Sistemi Distribuiti
Corso di Laurea in Ingegneria

Prof. Paolo Nesi

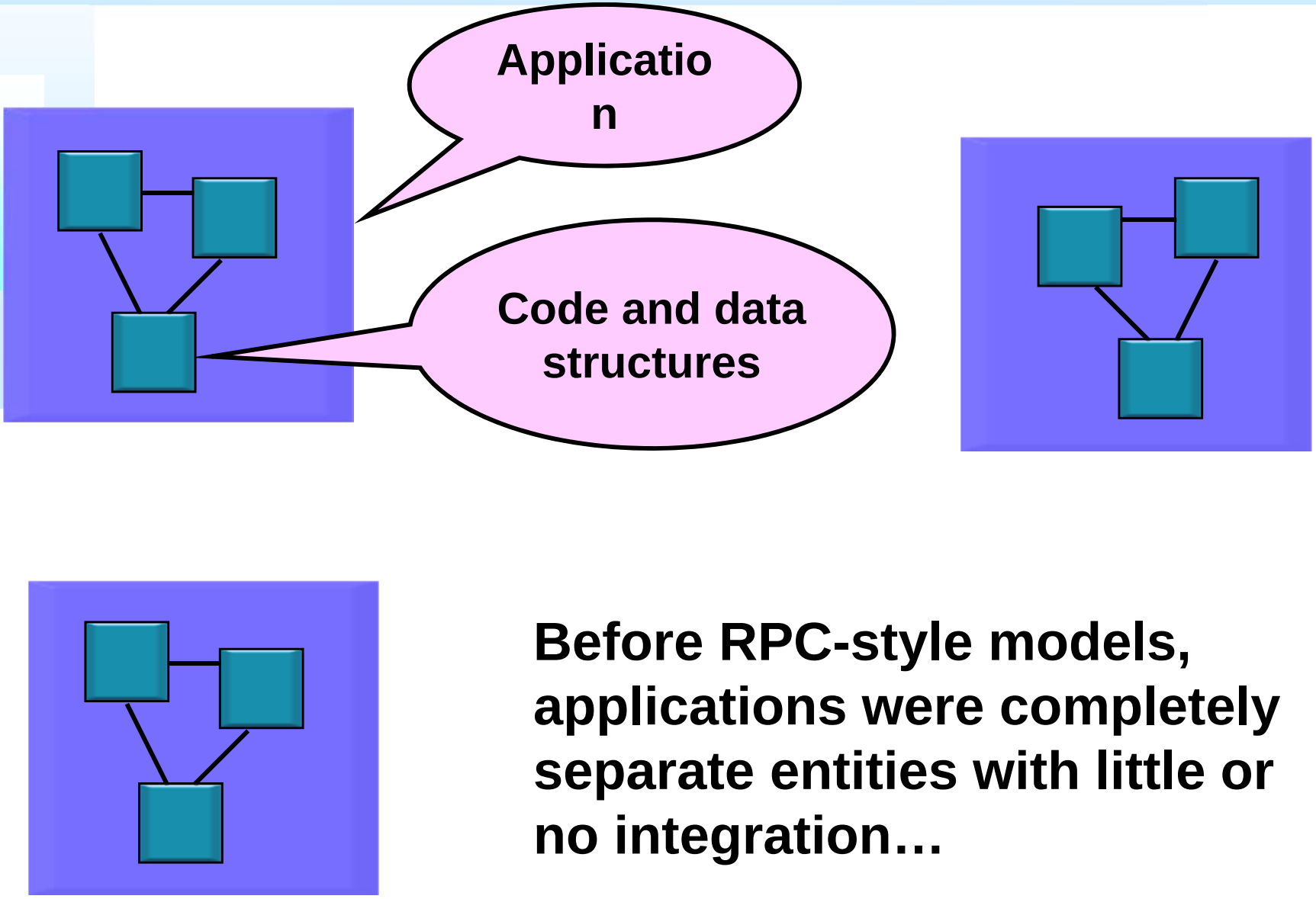
PARTE 8: **.net Framework**

Department of Systems and Informatics
University of Florence
Via S. Marta 3, 50139, Firenze, Italy
tel: +39-055-4796523, fax: +39-055-4796363

