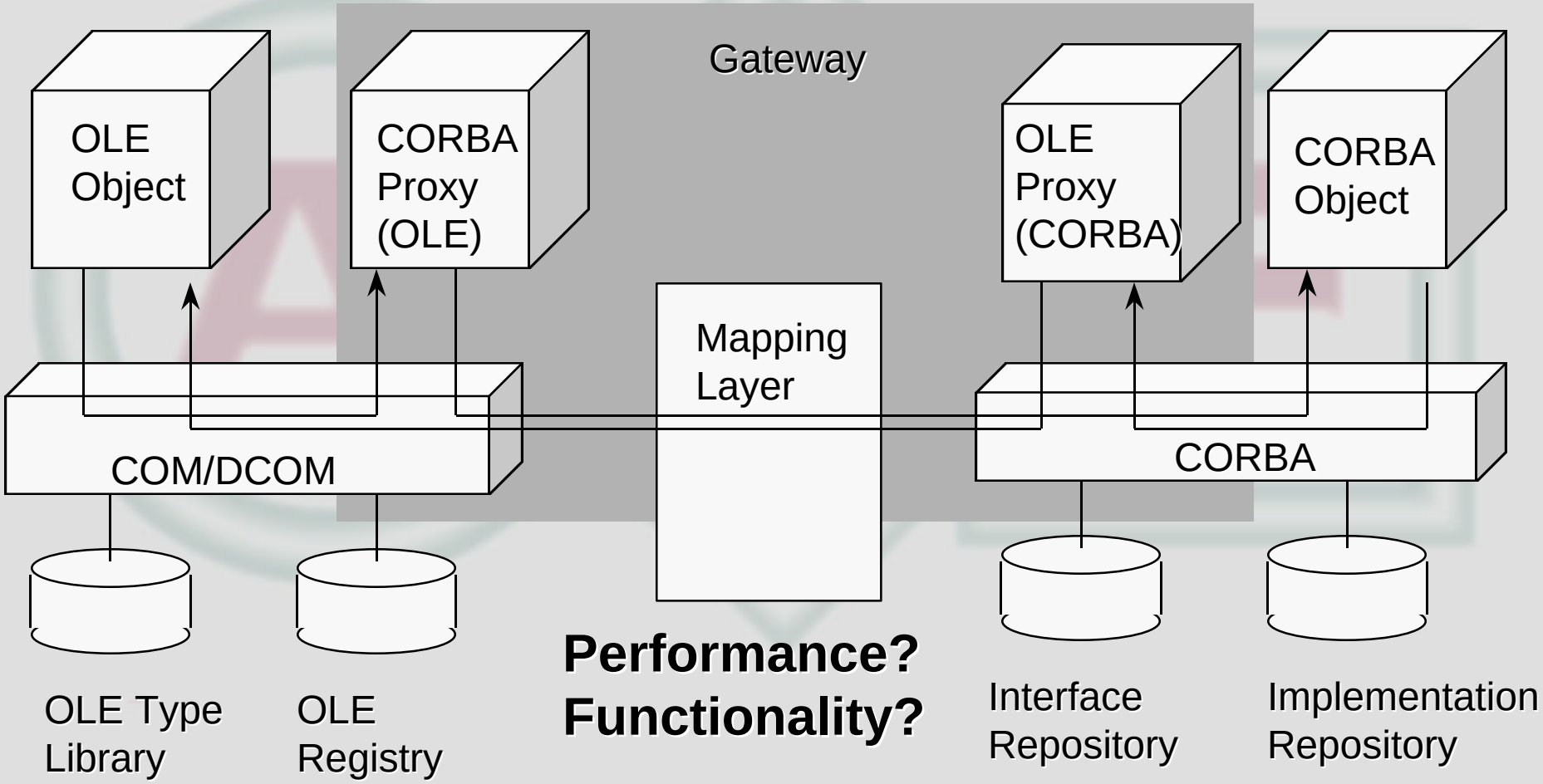
- DEC: ObjectBroker (COM/CORBA hybrid)
- Anyone else?

