Когда в C# не хватает C++

Сергей Балтийский
JetBrains

Why?

- Скорость
  - Оптимизация CPU
  - Затраты на переключение контекста
- Память
  - Управление памятью
  - GC
- Legacy
  - Библиотеки на C/C++

How?

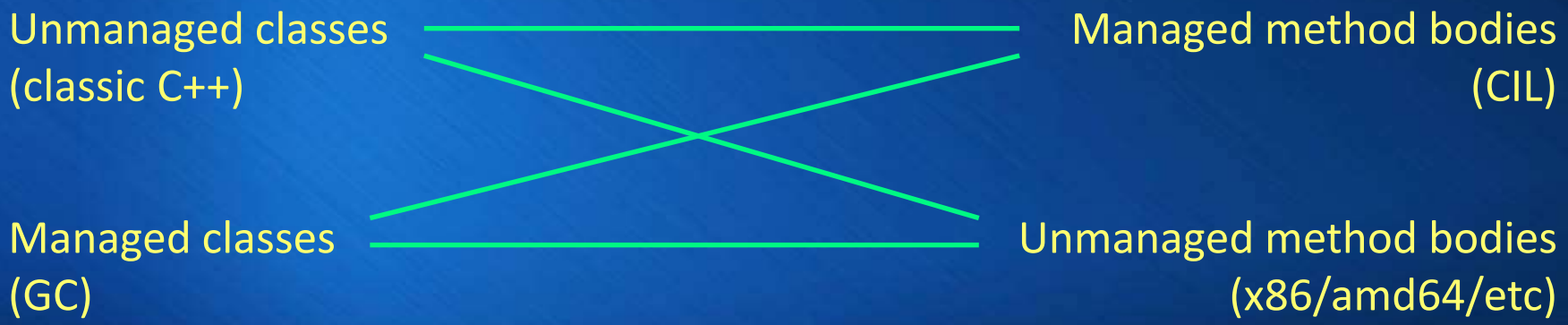
- C++/CLI
- COM
- PInvoke
- Native Memory in C#

C++ CLI

- C++ on Common Language Infrastructure
- Наследник MC++
- Взаимопроникающая интеграция C++ и C#

C++ CLI

- Взаимопроникающая интеграция C++ и C#



C++ CLI — pros

- Это интуитивно понятный выбор
  - Язык, специально созданный для стыковки C++ и .NET

C++ CLI — pros

- Вся мощь C++ optimizing compiler *
  - Оптимизация на промежуточном представлении, до выгрузки в IL (по аналогии с компиляцией в x86)
  - Заметный результат на подходящих задачах
    - lexing, parsing

C++ CLI — pros

- Нетривиальные типы данных
  - Передача между C++ и C# без маршаллинга

C++ CLI — cons

- Проблемный компилятор
  - Большое время компиляции
  - Internal compiler errors
  - Постоянные доработки и изменения
    - Разный набор ошибок компиляции и рантайма в каждой версии тулсета, v90 / v100 / v110 / v120

C++ CLI — cons

- Проблемный компилятор
  - Логические ошибки в поддержке языка
    - пример: использование структуры, объявленной в C# внутри generic структуры
    - templates and generics

C++ CLI — cons

- Проблемный оптимизатор
  - Из-за логических проблем с каждой новой версией компилятора отключают классы оптимизаций
    - v120 медленнее v100
  - Нет Profile-Guided Optimization
  - Не работает полноценный link-time optimization
    - Разница заметна с unity builds

C++ CLI — cons

- Проблемное окружение
  - Привязка к конкретной версии .NET Framework при использовании конкретной версии компилятора
    - v90 -> v3.5 (CLR2)
    - v100 -> v4.0 (CLR4)
    - v120 -> v4.5 (CLR4)

C++ CLI — cons

- Проблемное окружение
  - Привязка к Windows NT
  - CoreCLR is coming
  - Linux, MacOS на горизонте
  - Перспективы портирования C++ CLI туманны
    - -> /clr:pure -> /clr:safe