Lab: DISIT, Sistemi Distribuiti e Tecnologie Internet

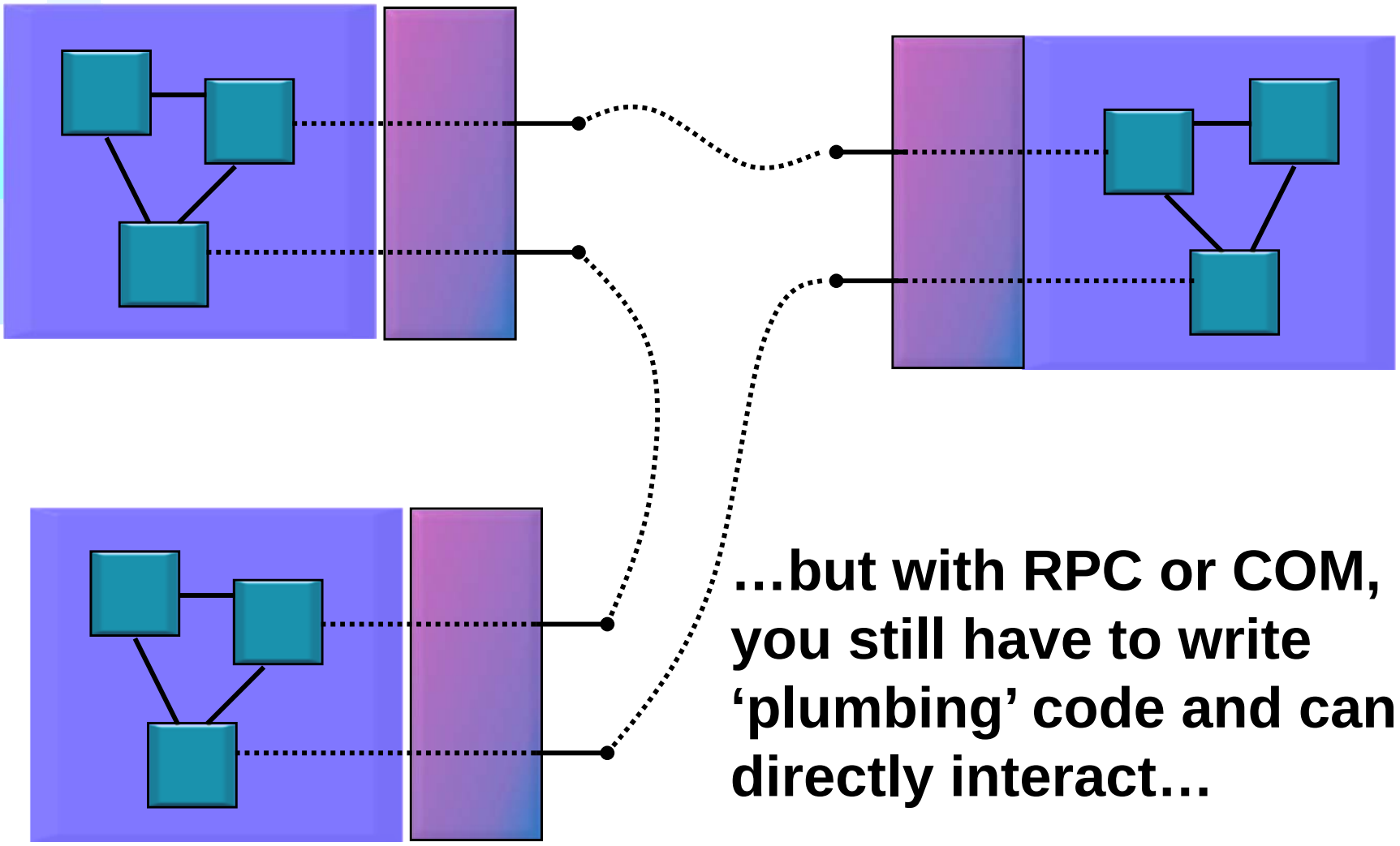
nesi@dsi.unifi.it nesi@computer.org
www: http://www.dsi.unifi.it/~nesi

