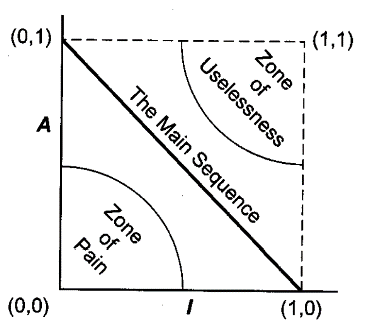
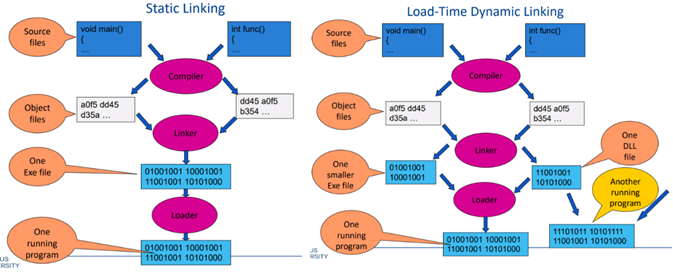
# 1: Redegør for ideerne bag komponentbaseret programudvikling, og tilhørende designprincipper.

* Hvad er en komponent?
* Hvorfor bruge komponenter?
* Lav komponenter selv / hente dem udefra
  + Fordele og ulemper ved begge
* Hvordan laver man en god komponent? (designprincipper)
* Hvad er et godt design?
  + Ingen design smells
* Agil
* SOLID
  + ISP (Interface Segregation Principle)
  + DIP (Dependency Inversion Principle)
* Granularity
  + REP (Release equivalency Principle)
    - Forfatteren skal vedligeholde komponenten
  + CRP (Common Reuse Principle)
    - Klasserne genbruges sammen
  + CCP (Common Closure Principle)
    - Klasserne opdateres sammen (Single responsibility)
* Stability
  + ADP (Acyclic Dependency Principle)
    - Ingen cykler
  + SDP (Stable Dependency Principle)
    - Afhæng kun af mere stabile komponenter
  + SAP (Stable-Abstraction Principle)
    - Skal være lige så abstrakt som stabil
* AI graph

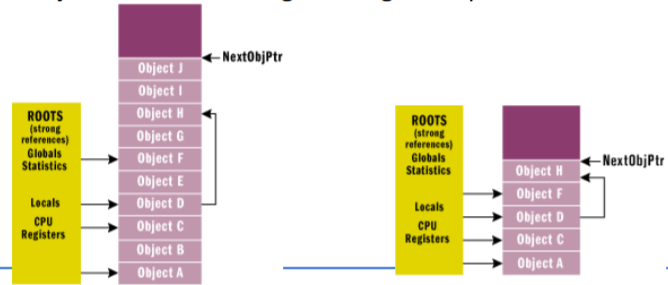
# 2: Redegør for ideerne bag komponentbaseret programudvikling og redegør for udvikling og brug af DLLfiler i C++ samt vis og forklar hvorledes man kan udvikle og anvende plugins i C++.

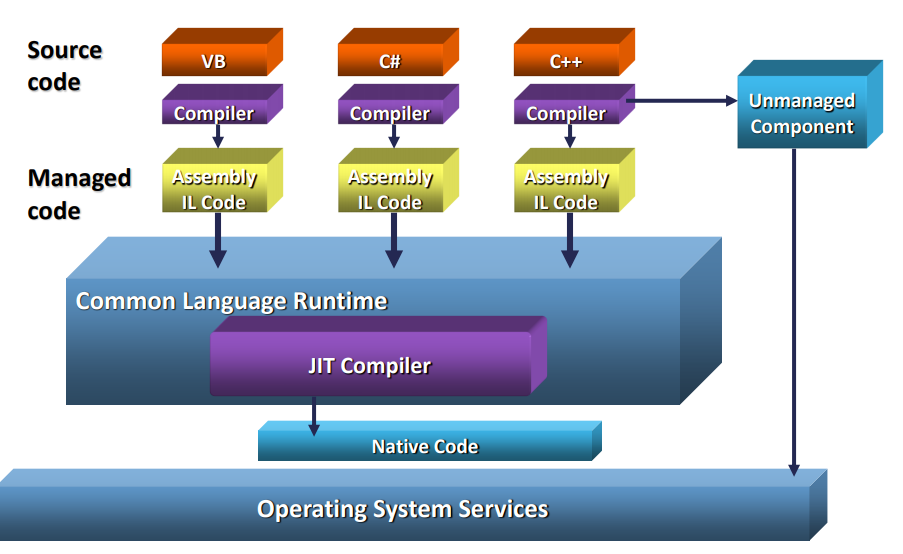
* Hvad er en komponent?
* Hvorfor bruge komponenter?
* Lav komponenter selv / hente dem udefra
  + Fordele og ulemper ved begge
* Hvad er en DLL?
  + Static/Dynamic linking
* Loadtime/Runtime Dynamic linking(Implicit/explicit)
  + Plugins
* Name mangling
  + Extern “C”



