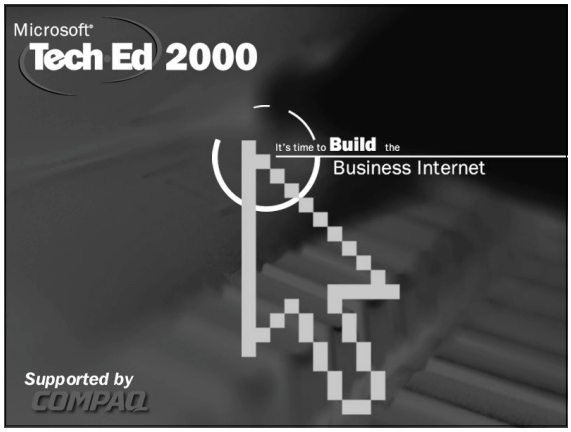
The State of Component Software

Summer 2000

COM+ Base Services

Don Box, Cofounder, DevelopMentor

developmentor™
A DEVELOPER SERVICES COMPANY



developmentor

components defined

Software components are binary units of independent production, acquisition and deployment that interact to form a functioning system

Clemens Szyperski, 1997
Component Software: Beyond Object-Oriented Programming, Addison Wesley

3

developmentor

components in nature

- Independent deployment implies dynamic linking
- Dynamic linking fairly widespread by late 1980's/early 1990's
- DLL acts as atomic unit of deployment
  - Molecules also possible…
- "Component-based" applications formed from these atoms

4

developmentor

components 1988: property bag o' code

- Early DLLs were largely raw executable code with symbol tables
  - Table of imported symbols -> DLL name
  - Table of exported symbols -> offsets

flib.dll

Export Table

Symbol	RVA
f	0x1024

Code

```
push ebp
mov esp, ebp
push [esp]
call iat[0]
```

Import Table

Library	Symbol
hlib.dll	h

5

developmentor

components 1993: code + type info

mlib.dll

Type Library

TypeName	Definition
Person	`Dog x; long y`
Dog	`float a; float b;`

Export Table

Symbol	RVA
Work(Person)	0x1024
Eat(Dog)	0x1326

Code

```
push
mov
push
call
```

6

The State of Component Software

Summer 2000

developmentor

from com to com+

- COM introduced type into loader
  - Units of code loaded based on concrete class, not filename
  - Entry point resolution/fixups performed based on typed object references, not flat symbol names
  - Calling convention expanded to encompass typed parameter lists, not numeric frame sizes

7

developmentor

from com to com+

- COM+ makes the type system of the loader extensible
  - Concrete classes are adorned with annotations that augment the type definition
  - COM+ loader examines these annotations and augments the new object accordingly
  - Very similar to mix-in style of development

8

developmentor

com+ attributes

- Concrete classes are adorned with attributes
  - Attributes indicate requirements of class code
  - Attributes indicate extended services
  - Attributes are examined at instantiation-time by the loader
  - Loader augments new object based on attributes of target class

9

developmentor

attributes as mixin

```
class MyClass : TransactionalObject,
                SynchronizedObject,
                PoolableObject,
                java.lang.Serializable
{
  void hello();
};
```

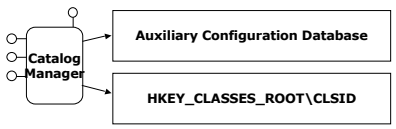
```
[
  Transaction(Required), Synchronized(Required),
  Poolable(true), Serializable(true)
]
class MyClass {
  void hello();
};
```

10

developmentor

attributes and the catalog

- Attributes are logically associated with class code by the catalog manager
  - Manages storage/caching of attribute values
  - Keeps HKEY_CLASSES_ROOT in sync
  - Exposed via scriptable, remotable COM interfaces
  - Exposed via Component Services Explorer



11

developmentor

com+ applications

- COM+ groups configured classes into applications
  - Each configured class belongs to exactly one application
  - A single DLL may have classes in multiple applications
  - Applications act as a unit of deployment/management
  - Application attributes are shared by all classes in application