A problem of communication



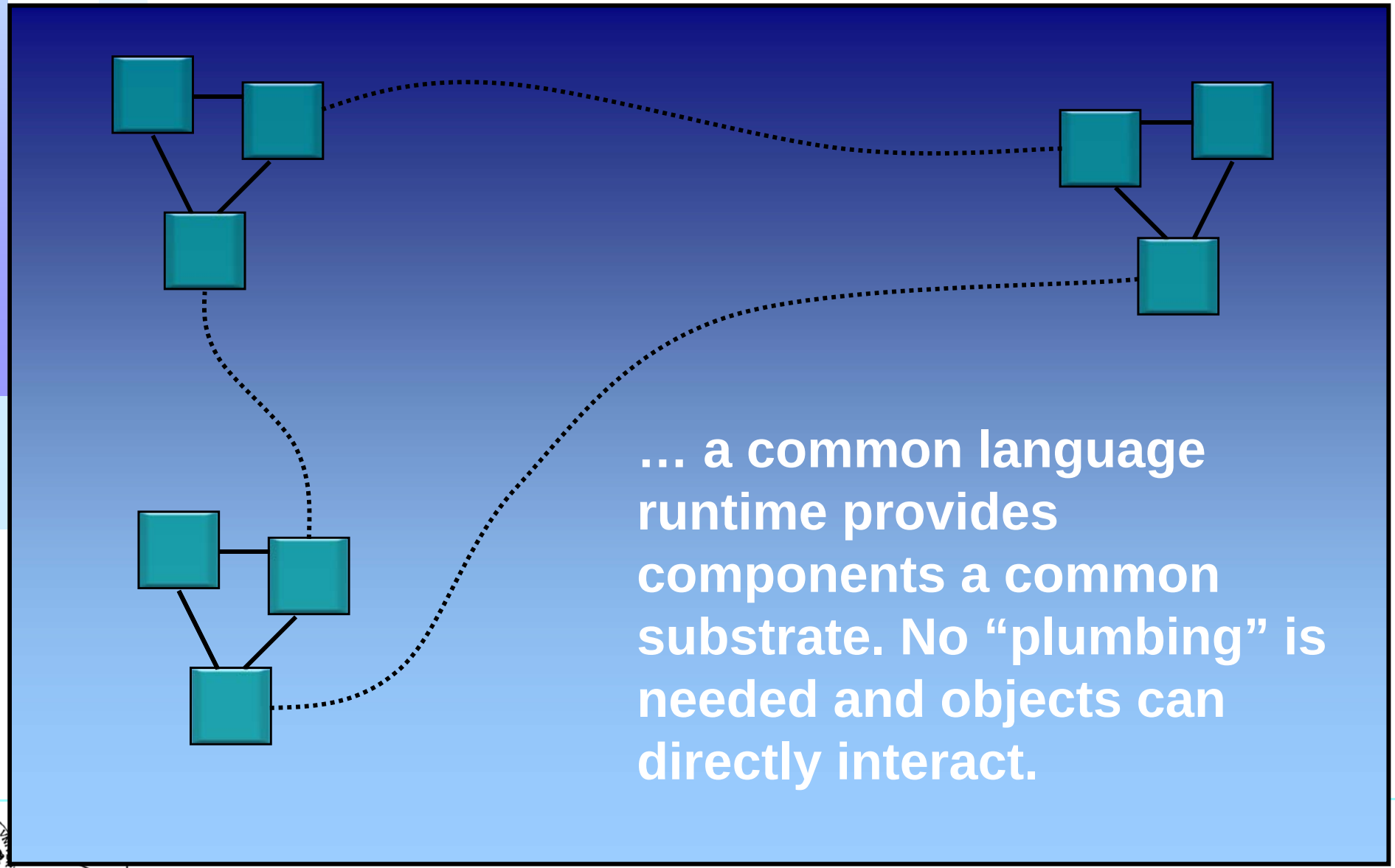
Before RPC-style models, applications were completely separate entities with little or no integration…