C++ CLI — cons

- Проблемное окружение
  - Разные архитектуты процессора даже внутри Windows NT
  - Файл содержит native code и не является AnyCPU
  - Множественная компиляция под разные CPU
  - Плохо сочетается с C# внутри одного SLN

C++ CLI — cons

- Проблемное окружение
  - CRT dynamic linking only
  - Невозможно получить самодостаточный статически слинкованный файл
  - CRT Redist
    - Для некоторых версий — только через SxS (v90)
    - Не поместить 32/64 в один каталог

C++ CLI — cons

- Неведомое и непознанное
  - При загрузке в secondary appdomain происходит утечка исполнения в primary appdomain
    - Пример сценария: nUnit unit tests загружает рабочий код в специальный аппдомен
  - В primary appdomain могут быть недоступны assembly references
    - Может отсутствовать appbase / private bin path / binding redirects / assembly resolver, который помогает находить файлы в родном appdomain
  - -> невозможность работы в таком сценарии

C++ CLI — cons

- Неведомое и непознанное: appdomain leak workarounds
  - Не использовать статические переменные с managed handles
  - Не использовать CRT (после v90)
    - И что угодно ещё в новых версиях компилятора

C++ CLI — cons

- Провоцирует писать код в managed режиме в духе C++

C++ CLI

COM

- Component Object Model
- Набор конвенций для вызовов между рантаймами разной природы (“ABI”)
  - Функции
  - Структуры данных
  - Объекты
  - Время жизни объектов
  - Exceptions

COM

- First-class support in .NET
  - .NET — идеологический потомок COM/COM+/DCOM
  - Windows Runtime — альтернативная ветвь развития
  - .NET CLR **не** основан на COM внутри
  - Но в нём встроена фундаментальная поддержка COM
    - .NET ← COM
    - COM ← .NET

