

Component-Based Architectures

THE RELATIONSHIP BETWEEN PRICE-PERFORMANCE GAINS AND MODULARITY

Dr. Thomas Kessler

Tom@denaliassociates.com

December 2003

Presentation Objectives

- **Explain component-based approach**
 - ◆ **Big changes in software architectures**
 - ◆ **Major vendors are embracing component-based design**
 - ◆ **Component-based design**
 - ☞ **What is it?**
 - ☞ **What is it good for?**
 - ☞ **What do I need to do to get ready for it?**
- **Motivate interest in component-based approach to software development**
- **Encourage organizations to anticipate trend and adequately positioning themselves**

Modularity is not a new concept – we have consistently segregated applications according to functionality

Context

Mainframe-Batch



Mainframe-Online



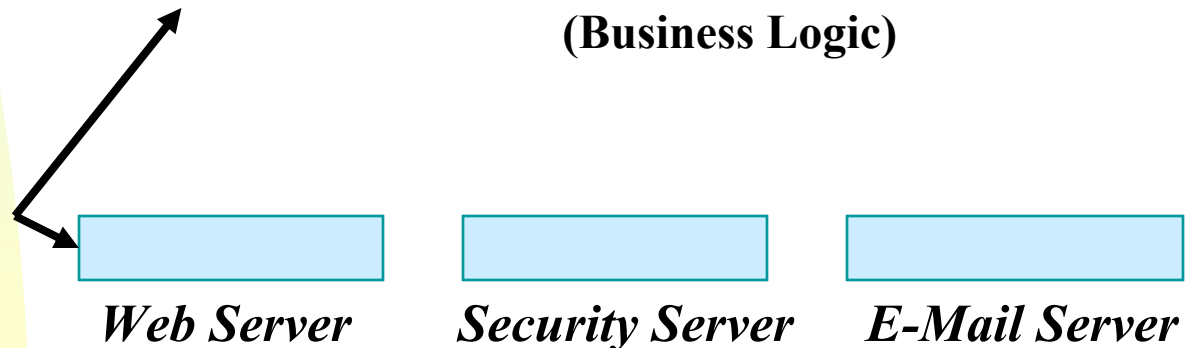
Client Server 2-tier



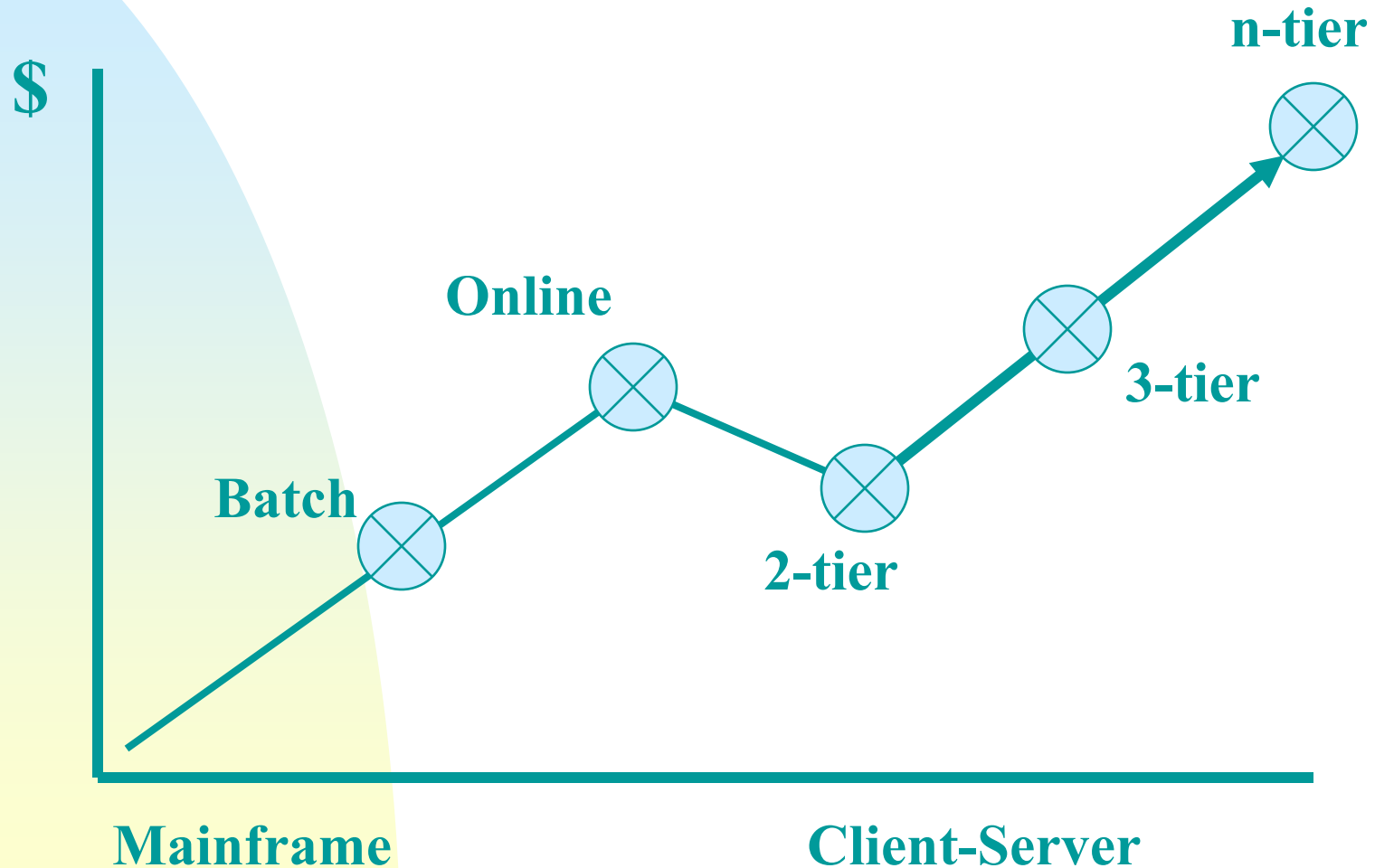
Client Server 3-tier



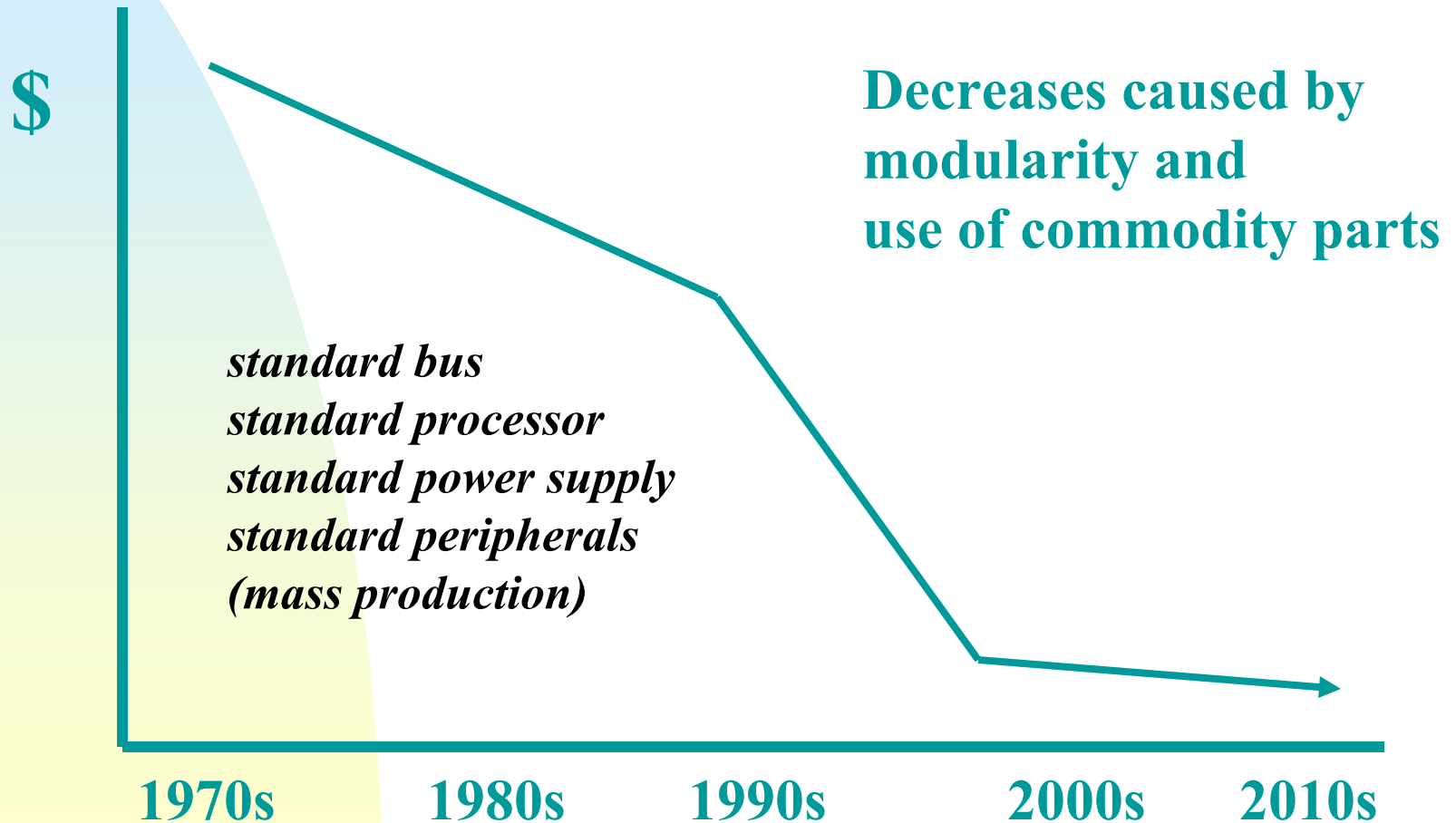
n-tier / multi-tier



Software Economics



Hardware Economics



Component benefits

- Reduce development time
- Improve time to market
- Build flexible applications
- Simplify portability
- Improve performance, availability, and scalability
- Focus on unique needs
- Mix and match solutions
- Operate across diverse infrastructures