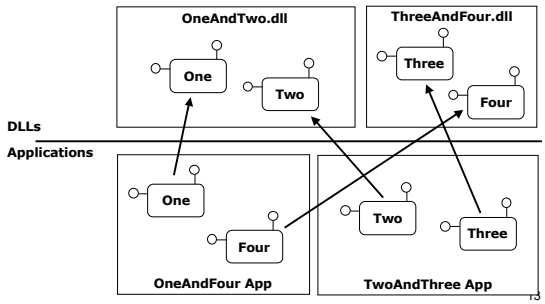
12

The State of Component Software

Summer 2000

developmentor

com+ applications



13

developmentor

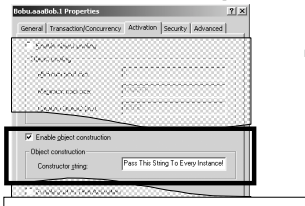
you and your attributes

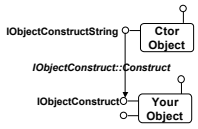
- Attributes influence object activation and method execution
  - Attributes often influence semantics of implementation code
  - Explicit programming in configured class often necessary
  - Attributes may require you to implement private interfaces
  - Attributes may augment your class with interfaces you have never seen before

14

developmentor

attribute example






```
interface IObjectConstruct : IUnknown {
  HRESULT Construct([in] IDispatch* pClassInit);
}
```

```
interface IObjectConstructString : IDispatch {
 [propget]
 HRESULT ConstructString([out, retval] BSTR* s);
}
```

15

developmentor

attribute inventory: applications

Activation Type	Library (inproc)/Server (surrogate)
Authentication Level	None, Connect, Call, Packet, Integrity, Privacy
Impersonation Level	Identify, Impersonate, Delegate
Authorization Checks	Application Only/Application + Component
Security Identity	Interactive User/Hardcoded User ID + PW
Process Shutdown	Never/N minutes after idle
Debugger	Command Line to Launch Debugger/Process
Enable Compensating Resource Managers	On/Off
Enable 3GB Support	On/Off
Queueing	Queued/Queued+Listener

Underlines indicate settings available under MTS

16

developmentor

attribute inventory: classes

Transaction	Non Supported, Supported, Required, Requires New	Class
Synchronization	Non Supported, Supported, Required, Requires New	Class
Object Pooling	On/Off, Max Instances, Min Instances, Timeout	Class
Declarative Construction	Arbitrary Class-specific String	Class
JIT Activation	On/Off	Class
Activation-time Load Balancing	On/Off	Class
Instrumentation Events	On/Off	Class
Declarative Authorization	Zero or more role names	Class Interface Method
Auto-Deactivate	On/Off	Method
Must Activate in Activator's Context	On/Off	Class

Underlines indicate settings available under MTS

17

developmentor

attributes as mixin revisited

- Attributes may need to augment instances of a class
  - May add extended properties to each instance
  - May add extended public methods to each instance
  - May require private methods of instance to work properly

```
class ColorAttribute : public IColorableObject {
private:
  virtual void OnColorChanged() = 0;
public:
  COLOR m_color;
  void colourMyWorld(COLOR c) {
    m_color = c; this->OnColorChanged();
  }
};
```

18

developmentor

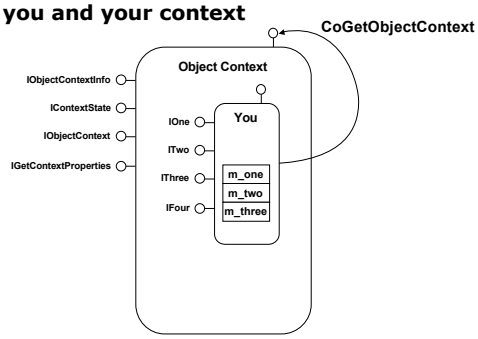
attributes and context

- COM+ envelopes every object in a context
  - Each thread has a notion of the “current” context
  - System switches context to match target object of the “current” method
  - Exposed to your method code via CoGetObjectContext

19

developmentor

you and your context



20

developmentor

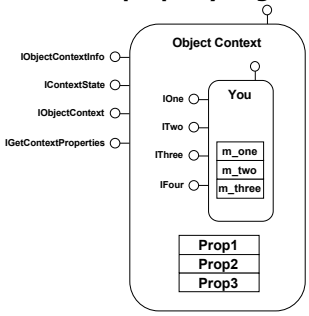
context as property bag

- A context is an ordered collection of named context properties
  - Each attribute can contribute context properties
  - Properties act as extended data members to target object
  - Properties are used by the plumbing that needs them