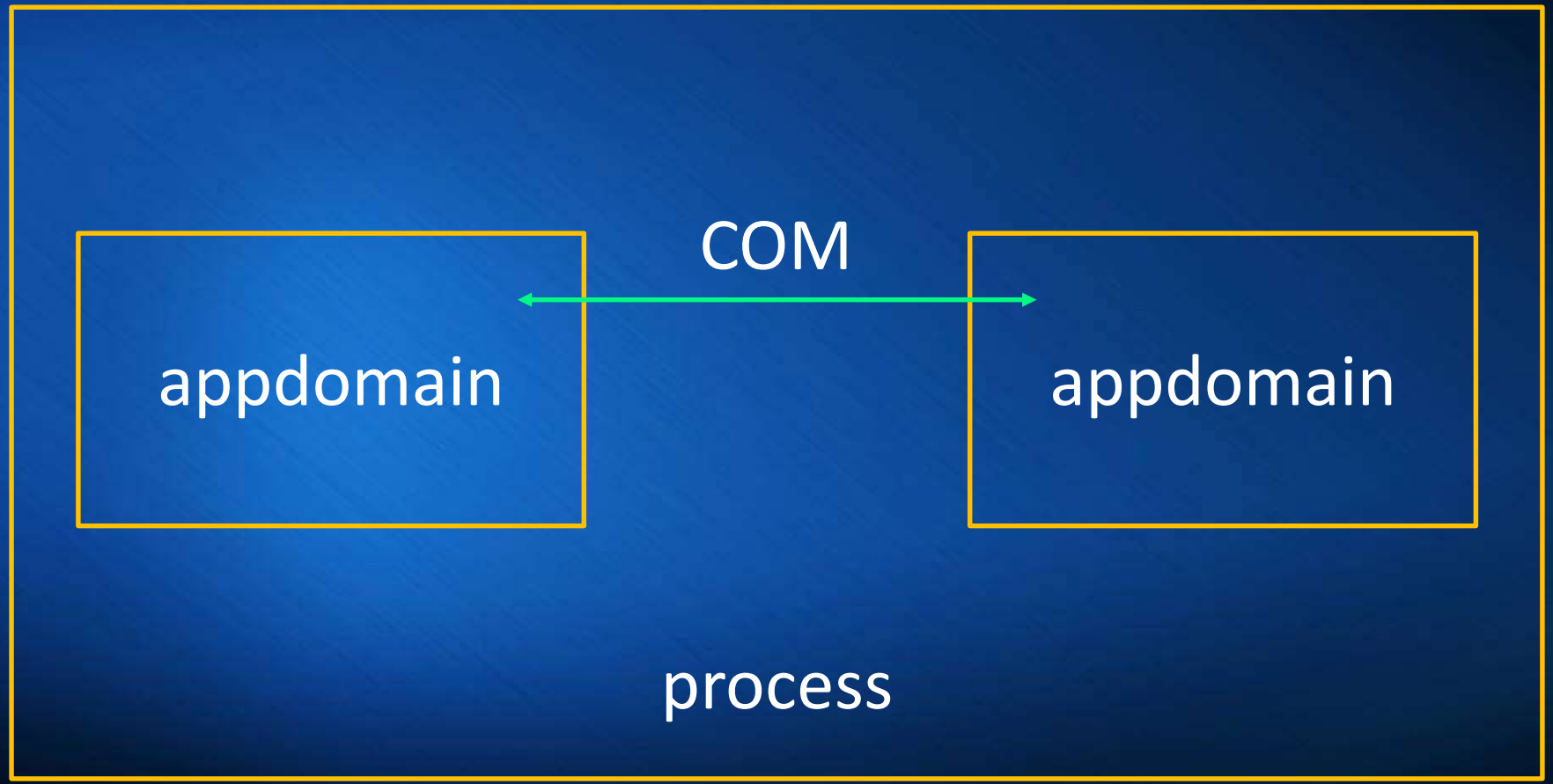
COM — pros

- Выглядит почти как managed object
- Легко импортируется в C#
- Интегрируется в GC
- Managed/native код живёт отдельно
- Можно линковать статически