A problem of communication

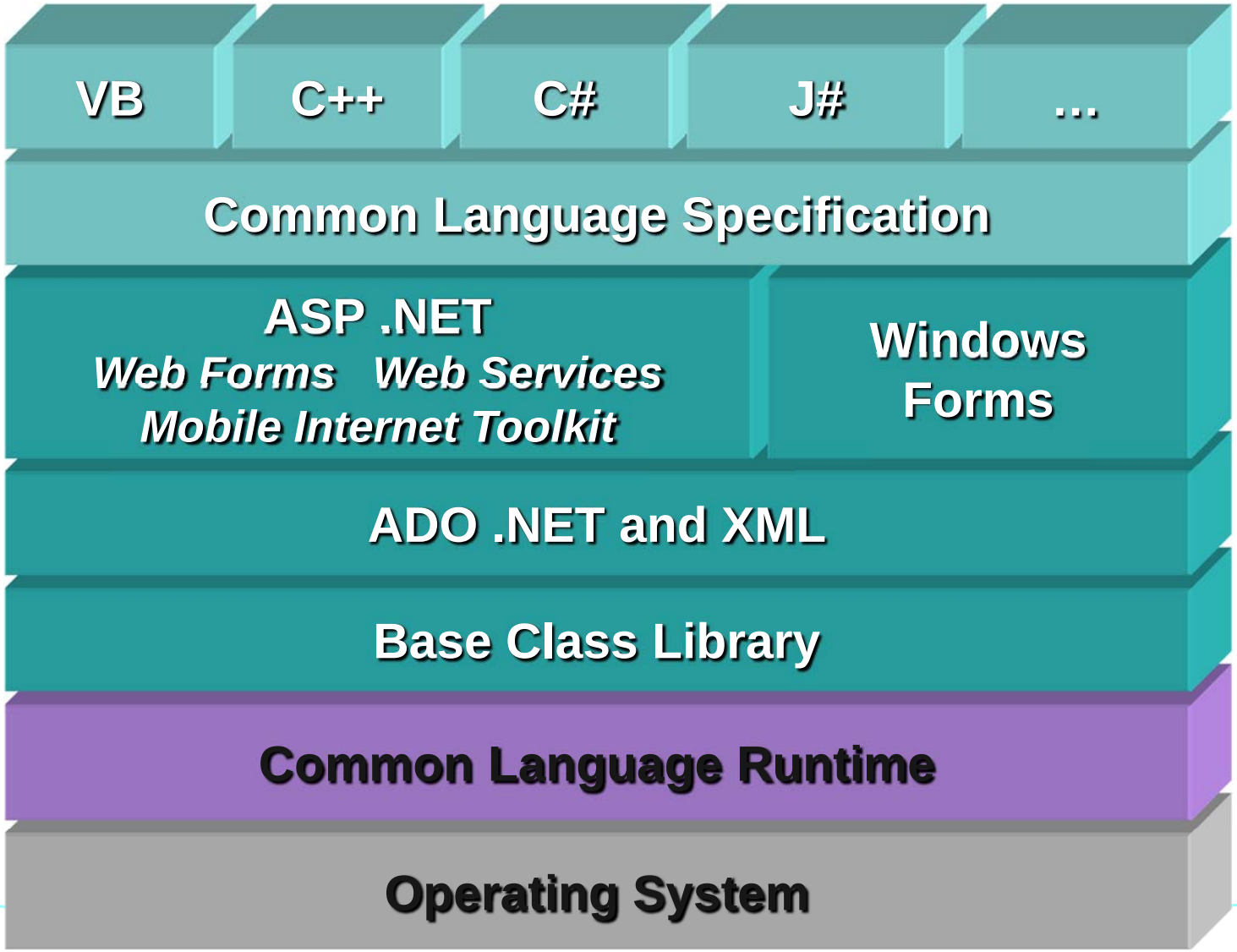


…but with RPC or COM, you still have to write ‘plumbing’ code and can’t directly interact…