21

developmentor

context as property bag



22

developmentor

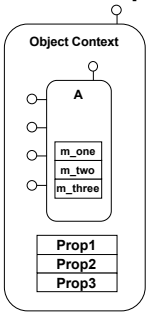
context propagation

- Context properties can propagate at instantiation time
  - The target class’s attributes can see properties of the creator
  - Attributes can vote on whether a creator’s context is OK
    - The lack of an attribute is a yes vote
  - If new context is needed, each attribute can use the creator’s properties to initialize the properties in the new context

23

developmentor

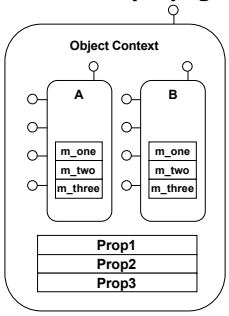
context propagation



24

developmentor

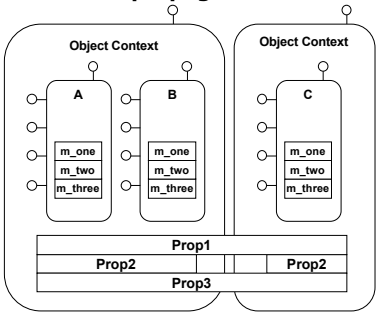
context propagation



25

developmentor

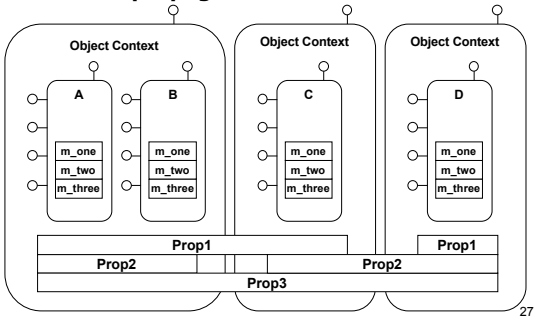
context propagation



26

developmentor

context propagation



27

developmentor

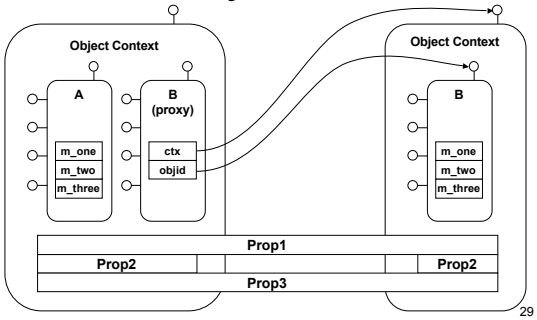
cross-context object references

- Proxies are used to allow references to appear in foreign contexts
  - Proxy implements all *remotable* interfaces of target object
  - Proxy may implement additional *local* interfaces
  - Proxy informs target context to invoke method on target object

28

developmentor

cross-context object references



29

developmentor

context relativity

- Object references are only valid in the context in which they were initialized
  - Cannot store raw object references in cross-context state
  - Proxies implicitly use system marshaling APIs to pass context-neutral cookies
  - You only need to use context-neutral cookies when bypassing method calls (e.g., global variable)

30

developmentor

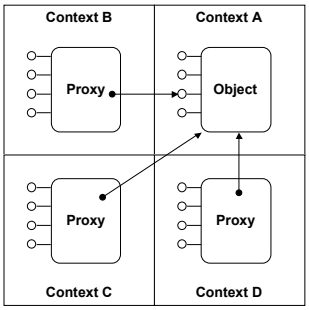
you and your context revisited

- Configured classes typically require a new context
- Non-configured classes typically live in creator's context
- All COM objects are context-bound by default
- Non-configured classes can implement IMarshal customize marshaling behavior
  - Can marshal by value by serializing into cookie
  - Can aggregate the free-threaded marshaler to become context-agile

31

developmentor

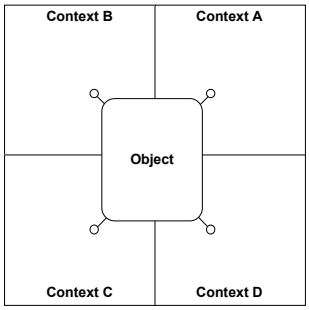
example: a context-bound object



32

developmentor

example: a context-agile object



33

developmentor

object pooling

- Classes can be configured to support object pooling
  - System maintains a per-process pool of instances
  - Instances are either in use or in inventory
  - Per-class high and low-watermarks for inventory + in use
  - Per-object decision to recycle or be destroyed

```
interface IObjectControl : IUnknown {
  HRESULT Activate();
  void    Deactivate();
  BOOL    CanBePooled();
}
```

