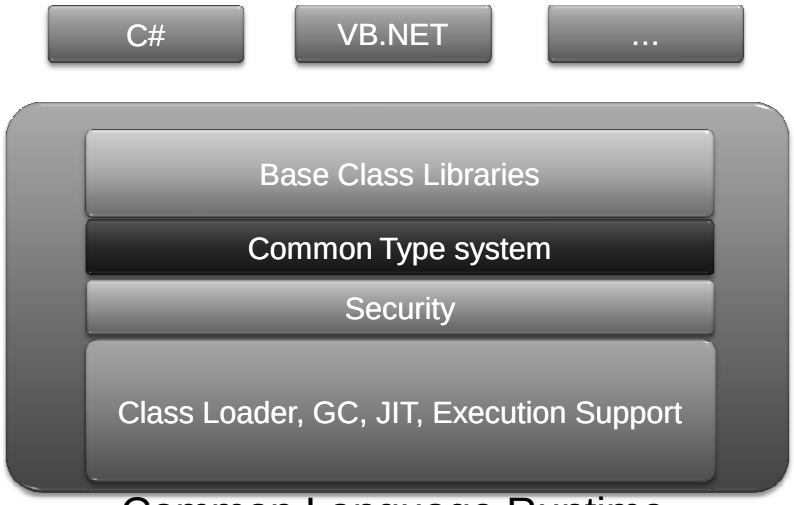
.NET Programming Language Pragmatics

Joel Pobar
Language Geek
http://callvirt.net/blog

Agenda

- The .NET Language Ecosystem
- Language Design Spectrums
- A Tour of Language Features
- Interoperability
- Driving the Future

.NET Language Ecosystem
CLR Refresher



.NET Language Ecosystem
Standards based

- ECMA/ISO Standardized
  - CLR + Parts of the BCL + C# language
  - Up to the 4th edition
- Common Type System
  - Contract for types and interoperability
  - Rules for wide reach
  - Object-oriented in flavour
  - Function + Dynamic possible

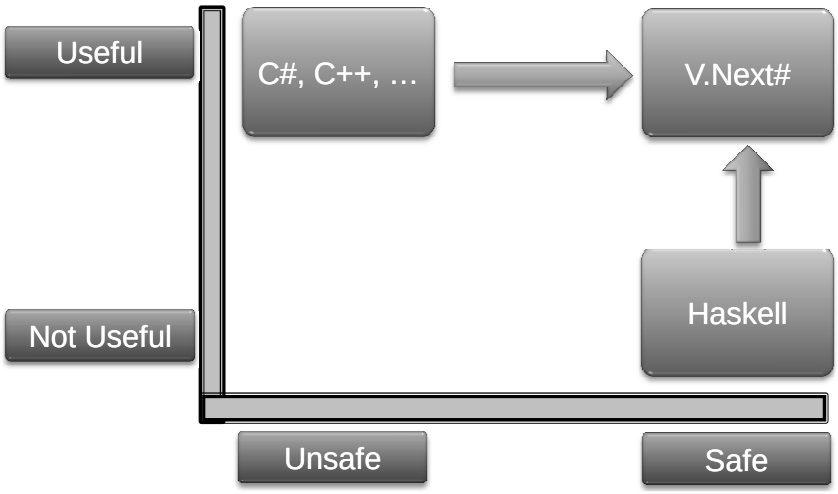
.NET Language Ecosystem
Multi-language

- Version 1.x (2000-03)
  - Focused on statically typed OO languages (C#, VB.NET, Eiffel, C++)
- Version 2.0 (2005)
  - Better support for Function languages (Generics, faster and relaxed delegates)
  - Enabled support for Dynamic Languages (LCG, better delegate support)
- Version 3.x (2008)
  - Dynamic languages (DLR)
  - Flexible type systems

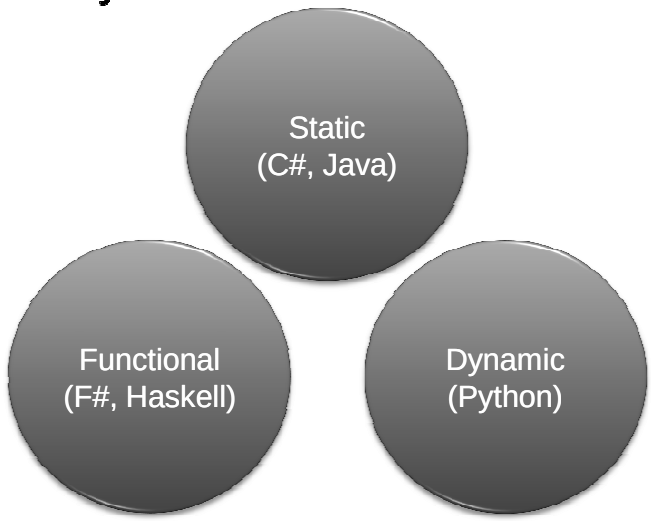
Language Spectrums
Design Pressures

- Productivity
  - Line for line: developer productivity
- Application paradigm shifts
  - “Cloud” programming
  - Web/Rich Client/User Experience
  - Unstructured data
- Hardware changes
  - Concurrency and Parallelism
- Developer happiness