The .NET Approach



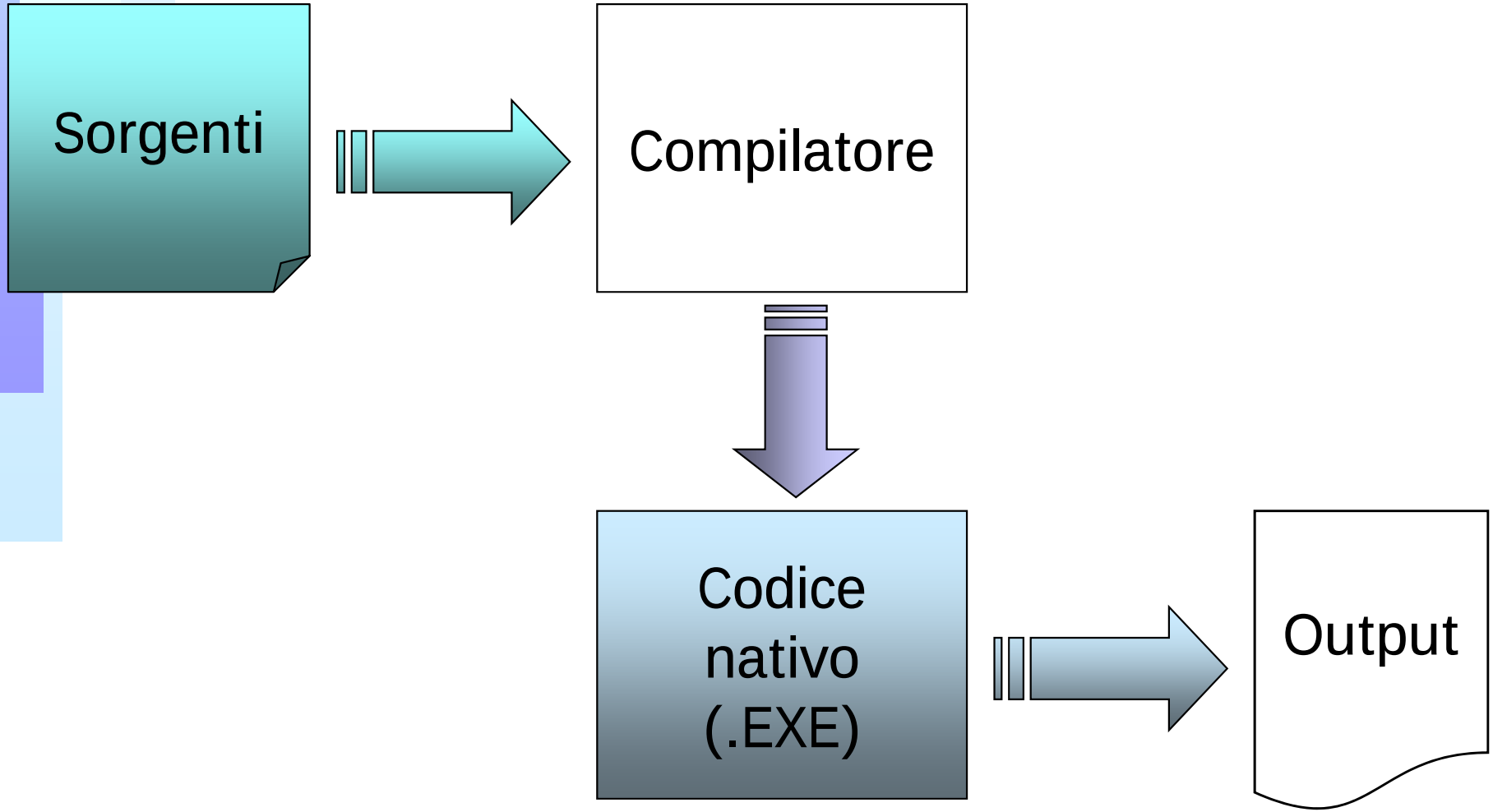
.NET Framework and Tools



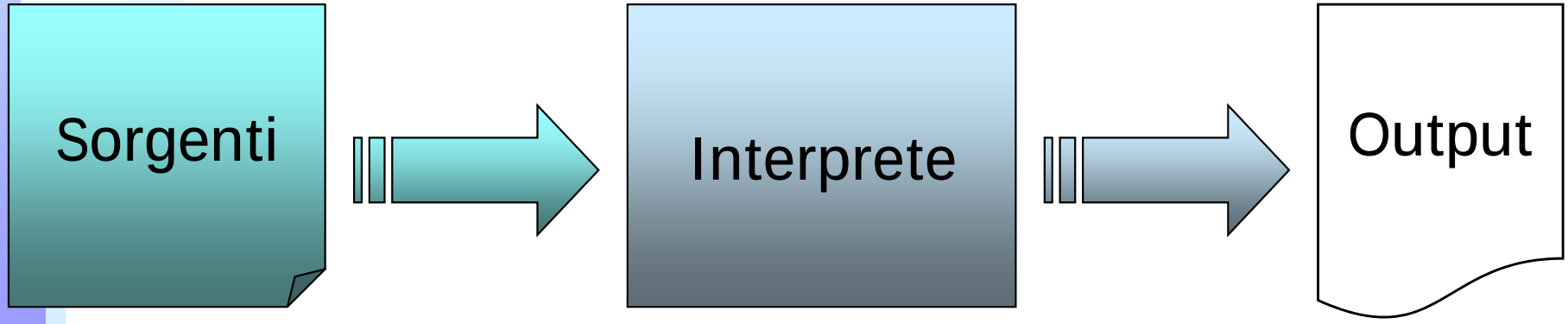
Codici

- Evoluzione
  - Codice nativo
  - Codice interpretato
  - Codice MSIL
- IL:
  - Intermediate Language, Standard ECMA del 1997
- MSIL:
  - Microsoft IL, Implementazione Microsoft di IL