34

developmentor

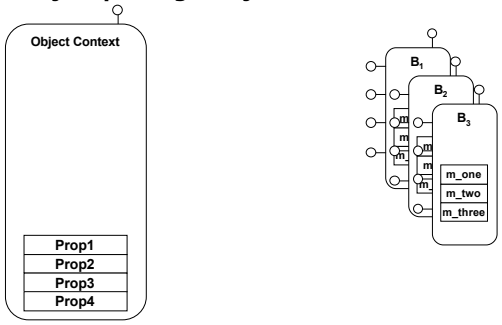
object pooling and jita

- Classes can also be configured to support just-in-time activation to further control association with context
  - JITA contexts maintain a "done" bit
  - Object is deactivated when leaving a method with done=true
  - Object will then be destroyed *unless* pooling=true
  - Next call from proxy will activate an object from inventory or create a new instance

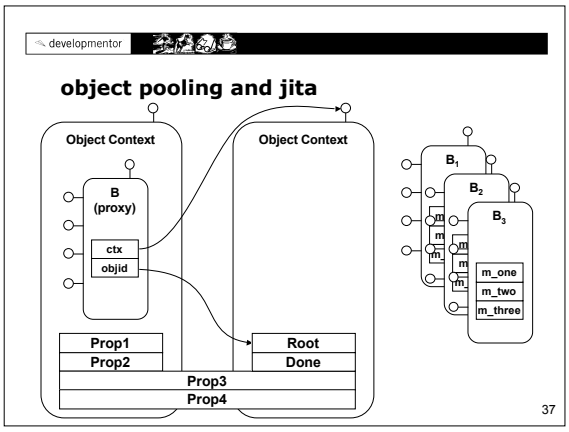
```
interface IContextState : IUnknown {
  HRESULT SetDeactivateOnReturn([in] VARIANT_BOOL bDone);
  HRESULT GetDeactivateOnReturn(
                    [out,retval] VARIANT_BOOL *pbDone);
// remaining methods elided for clarity
}
```