# 3: Redegør for .Nets komponentmodel og Lifecycle Management. Samt vis og forklar hvorledes man kan udvikle og anvende komponenter i C#.

* Hvad er .net?
* Assembly
  + Manifest
    - Strong name
  + Metadata
  + CIL/native Code
  + Resources’
* GAC
  + Shared assemblies med Strong name
  + Fordele/ulemper
* Garbage collector
  + Mark and compact
    - Root (memory location)
  + NextObjPtr
  + Generations
  + Drawbacks
    - Subscription
  + Weak refs



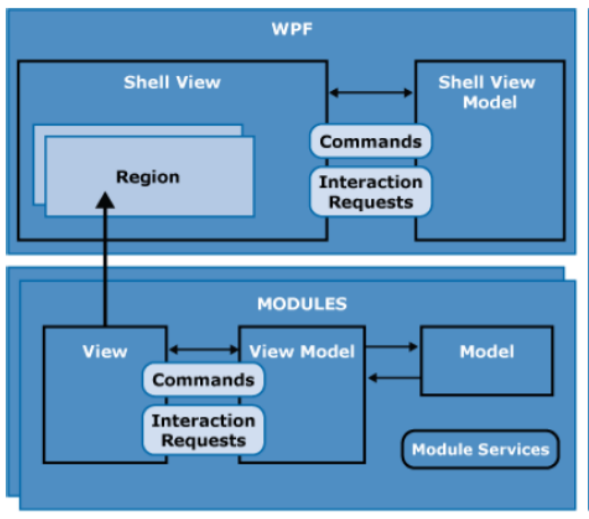


# 4: Redegør for begrebet dependency injection og brugen af IoC-containere. Giv et overblik over Microsofts forskellige dependency injection og/eller Extensibility Frameworks. Redegør for de grundlæggende begreber i MEF og vis et eksempel på brug af MEF eller Unity.

* Dependency injection
  + Constructor injection
  + Property/setter injection
  + Fordele
    - Testability
  + Ulemper
    - Kompliceret injection
* IoC container(inversion of control)
  + Hvad er det? Hvorfor bruge det?
  + Unity
    - Cross platform
    - Ulemper
      * Kræver eksplisit type (RegisterType og Resolve)
  + MEF(Managed Extensibility Framework)
    - Export/Import
    - Catalog
      * Aggregate
      * Directory
      * Assembly
    - Lazy loading

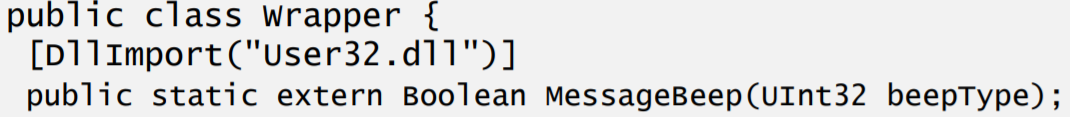
# 5: Redegør for den grundlæggende arkitektur og begreber i Prism. Hvilke fordele og ulemper er der ved at bruge Prism? Vis et eksempel på brug af Prism.