- Modify quickly
- Lower cost of code maintenance
- Support new business models
- Deliver value to your organization faster

Dilemma: Can We Ever Produce Commodity Software?

- Tried it with COTS/GOTS
 - ◆ Some success, but applications are expensive, frustrating, and difficult to install
 - ◆ Tried it with CASE tools, but life cycle is still life cycle
 - ◆ Succeeded in some areas
 - ☞ Office automation (sort of)
 - ☞ Basic, simple applications such as QuickBooks
 - ☞ Operating systems (sort of)

Are Components Really A New Concept?

- Isn't this just “old news?”
 - ◆ We have statically and dynamically linked to sub-routines and standard routines for many years
 - ◆ Reusable function libraries
- Answer – Components are different
 - ◆ Allows self-contained binary modules to be created by independent developers
 - ◆ Components are “plugged into applications at runtime
 - ◆ Once a component has been written, it can be used by an unlimited number of applications
 - ◆ Components can be written in any language and used by applications written in the same or other languages

Architecting A Solution

- What products, tools, and techniques are necessary in order to achieve dramatic reductions in software cost curves?
 - ◆ Develop standard software modules (components) that are small enough to avoid COTS/GOTS problems
 - ◆ Distribute components across heterogeneous environments in ways that enable them to work together to process a transaction

Issue 1: Components

- Component is unit of packaging, distribution, and maintenance
- It is part of an application
- Distributed components contain intelligence that can be distributed across networks
- Can we develop standard software components and use them easily, quickly, and without additional procedural code?

Developing Standard Software Components

- **Recognize observed, common, repeating patterns, at all levels of granularity**
 - ◆ Subtract one date from another date
 - ◆ Determine if elements of a person's address are reasonable, error-free, and complete
 - ◆ Register a student for a course
 - ◆ Sent an automated e-mail to someone
 - ◆ Schedule a patient for a visit
 - ◆ Update someone's calendar

Can I Build Software Libraries?

- In fact, this is a primary benefit of object-oriented programming languages
 - ◆ Reuse
 - ◆ Modularity (Coupling)
 - ◆ Insularity (Cohesion)
- Java is a good example
 - ◆ Java Class Libraries (Next page)
- www.ibm.com/developer
 - ◆ 360,000-plus Java code examples
 - ◆ 160,000 Extensible Markup Language Examples (XML)