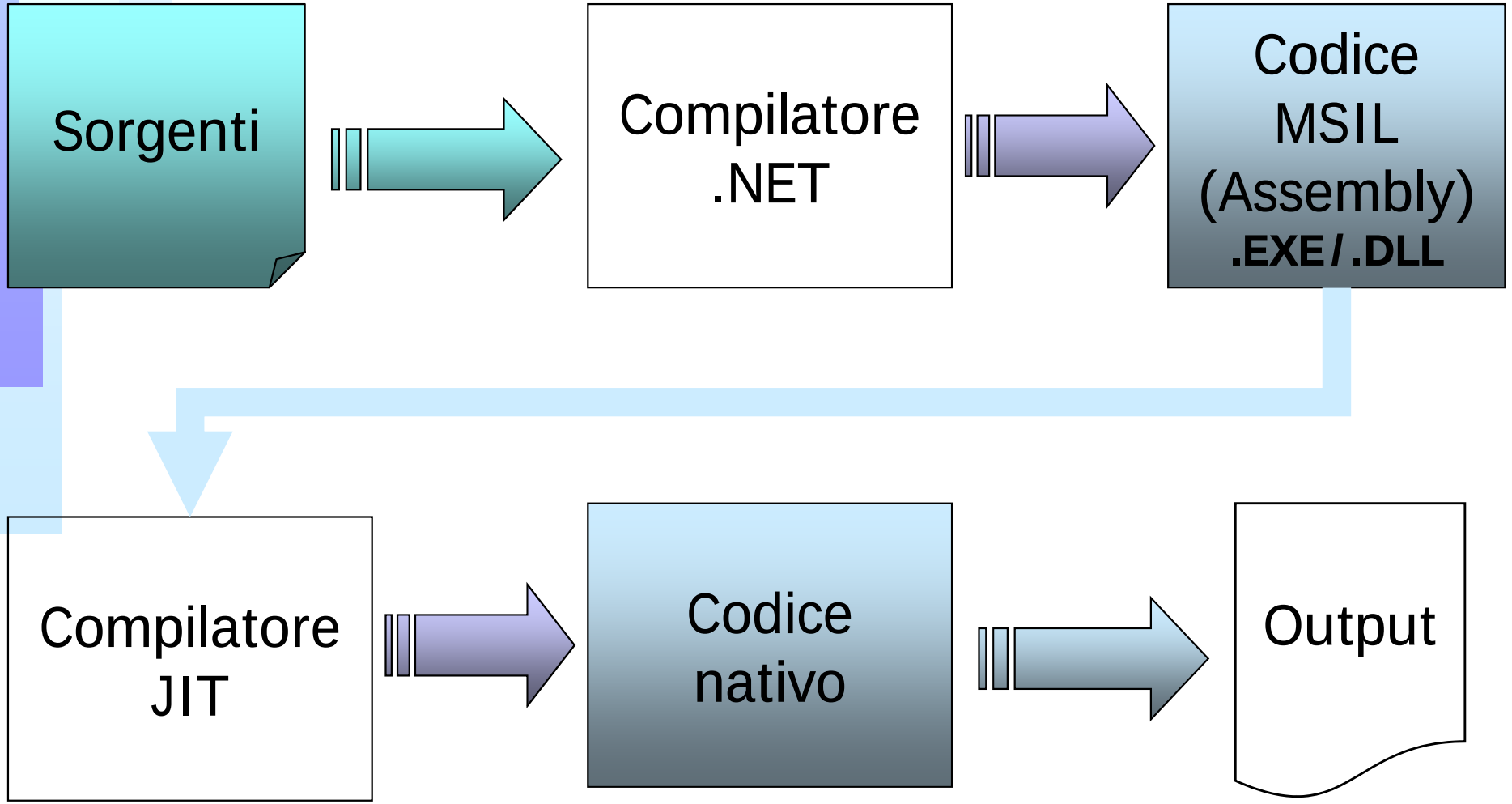
Codice nativo



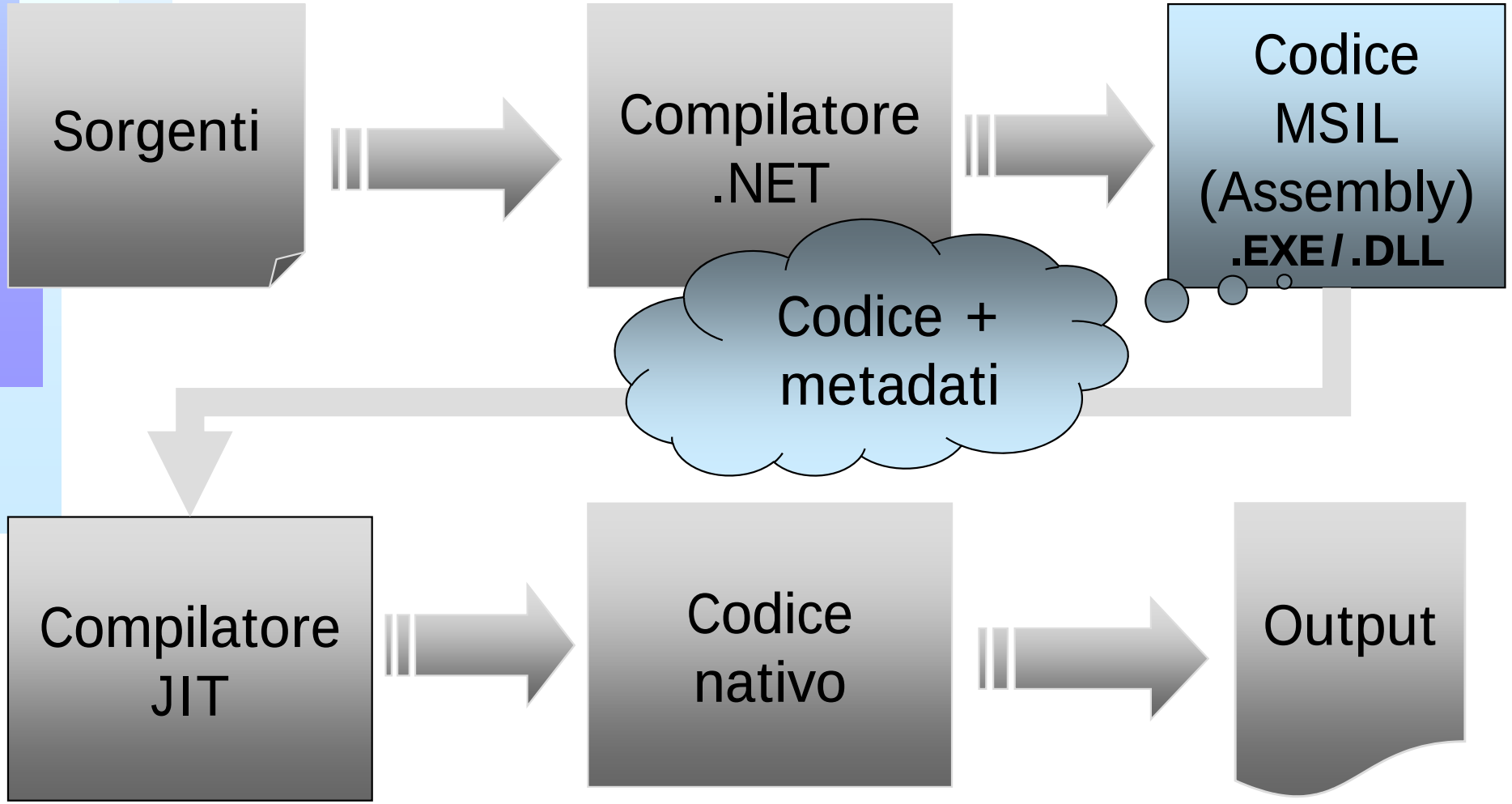
Codice interpretato