* Hvad er Prism?
* Hvornår bruges Prism?
* Key concepts
  + Shell
  + Regions
  + Module
  + MVVM
  + Navigation
    - State-based
    - View-switching
  + Controller
    - Styrer din view-switching
  + Services
    - EventAggregator
    - RegionManager
  + Commands
    - DelegateCommand
    - CompositeCommand
  + BootStrapper
    - Dependency Injection
* Arkitektur

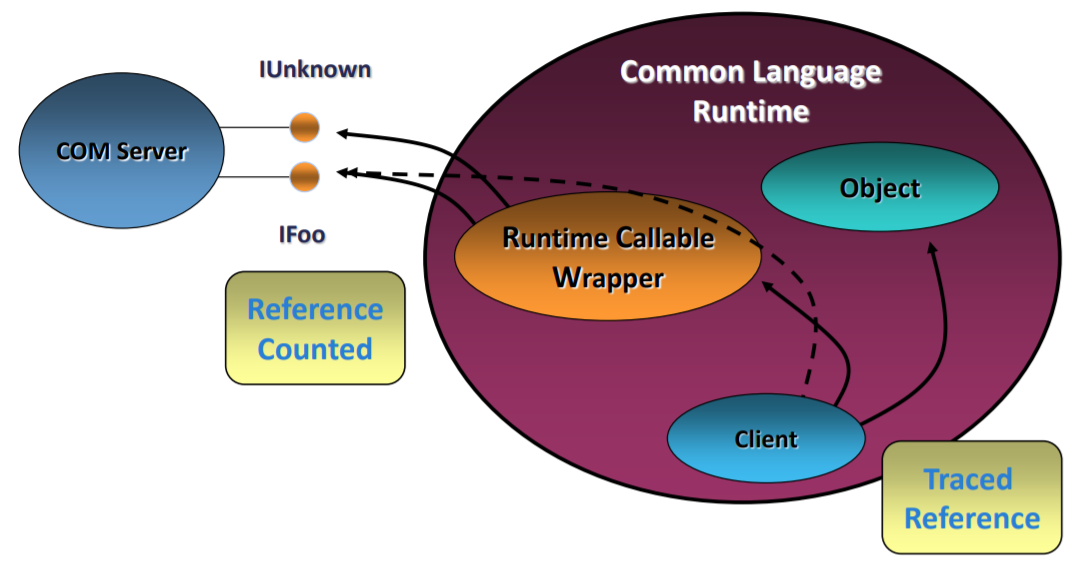


# 6: Redegør for begrebet "Interoperability" generelt, og redegør for brugen af PInvoke samt interoperability mellem COM og .Net. Vis eksempler på interoperability.

* Interoperability
  + Unmanaged/managed
  + PInvoke



* + - EntryPoint
    - CharSet(Auto som default)
    - SetLastError
    - CallingConvension
      * Cdecl(Caller)
      * Stdcall(Callee)
  + Marshalling
    - Strings
    - Pointer
  + Considerations
    - Overhead
* .NET til COM interoperability
  + Kan bruge COM komponenter i .NET
  + Bruges ofte ved død kode
  + Runtime Callable Wrapper
    - Reference counted (IUnknown)
      * AddRef
      * Release
      * QueryInterface
    - Garbage Collector eller Finalize(Caller)
    - COM Callable Wrapper
  + Vis project 8\_2



# 7: Redegør for problemer og muligheder for cross platform development på .Net platformen. Vis et eksempel på en cross platform component. Redegør for hvorledes man designer og implementerer Windows RT komponenter.

**Problem**

* **Forskellig kode for forskellige platforme**

**.Net Core**

* **Cross platform**
* **Fælles base class library**
* **Bliver delivered som NuGet pakker**

**.Net Standard**

* **Specifikation af API’er som implementeres af .Net Framework, .Net Core, Xamarin**
* **.Net Standard Lib: lag som mapper til Base Class Libs af platforme**
* **Erstatning for PCL da den kun kører mod specifik version**
* **.Net Standard løser: Nemt at tilføje platforme! (skal blot overholde *API version)***