Java Class Libraries

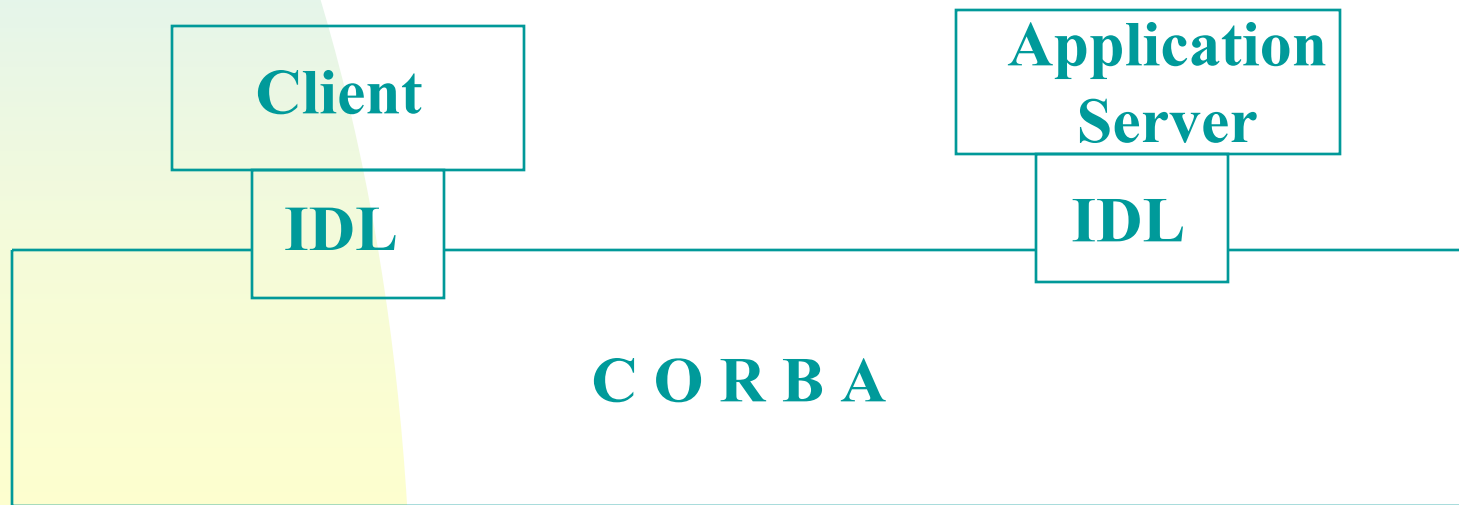
- **Java.awt** (Graphics and GUI)
- **Java.beans** (Reusable custom software components)
- **Java.io** (Input and output access to files and other streaming media)
- **Java.math**
- **Java.net** (Performing network operations)
- **Java.security**
- **Java.sql**
- **Java.text**
- **Java.util**

Issue 2 Distributing Components Across A Heterogeneous Environment

- Welcome to the CORBA/OpenDoc versus COM/OLE debate!
- Say goodbye to the hardware bus (system board)
- Say hello to the distributed object bus
- Distributed object bus solves problem of interoperability among objects
- Common term is “Middleware”

Corba Example

- ORB is middleware component that implements the CORBA bus
- It acts as a broker between clients and servers



IDL = Interface Definition Language

Component Summary

- **Components (Objects) are a major paradigm shift**
 - ◆ **They represent a new client-server revolution within a client-server revolution**
- **Objects break up the client-side and server-side of an application into smart components that work together across the network**
- **This is practical today, unlike in past years, because of increasing availability of high-speed, low cost bandwidth**

Enabling Products, Tools, and Techniques

- **Change in way we think about software**
- **Emphasize decoupling and cohesion – hallmarks of software modularity**
- **Demand reuse – require object-oriented languages with the rigor of Java (current de facto OO standard)**
- **Study and adopt a distributed object bus architecture**
 - ◆ **Corba, Com, or way to integrate the two**

Must Think “Parts”

- Carve out components from existing applications
- Use *Designer Containers*
 - ◆ Libraries of business objects that represent people and things
- Begin to assemble component suites
 - ◆ Component suites are frameworks for components (see next slide)

Component Suites

- Frameworks for Components
 - ◆ Common semantics and protocols
 - ◆ Common data structures
 - ◆ Custom events
 - ◆ Common services
 - ◆ Shared rules of engagement
- E.g., Airline Reservation Suite

Cafeteria-Style Suites

- **Pre-assembled Suites**
 - ◆ Turn-key (e.g., dental office suite)
- **Built-to-Order Suites**
 - ◆ Chose parts, places, and platform from a multi-vendor parts catalog
- **Client-Server Suites**
 - ◆ Assembled from networked parts and tested for a customer's environment

Parts Business for Components

- **Requires these services**
 - ◆ **Component distribution and tracking channels**
 - ◆ **Support**
 - ◆ **Component assembly and systems integration**
 - ◆ **ISV recruiting and relationships**
 - ◆ **Consulting, education, and tools**

Major Challenges

- Challenges:
 - ◆ Availability of standard, interchangeable parts
 - ◆ Ability to find the right parts for the job at hand
- Solutions:
 - ◆ Component technology, such as JavaBeans and ActiveX, provides a basis for interchangeable parts
 - ◆ Web provides a means for locating available components

Reality Check

- **Components are inevitable in order to build commodity software and reduce cost**
- **We are not organized for this revolution**
- **Our people do not think in object-oriented terms**
- **It will take long-term investment perspective**
 - ◆ **Resistance at senior management levels**
- **Must begin now to reap rewards in a year or two**

References

- **Middleware (Book)**
 - ◆ Daniel Serain
- **The Essential Distributed Objects Survival Guide (Book)**
 - ◆ Orfali, Harkey, and Edwards
- **Dr. Dobb's Journal, June 2000**
 - ◆ Object-Oriented Software Design

WebSites

- Carnegie Mellon University
 - ◆ <http://www.sei.cmu.edu/>
- WWW.COMPONENTSOURCE.COM
- www.itseeds.com
- www.ibm.com/developers/