Distributed Components

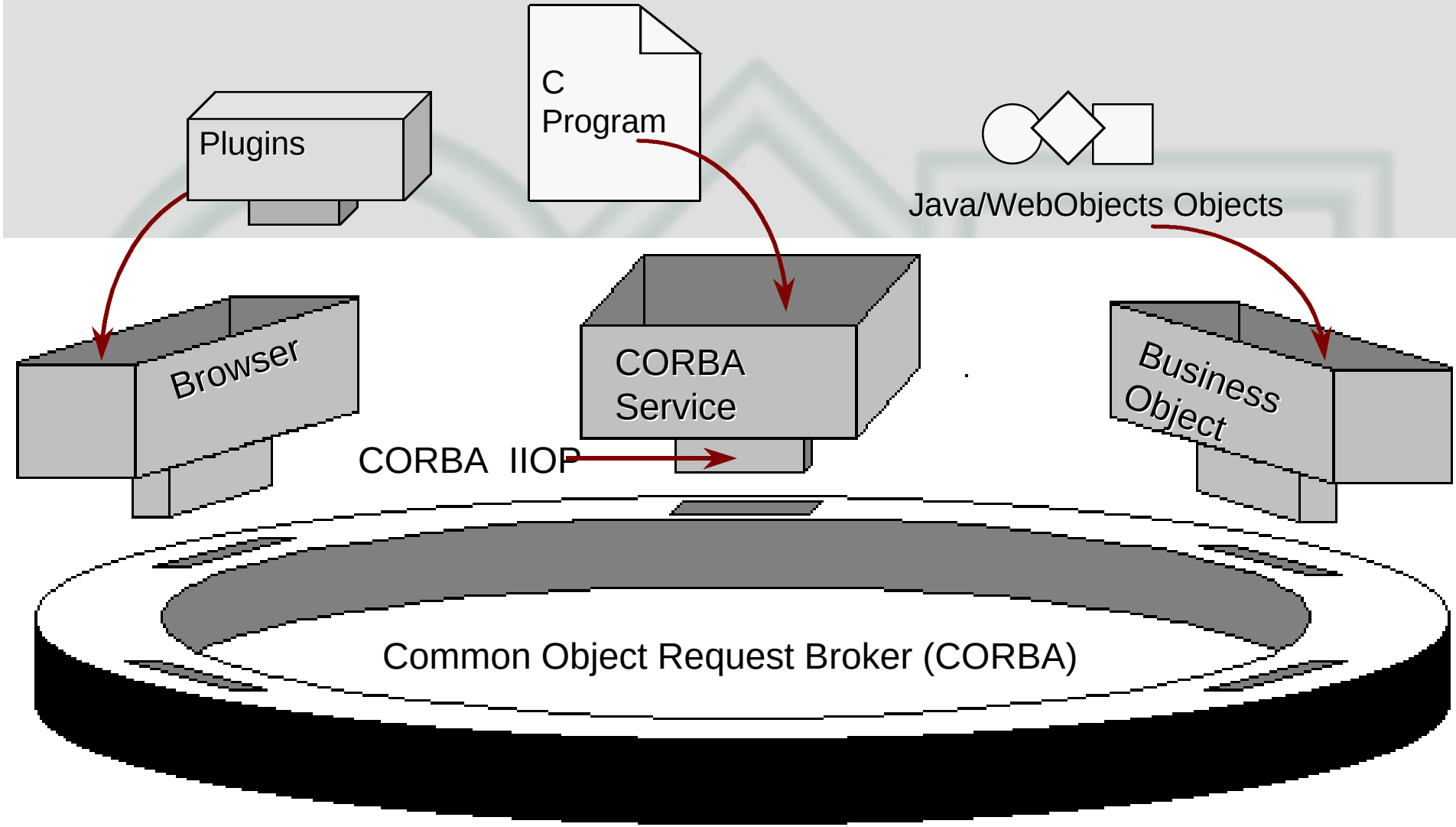
An Excel/PowerBuilder COM Scenario



So Which Distributed Component Model?



And/Or “The Web”?