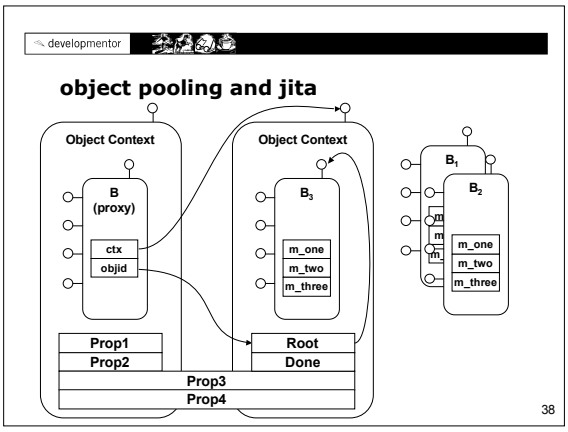
developmentor

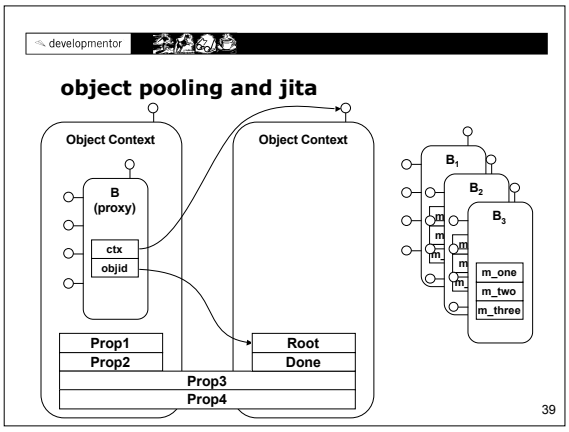
object pooling and jita

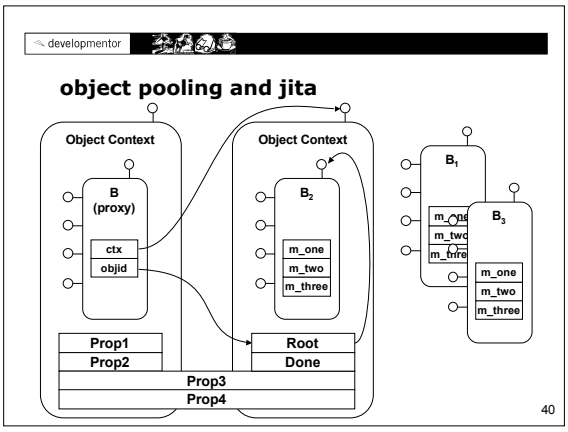


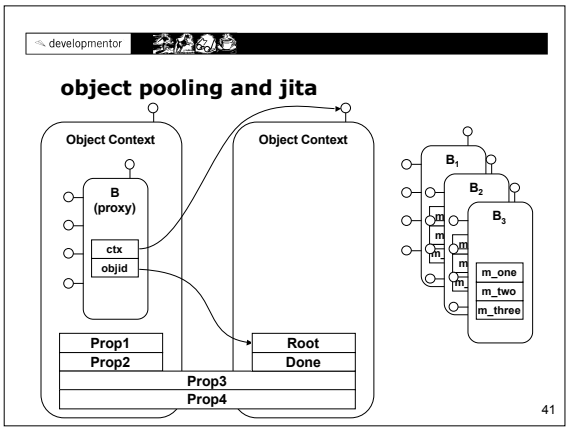
36

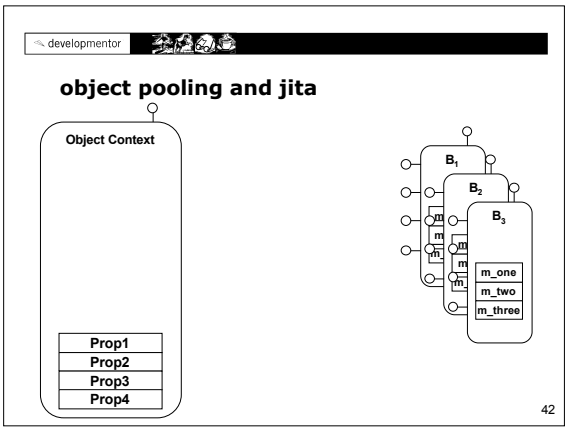












developmentor

attribute/context examples: ThreadingModel/Apartments

- COM groups objects into apartments based on the ThreadingModel attribute
  - An apartment is a group of contexts in a process that map to a particular thread (or set of threads)
  - Apartments are used to control thread affinity
  - Apartment logically acts as a context property

43

developmentor

attribute/context examples: synchronization/activities

- COM groups objects into activites based on the Synchronization attribute
  - An activity is a group of contexts across the network in which no concurrency is expected or required
  - Activities are used to serialize access to a group of contexts within a process
  - Activities are used to allocate STA-based contexts intelligently

44

developmentor

attribute/context examples: Transaction/Transaction Streams

- COM groups objects into transaction streams based on the Transaction attribute
  - A TxStream is a group of contexts across an activity whose work must be composed into an atomic transaction
  - TxStreams are used to manage concurrency at an application-level
  - Transactions are used to manage failure recovery

45

developmentor

attribute/context examples: msmq accessible types (QC)

- Configured classes can be configured to support access via MSMQ-based proxies
  - Proxy (QC Recorder) sends batched calls at deactivation
  - Proxy only supports interfaces marked *queueable*
  - Process/Application-level dispatcher handles poison message + security grunge

46

developmentor

attribute/context examples: Type-based Publish/Subscribe (LCE)

- COM's publish/subscribe system based on special type of configured class called an event class
  - Event class's catalog entry contains list of subscriber classes
  - All method calls are broadcast to subscriber instances
  - Event class's code totally synthetic – driven by TLB
  - Context can flow from publisher to event class to subscribers

47

developmentor

attribute/context examples: IIS/WAM/ASP

- IIS dispatches HTTP requests through a configured COM+ component called the WAM
  - Used to manage processes in a web server
  - Security uses context plumbing to dispatch calls under correct security principal
  - ASP adds HTTP-specific properties to context

48

developmentor

summary

- COM components are typed and self-describing
- COM object references are typed via abstract interfaces
- COM objects are typed via concrete classes
- COM classes can be annotated via declarative attributes
- Attributes use context to augment instances of classes
- Most COM+ 1.0 services exposed via attributes and context

49

developmentor™
A DEVELOPER SERVICES COMPANY