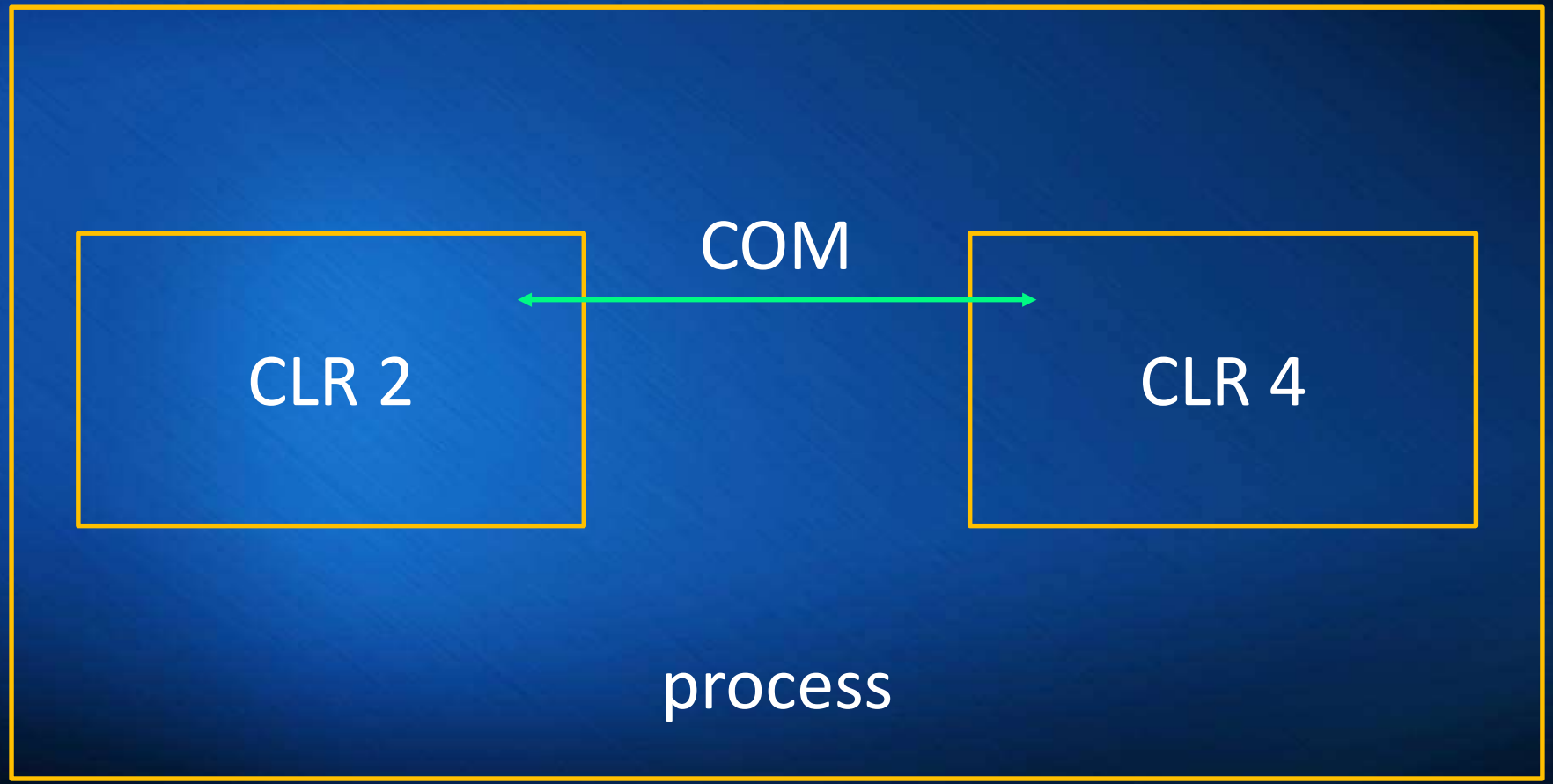
COM — cons

- Другая threading model
- Спецприём для создания самого первого объекта для xcopy deployment
- Утечки памяти при неосторожном обращении
- Portability*

