Why COM and .Net?

Jim Fawcett

CSE775 – Distributed Objects

Spring 2005

COM versus .Net

- COM Strengths
  - It's everywhere:
    - Windows operating system
    - GUI controls
    - Word, Excel, Visio
      - COM makes these programmable!
    - .Net CLR is a COM component
  - It is accessible from clients built with different languages:
    - C, C++, C#, VB, Javascript, …
  - Clients and components don't need to support the same threading models
- COM Weaknesses
  - COM is complex!
  - COM has a very weak object model
  - It is an aging technology
    - Still, heavily used by Microsoft (even in WinRT)

- .Net Strengths
  - Framework and language support is very well designed.
  - Component development is much simpler than COM.
  - Is accessible from different .Net languages.
    - Really, they are almost the same!
  - Documentation is excellent.
  - There are a lot of new books and articles (some are even good).
- .Net Weaknesses
  - Executables only run on machines equipped with the CLR.
    - Can download .Net framework, free, from microsoft for XP and Vista.
    - Won't run on Lunix, … (neither will COM)
    - Managed model is not always appropriate.

Comparison of Object Models

- ***C++ Object Model***
  - All objects share a rich memory model:
    - Static, stack, and heap
  - Rich object life-time model:
    - Static objects live for duration of the program.
    - Objects on stack live within a scope defined by { and }.
    - Objects on heap live at the designer's discretion.
  - Semantics based on a deep copy model.
    - That's the good news.
    - That's the bad news.
  - For compilation, clients carry their server's type information.
    - That's definitely bad news.
    - But it has a work-around, e.g., design to interface not implementation. Use object factories.

- ***.Net Object Model***
  - More Spartan memory model:
    - Value types are stack-based only.
    - Reference types (all user defined types and library types) live on the heap.
  - Non-deterministic life-time model:
    - All reference types are garbage collected.
    - That's the good news.
    - That's the bad news.
  - Semantics based on a shallow reference model.
  - For compilation, client's use their server's meta-data.
    - That is great news.
    - It is this property that makes .Net components so simple.

Comparison of Object Models

- ***C++ Object Model***
  - All objects share a rich memory model:
    - Static, stack, and heap
  - Rich object life-time model:
    - Static objects live for duration of the program.
    - Objects on stack live within a scope defined by { and }.
    - Objects on heap live at the designer's discretion.
  - Semantics based on a deep copy model.
    - That's the good news.
    - That's the bad news.
  - For compilation, clients carry their server's type information.
    - That's definitely bad news.
    - But it has a work-around, e.g., design to interface not implementation. Use object factories.

- ***COM Object Model***
  - Weak object model – based on C++, but:
    - No inheritance of implementation
    - No deep copies
    - No deep assignment
    - No construction with parameters
    - COM functions accept a very limited set of argument types
  - COM strengths
    - Strong support for updating systems composed of COM components – simply copy the revised dll over the original. You don't need to rebuild the client or other parts of the system
    - Supports a limited form of garbage collection based on reference counting.
  - .Net also supports updating and has full garbage collection.

Language Comparison

- Standard C++
  - Is an ANSI and ISO standard.
  - Has a standard library.
  - Universally available:
    - Windows, UNIX, MAC
  - Well known:
    - Large developer base.
    - Lots of books and articles.
  - Programming models supported:
    - Objects
    - Procedural
    - Generic via templates
  - Separation of Interface from Implementation:
    - Syntactically excellent
      - Implementation is separate from class declaration.
    - Semantically poor
      - See object model comparison.

- .Net C#
  - Is an ECMA standard, becoming an ISO standard.
  - Has defined an ECMA library.
  - Mono project porting to UNIX
  - New, but gaining a lot of popularity
    - Developer base growing quickly.
    - Lots of books and articles.
  - Programming models supported:
    - objects.
  - Separation of Interface from Implementation:
    - Syntactically poor
      - Implementation forced in class declaration.
    - Semantically excellent
      - See object model comparison.

Language Comparison

- Standard C++
  - Uses header files to declare class services
  - Program parts must be compiled using same compiler to ensure interoperability
    - Client and components must share the same threading and security models

- COM
  - Uses Interface Definition Language (IDL) to declare component services
  - IDL compiler generates:
    - Cmpnt.h – interface declarations
    - Cmpnt_i.c – defines GUIDS
    - Cmpnt_p.c – defines proxy
  - Interoperability across multiple languages and any compiler that provides support for COM
    - C, C++, Visual Basic, JavaScript, …
    - Components and client may use different threading and security models.

End of Comparison