Language Spectrums
Safe vs. Useful



Language Spectrums
Static to Dynamic



Language Feature Tour
Static OO Languages

- Statically Typed
  - Type system is strictly enforced at compile time
  - Typically strongly typed (adds runtime checks)
- Encapsulation and Reuse
  - Visibility
  - Polymorphism
  - Contracts
- Problems & Misconceptions
  - Type safety == safe program
  - Complicated program types
  - Contracts must be known at compile time
  - Can be verbose

Language Feature Tour
Functional Languages

- Generally more “safe”
  - Less (or no) side effects
- Mathematical roots
- Interesting Language Features
  - Higher-order & first-class functions
  - Lazy evaluation
  - Concurrent
- Problems & Misconceptions
  - Poor Adoption
  - Hard – requires mindset shift

F#

demo

Joel Pobar
Languages Geek

Language Feature Tour
Dynamic Languages

- Flexible type systems
  - Dynamically typed
  - Indispensable for apps with dynamic behavior
- Rapid application development
  - REPL (Read Eval Print Loop)
  - Focus on domain problem, not code
- Adapts well to unstructured data
  - Great for the “cloud”
- Great extensibility experience
  - Allow scripting of your apps

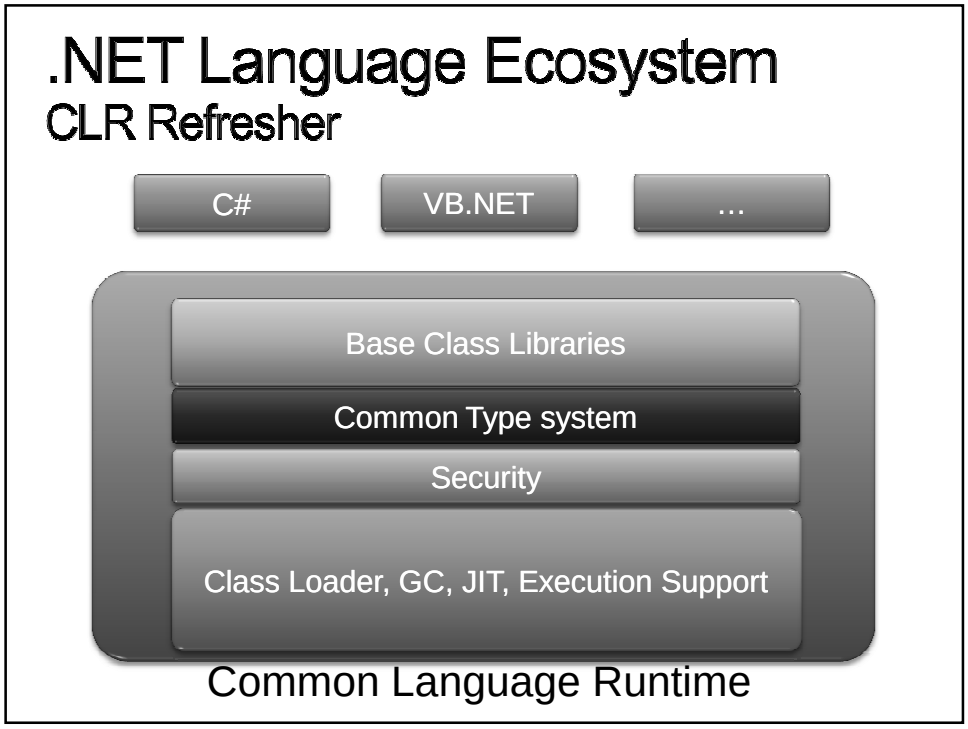
Language Feature Tour
Dynamic Languages