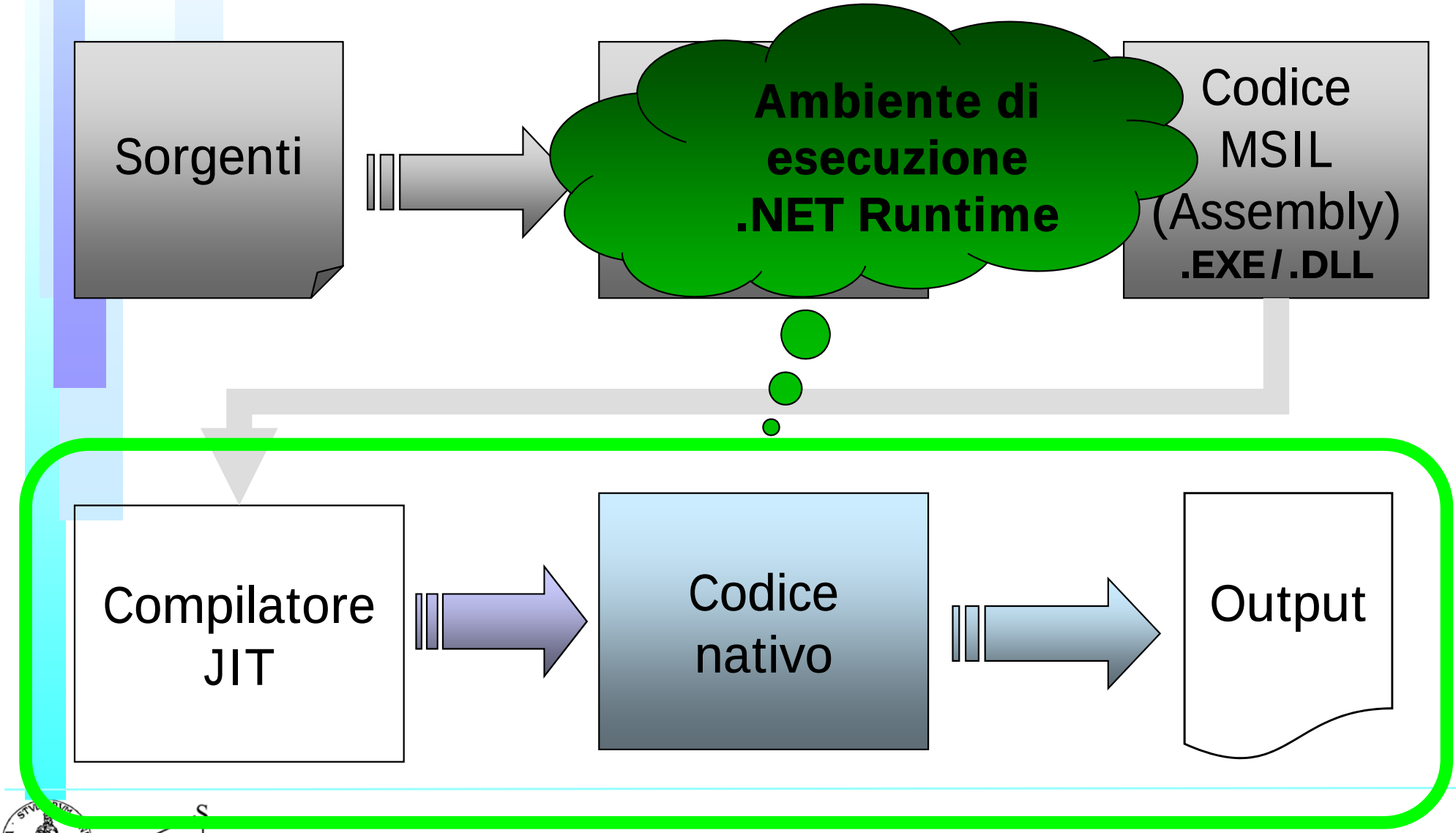
Codice MSIL


Sorgenti
Compilatore .NET
Codice MSIL (Assembly) .EXE / .DLL
Compilatore JIT
Codice nativo
Output