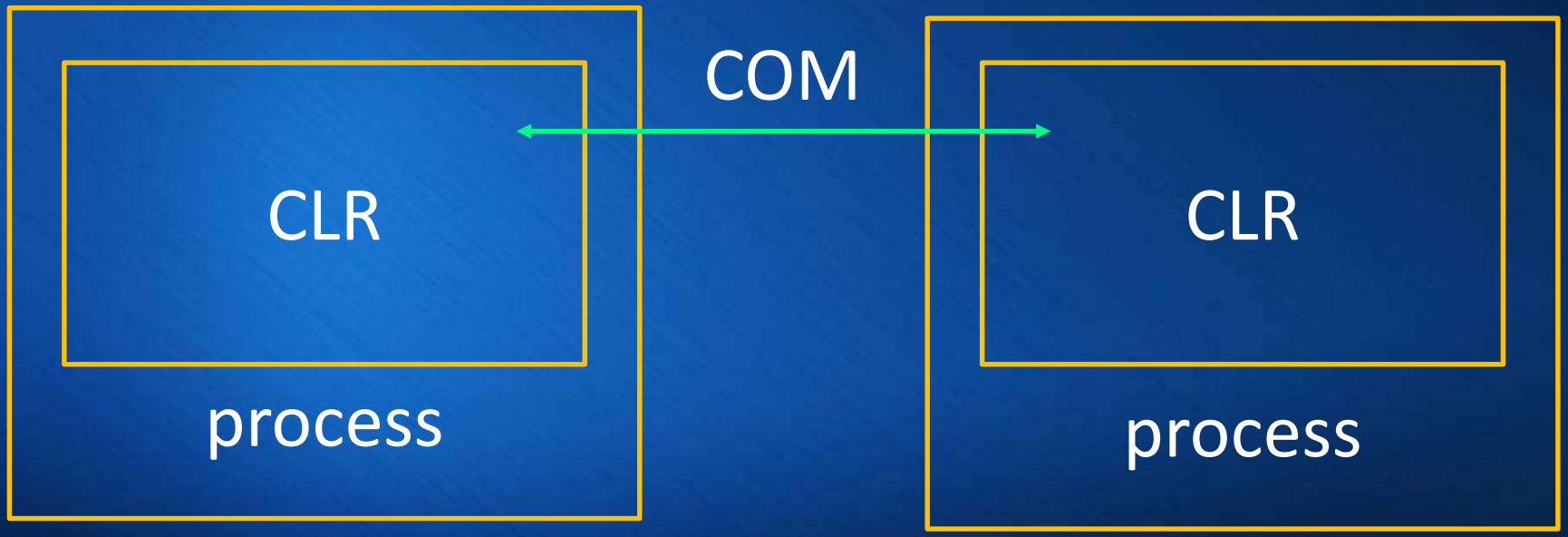
COM within .NET: appdomains



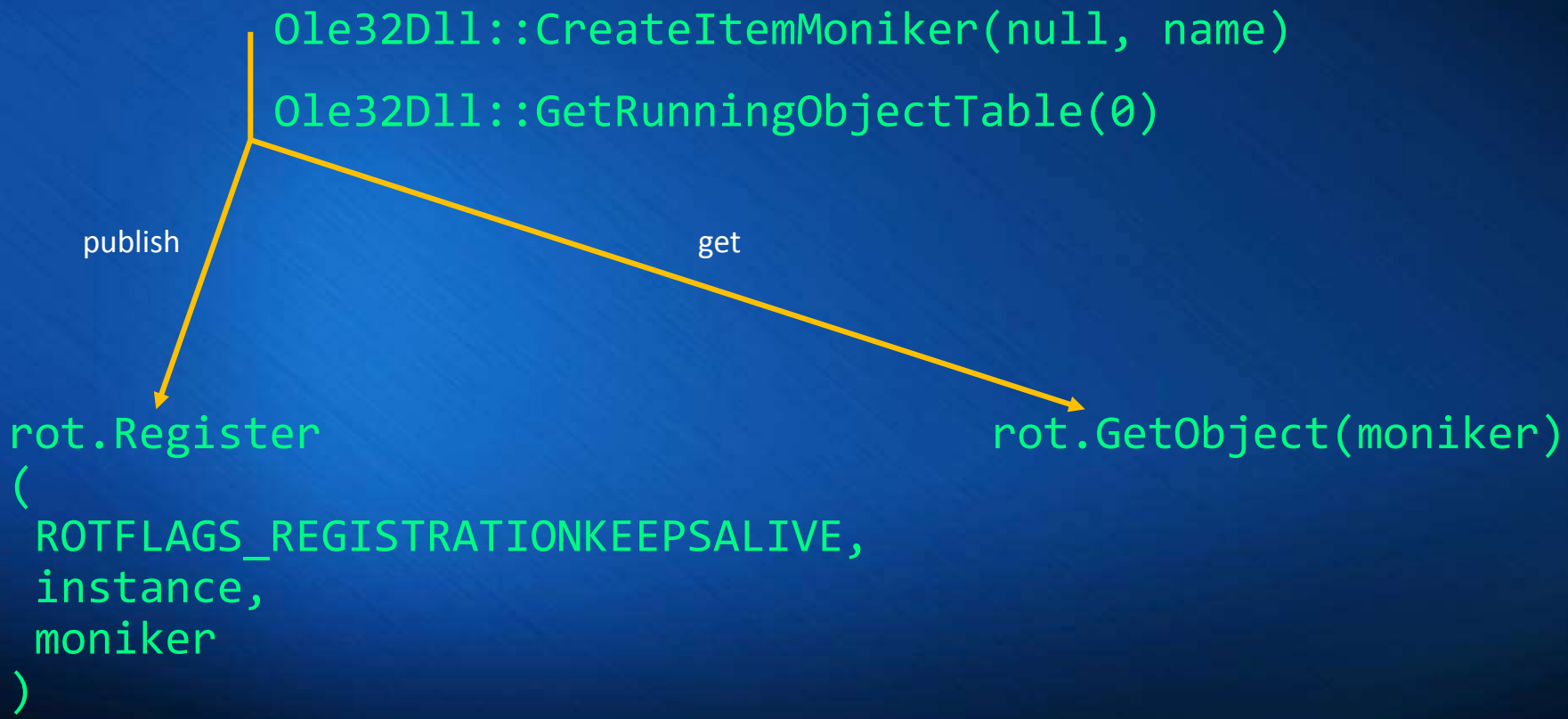
COM within .NET: runtimes



COM within .NET: processes



COM within .NET: Interprocess Call



IDispatch vs Reflection

- Late-bound calls по имени функции
- .NET ← COM
  - Если есть `IDispatch`
  - Reflection, `dynamic` keyword
- COM ← .NET
  - `ComInterfaceType: InterfaceIsIDispatch, InterfaceIsDual`

COM in .NET

- RCW: Runtime Callable Wrapper
  - Автоматическое преобразование в COM-вызовах
  - Вручную через `Marshal::GetObjectForIUnknown`

COM Object Lifetime

- Reference counting: AddRef, Release
  - +1 при создании объекта
  - При достижении 0 объект сам себя уничтожает
- Правила обращения со счётчиком ссылок
  - При получении в in-параметре: не трогать
  - При записи в field: +1, потом не забыть -1
  - При отдаче в out-параметр: +1
  - При получении из out-параметра

COM AddRef/Release in C++

- При получении в in-параметре: ничего не трогать

```
HRESULT DoWork(IUnknown *pUnk)
{
    // просто используем pUnk внутри функции
}
```

COM AddRef/Release in C++

- При длительном хранении: +1, -1

```
HRESULT DoWork(IUnknown *pUnk)
{
    // запись в member variable
    pUnk->AddRef();
    m_pUnk = pUnk;
    // не забыть сделать Release() в деструкторе
    // или при стирании
}
```

COM AddRef/Release in C++

- При возврате (out-параметр, return): +1 (-1 должен сделать получатель)

```
HRESULT DoWork(IUnknown **ppUnk)
{
    // отдаём свой объект
    *ppUnk = m_pUnk;