**Windows RT**

* **Windows Store apps**
* **Fået en del af generelle .NET funktioner**
* **COM komponenter med .NET typesystem**
* **Operativsystem**
* **Objektorienteret**
* **Kører i sandbox**
* **Komponentmodel: COM**
* **Projections**
  + **CLR og Framework**
* **CCW and RCW**
* **Ingen System.Object – men ligner pga. projection**
* **WinRT komp: Kaldbar fra Javascript, C/C++, C#, CB**
  + **Public typer skal være WinRT**

**Dynamic Language Runtime**

* **Bedre dynamic language compability**
* **Easy code**
* **Langsommere end statisk. Hurtigere end reflection**
* **Lettere COM interop**

# 8: Giv et overblik over de forskellige muligheder for cross platform udvikling for Android og iOS enheder. Redegør for Xamarins arkitektur og virkemåde for de forskellige understøttede platforme. Samt redegør for både fordele og udfordringer ved brug af Xamarin til App udvikling. Vis et eksempel på brug af Xamarin.

Cross Platform muligheder

* Native
  + Mest hardware funtionalitet
  + Bedst performance
* Non-Native - Browser
  + Begrænset af browerseren
* Hybrid – En skal med browser
  + Middelpunkt, adgang til hardware og benytter webviews.

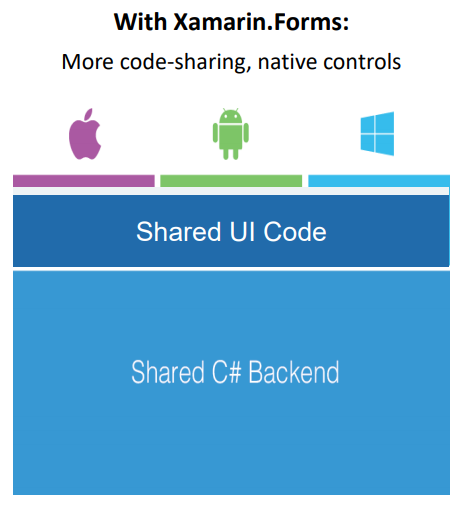
Xamarin arkitektur

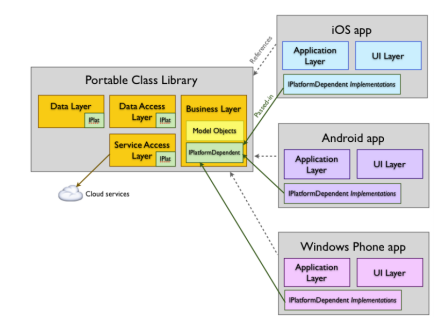
* Fælles kodebase
* PCL (Portable Class Library) (Obsolete) – SCL (Standard Class Library)
* SAP (Shared Asset Project)

Xamarin Native

* Native UI

Xamarin forms

* Fælles UI I XAML – Device.OnPlatform<T>(T,T,T)



Kode afvikling

* IOS
  + AOT
    - Performance
* Android
  + JIT
  + MONO Runtime – Mono Callable Wrapper (MCW)
  + ART – Android Callable Wrapper (ACW)

# 9: Redegør for begrebet ”Services”, og redegør for hvorledes man designer og implementerer en Windowsservice ved brug af .Net og C#. Vis et eksempel på en windowsservice.

* Hvad er en service?
* Typer af services
  + Remote service (web)
  + Local service (windows service)
  + Komponent service(Service i Prism)
* Windows service
  + Egen Windows Session
  + SCM – Service Control Manager
  + Implementering af windows service
    - Override onStart & onStop
    - Funktioner køres i tråde
* Debug
  + Installering + start
  + Event log
* Hvordan tilgås services?
  + Service controller
    - getServices()
    - Custom commands (int 128-255)
  + Filer

# 10: Redegør for brugen af komponenter ved client side webudvikling. Hvad skal browseren understøtte for at man kan lave UIkomponenter, som er uafhængige af et JavaScript framework. Vis et eksempel på en web komponent, som er udviklet uden brug af framework eller evt. ved brug af LitElement.

* Uafhængig af framework
  + HTML element
  + Navngivning
  + customElement.define()
* Attributes/properties
* connectedCallback
* DOM og Shadow DOM
* ES module (javascript module)
  + Markup i JS fil (template.js)
    - Render()
* Document fragment
* LitElements
  + Fast and light-weighted
  + Gzipped and minified