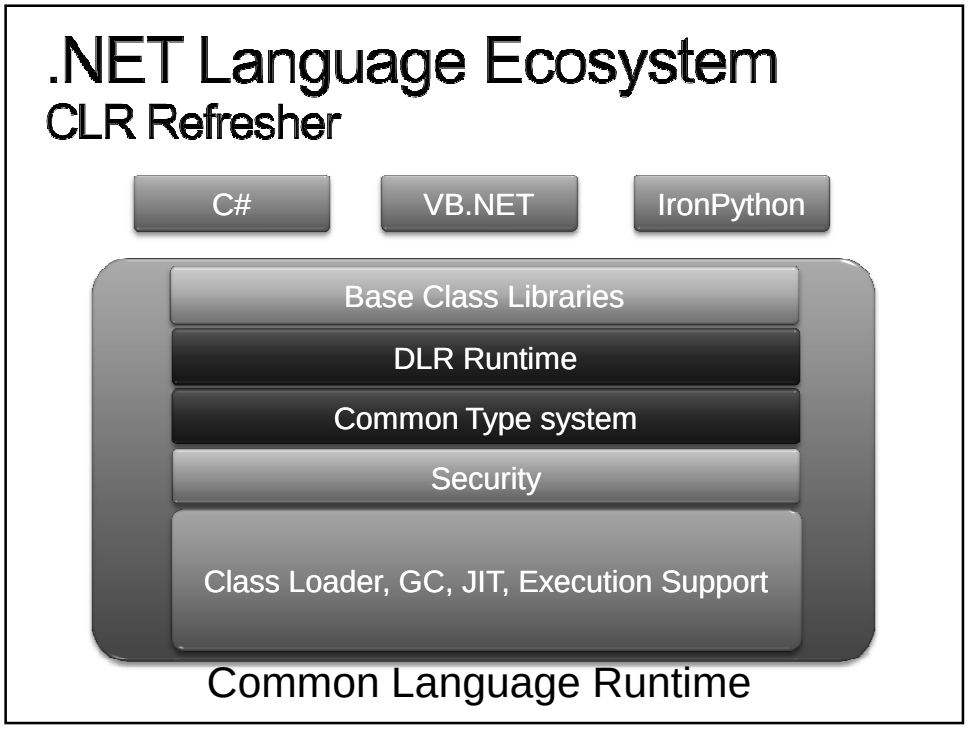
- Interesting language features
  - Functions as objects
  - Dynamically scoped variables & implicit arguments
  - Ad-hoc relationships (mixin's)
  - Duck typing
- Problems & Misconceptions
  - Delays type checking to runtime
  - eval() is a poor substitute
  - Jury is out on "less bugs" argument

IronPython
IronRuby

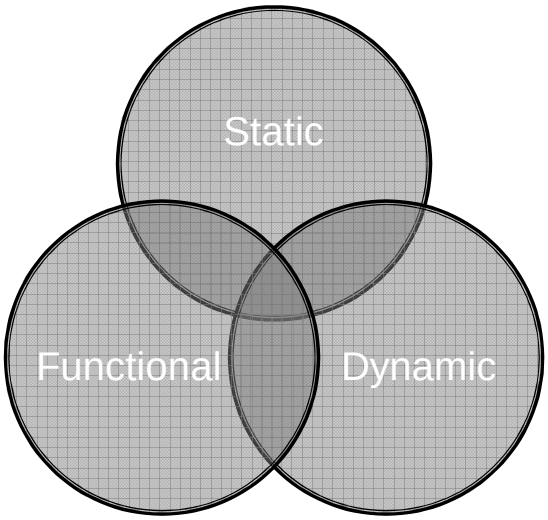
demo

Joel Pobar
Languages Geek

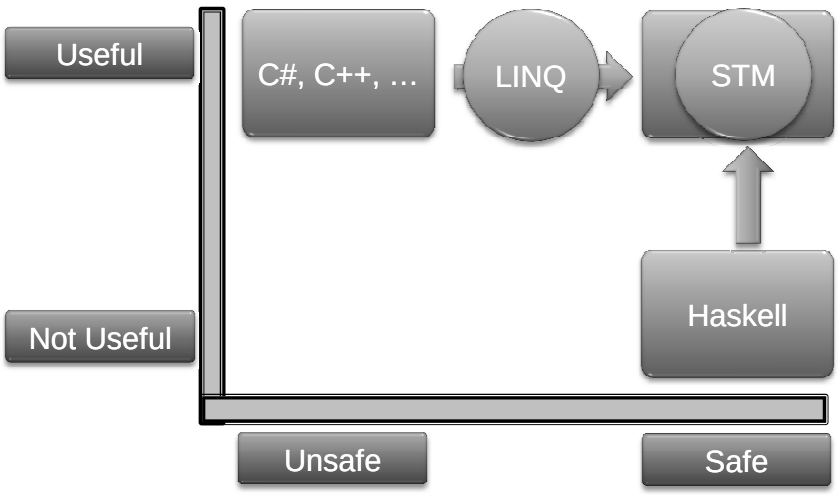




Language Spectrums
Static to Dynamic: Hybrid languages



Language Spectrums
Safe vs. Useful



Driving the Future
VB.NET 9 and C# 3.0

- Leverages features from both Dynamic and Functional languages
  - The Erik Meijer mantra: “Static typing where possible, dynamic typing when needed”
  - “Looks a lot like”: integrating dynamic features in to a static language like C#
  - Great for three-tier
- STM (Software Transactional Memory)
- LINQ: Don’t consolidate, comprehend!

Resources

- Silverlight: http://www.silverlight.net
- F#: http://research.microsoft.com/fsharp/fsharp.aspx
- IronPython: http://www.codeplex.com/Wiki/View.aspx?ProjectName=IronPython
- IronRuby: http://www.ironruby.net
- MSDN Article: http://msdn.microsoft.com/en-us/magazine/cc507636.aspx

Visual Basic

demo

Joel Pobar
Languages Geek