    m_pUnk->AddRef();
}
```

COM AddRef/Release in C++

- При получении возвращаемого значения: -1

```
IUnknown *pUnk = NULL;
DoWork(&pUnk);

// тут пользуемся pUnk

pUnk->Release();
```

Reference Counting & GC

- RCW держит одну ссылку на COM Object
  - Одна на все интерфейсы одного объекта
  - Делает AddRef при первом знакомстве
  - Делает Release в своём finalizer (или вручную)

Reference Counting & GC

- У RCW есть ещё один reference counter
  - Отражает количество попаданий COM объекта в CLR
  - Уменьшается при `Marshal::ReleaseComObject`
  - При достижении 0 отпускает COM объект
    - и становится недоступным для использования из дотнета

Marshal::ReleaseComObject

- Позволяет вовремя отпустить COM объект
  - Можно этим не заниматься
  - Тогда объект отпустится *когда-нибудь*, по GC
- Для этого и нужен второй reference counter
  - Отражает представление пользователя о том, сколько раз объект был независимо получен из внешнего мира
- Опасно вызвать слишком много раз
  - “COM object that has been separated from its underlying RCW cannot be used.”

Reference Counting & GC

- GC не боится циклов
- Но если цикл проходит через COM, то опять боится

.NET in COM

- CCW: COM Callable Wrapper
  - Автоматическое преобразование в COM-вызовах
  - Автоматически при создании .NET-COM-объекта из нативного кода
  - Вручную через `Marshal::GetIUnknownForObject`

.NET in COM

- Атрибуты на managed типах
  - ComVisibleAttribute
  - GuidAttribute
  - ComInterfaceTypeAttribute

