From Object-Oriented Programming to Component Software

• OO Languages:
  – Ada, Smalltalk, Java, C++

• Class versus Object:
  – Express existence of objects directly in code
  – Code is more expressive, easier to develop, less costly to maintain

• Main Concepts:
  – Encapsulation – hiding of implementation details
  – Inheritance – reuse existing objects in creation of new objects
  – Polymorphism – exhibit multiple behavior depending on object used

• Reuse:
  – Code must be written in a general enough manner
  – Language-independent
Component Software

• **Object-Oriented Analysis and Design:**
  – Breakdown of a project in its logical components

• **Components:**
  – Reusable pieces of software in binary form
  – Interoperability

• **Interfaces:**
  – Semantically related set of methods
  – Strongly typed contract between software component and its clients
  – Articulation of expected behavior
  – Reusable in a variety of contexts
Evolution of COM+

COM = Component Object Model

- Clipboard 1987
- OLE 1992
- COM 1995
- Distributed Computing 1980s
- OSF DCE RPC on Windows 1992
- Distributed COM 1996
  - Microsoft Distributed Transaction Coordinator 1996
  - Microsoft Transaction Server 1997
  - Microsoft Message Queue Server 1997
- COM+ 1999

COM = Component Object Model
Problems of Complex Software

• Apps are large and complex:
  – Time consuming to develop, difficult and costly to maintain,
  – Risky to extend with additional functionality

• Monolythic style:
  – Prepackaged with a range of static features
  – Add/remove/upgrade/replace features is difficult (impossible)

• Apps do not lend themselves to integration:
  – Neither data nor functionality is available to another program

• Programming models reflect provider‘s upbringing:
  – No location-transparency

COM Software can better meet these challenges.
COM and COM+

- **COM**: Fundamental programming architecture for building software components
  - Unconfigured components
- Plus (+) an integrated suite of component services with an associated runtime environment
  - Configured components
- Support for robust server-size systems
  - Threading, concurrency, security
  - Administration, robustness
  - Example: Microsoft SQL server
Windows DNA: a COM+-based three-tier architecture
Evolution of Component Services

- Standard implementation of services that are frequently needed by component developers

<table>
<thead>
<tr>
<th>COM+</th>
<th>Distributed COM</th>
<th>Microsoft Transaction Server</th>
<th>COM</th>
</tr>
</thead>
<tbody>
<tr>
<td>load balancing, in-memory database</td>
<td>remoting architecture</td>
<td>transaction services, resource pooling</td>
<td>interface-based programming</td>
</tr>
<tr>
<td>object pooling, queued components</td>
<td>distributed component services</td>
<td>role-based security, administration</td>
<td>basic component facilities</td>
</tr>
<tr>
<td>event model, administration</td>
<td></td>
<td>just-in-time activation</td>
<td></td>
</tr>
</tbody>
</table>
Just-in-time activation

- Scalability of middle-tier components
  - Clients obtain references to context objects
  - COM+ instantiates actual business objects (transparently)
  - COM+ may de-activate objects (resource sharing)

System-created context object shadows each user object
Scalability Enhancements

Object Pooling
• COM+ may recycle objects for later reuse
  – Automatic instantiation of new objects when pools is empty
  – Useful technique when object creation is very expensive (time)

Load Balancing
• Client workload can be distributed among multiple servers in a network
  – Load balancing at component level
  – Clients contact load balancing router first
  – COM+ uses response-time analysis algorithm to determine server
  – Windows 2000 clustering service can be used to eliminate balancing router as single-point-of-failure
Queued Components

- Execute method calls against unavailable components
  - Based on Microsoft Message Queue Server (MSMQ – Windows 2000)
Transactions

• COM+ components may automatically participate in distributed transactions

• Implemented by Distributed Transaction Coordinator:
  – Object-oriented two-phase commit protocol based on COM
    (OLE Transaction specification: ITransaction, ITransactionDispenser,
    ITransactionOptions, ITransactionOutcomeEvents interfaces)
  – Support of the X/OPEN DTP XA standard (two-phase commit)
  – Originally bundled with SQL Server

• ACID properties of transactions:
  – Atomic, Consistent, Isolated, Durable

• Four levels of transaction support for components:
  – Requires/requires new/supports/does not support transactions
Security & Events

• Role-based Security:
  – Leverage Windows 2000 security model
  – Declarative and programmatic security
  – Security settings on component and interface basis

• Events:
  – Publisher/subscriber style of communication
  – External event model: publisher/subscriber do not need to execute simultaneously
  – Subscriptions are maintained outside of publisher/subscriber: persistent subscriptions
  – Subscriber is any component that implements a given class interface