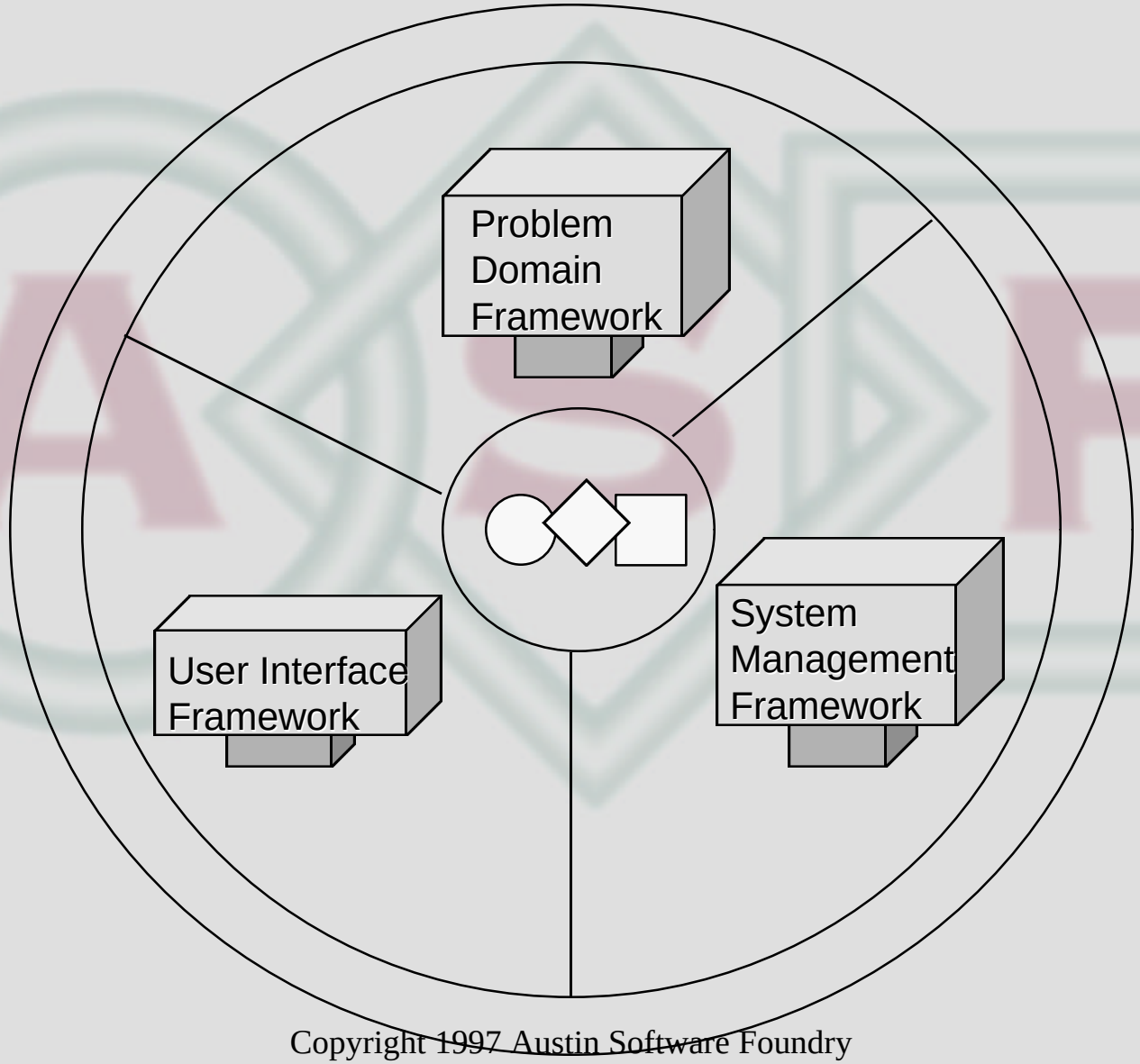
Overview

- **What is the Software Component Industry?**
- **Current Trends in the Software Component Industry**
- **Implications for Application Development**

Implications for Application Development

- Applications are created as suites of collaborating components
- The client side of an application is created as a “container component” representing a visual metaphor consistent with the application
- The server side of an application is created as a series of “domain and service components” that are available either locally or remotely through a communications infrastructure
- OOA, OOD and OOP are used to design and create the contents of the components, and the collaborations between them

Implications for Application Development



Implications for Application Development