Reference Counting & GC

- CCW не даст GC собрать объект
  - Это вам не delegate в PInvoke

.NET → COM → .NET

- Из цепочки RCW-CCW извлекается настоящий .NET объект

Importing Declarations

- COM → .NET
  - .TLB → .DLL/C#
  - TLBIMP.EXE
  - “Add COM Reference” in Visual Studio
  - Написание интерфейса вручную по MIDL/MSDN/.h
  - dynamic / Reflection
- .NET → COM
  - .DLL → .TLB (→ `#import`)
  - TLBEXP.EXE
  - Case collisions
    - нельзя иметь COM-visible типы, отличающиеся только регистром

Primary Interop Assembly

- Эталонный авторский TLBIMP
- С native COM objects можно иметь несколько альтернативных деклараций одновременно
  - Совместимость интерфейсов только по GUID
  - Совместимость функций только по порядковому номеру
- Managed COM objects требуют PIA
  - Цепочки RCW-CCW сворачиваются, виден настоящий CLR объект
  - Совместимость интерфейсов/методов по правилам CLR
  - Послабления в CLR4 (for Visual Studio)

COM & Exceptions

- Error codes вместо exceptions
- HRESULT (32-bit integer)
  - `0` → OK
  - `<0` → Error
  - `>0` → Success with details
- `IErrorInfo`
- .NET умеет преобразовывать HRESULT ↔ exception
  - optionally
  - С потерей success code

PreserveSig

- MIDL:
  - ```
    HRESULT GetObject
    (
      [in, unique] IMoniker *pmkObjectName,
      [out] IUnknown **ppunkObject
    );
    ```
- C# PreserveSig
  - ```
    [MethodImpl(MethodImplOptions.PreserveSig)]
    Int32 GetObject(IMoniker pmkObjectName,
    [MarshalAs(UnmanagedType.Interface)] out object ppunkObject);
    ```
- C# non-preserve-sig
  - ```
    [return:MarshalAs(UnmanagedType.Interface)] object
    GetObject(IMoniker pmkObjectName)
    ```
    - Можно не проверять вручную, была ли ошибка
    - Невозможно отличить S_OK от S_FALSE
    ```

.NET Threading

- `ApartmentState::STA`
- `ApartmentState::MTA`

COM Apartments

- Условное понятие

- Группа объектов, которые могут вызывать друг друга напрямую
- Группа тредов, в которой объекты вызывают друг друга напрямую

- COM Thread принадлежит одному Apartment
- COM Object принадлежит одному Apartment
  - Функции объекта будут вызываться только с его тредов*

Single-Threaded Apartment

- Single thread
- Cooperative multitasking
- Windows message pump
  - Обязательно прокачивать сообщения на таком треде
  - Вызовы из других тредов приходят как windows messages
    - Синхронно либо асинхронно
- Reentrancy

Multi-Threaded Apartment

- Максимум один в процессе
- Содержит все MTA-треды
- Использует thread pool для входящих вызовов

Threads & Apartments

- STA
- STA0
- MTA
- n/a

COM Object Threading Models

- n/a
- Single
- Apartment
- Both
- Free
- (Free-Threaded Marshaller)
  - (+ neutral)
  - Влияет на создание объекта системными средствами

Creating a COM Object

	STA0	STA	MTA
n/a	•	→ STA0	→ STA0
Single	•	→ STA0	→ STA0
Apartment	•	•	→ STA
Both	•	•	•
Free	→ MTA	→ MTA	•

COM Object Pointer & Apartments

- ISomeInterface*
- В пределах одного apartment можно использовать напрямую
- При передаче в другой apartment:
  - ISomeInterface* → byte stream → другой ISomeInterface*
  - Новый указатель будет неявно исполнять код в другом треде (если надо)
- .NET делает это сам внутри RCW

RCW & Apartments

- Неявный блокирующий вызов в домашний apartment
- STA
  - Домашний STA объекта должен быть доступен
  - Message pump
  - Особенно при финализации RCW

COM Registration

- Classic: Windows Registry
  - Per-machine or per-user
  - Не дружит с XCOPY
- SxS
  - “Registration-Free COM Interop” @MSDN
- Classic WinAPI
  - `__declspec(dllexport) extern “C”` функция
- Ручной вызов стандартных функций
  - `DllGetClassObject`

THE END