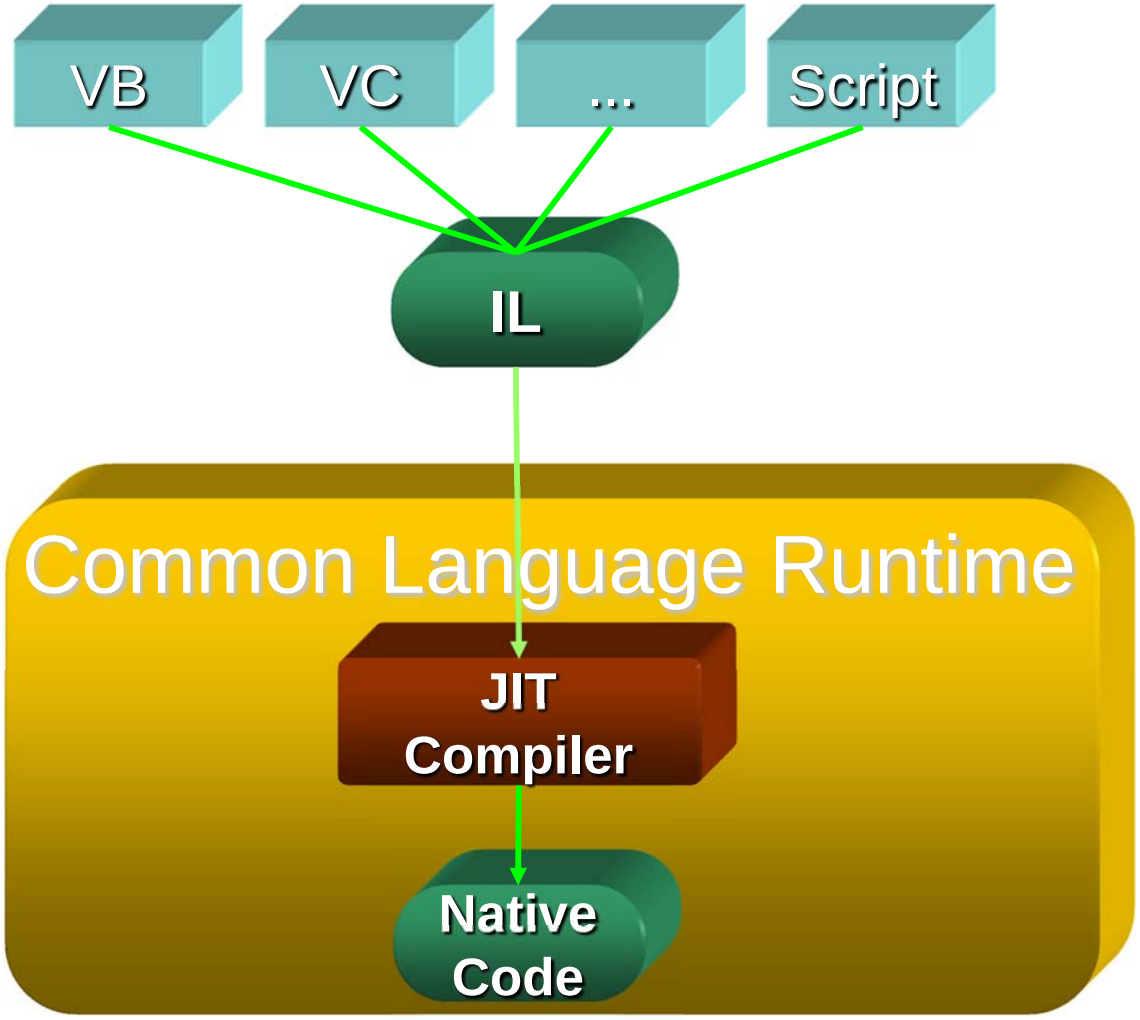
Codice MSIL


Sorgenti
Compilatore .NET
Codice MSIL (Assembly) .EXE / .DLL
Codice + metadati
Compilatore JIT
Codice nativo
Output

Codice MSIL


Sorgenti
Ambiente di esecuzione .NET Runtime
Codice MSIL (Assembly) .EXE / .DLL
Compilatore JIT
Codice nativo
Output

Intermediate Language (IL)



Assembly Language of CLR

Code is never interpreted

Intermediate Language

- Presenta similitudini con linguaggi ad alto livello, ma anche con il linguaggio assembly:
  - Istruzioni per
    - il caricamento, la memorizzazione e l'inizializzazione dei dati
    - richiamare metodi da oggetti
    - aritmetiche e logiche
    - gestione eccezioni di tipo "Try/Catch"
    - Operazioni sui registri, ma indipendente dalla piattaforma
    - Operazioni "atomiche"

Intermediate Language

- Permette al CLR controlli durante la compilazione:
  - Codice Type Safe
  - Puntatori corretti
  - Conversioni corrette
  - ecc.
- Di fatto rappresenta il linguaggio a livello più basso e l'unico "eseguibile" dal CLR

Intermediate Language

- Un compilatore conforme al CLS produce
  - Codice IL
    - Rappresenta il programma vero e proprio
  - Metadati
    - Descrivono i tipi specifici appartenenti al Common Language Types (CLT) utilizzati nel codice, comprendente la definizione di ogni tipo, le signature per ogni membro del tipo, i membri ai quali il codice fa riferimento e gli altri dati che il runtime usa durante l'esecuzione.
    - Permettono componenti autodescrittivi

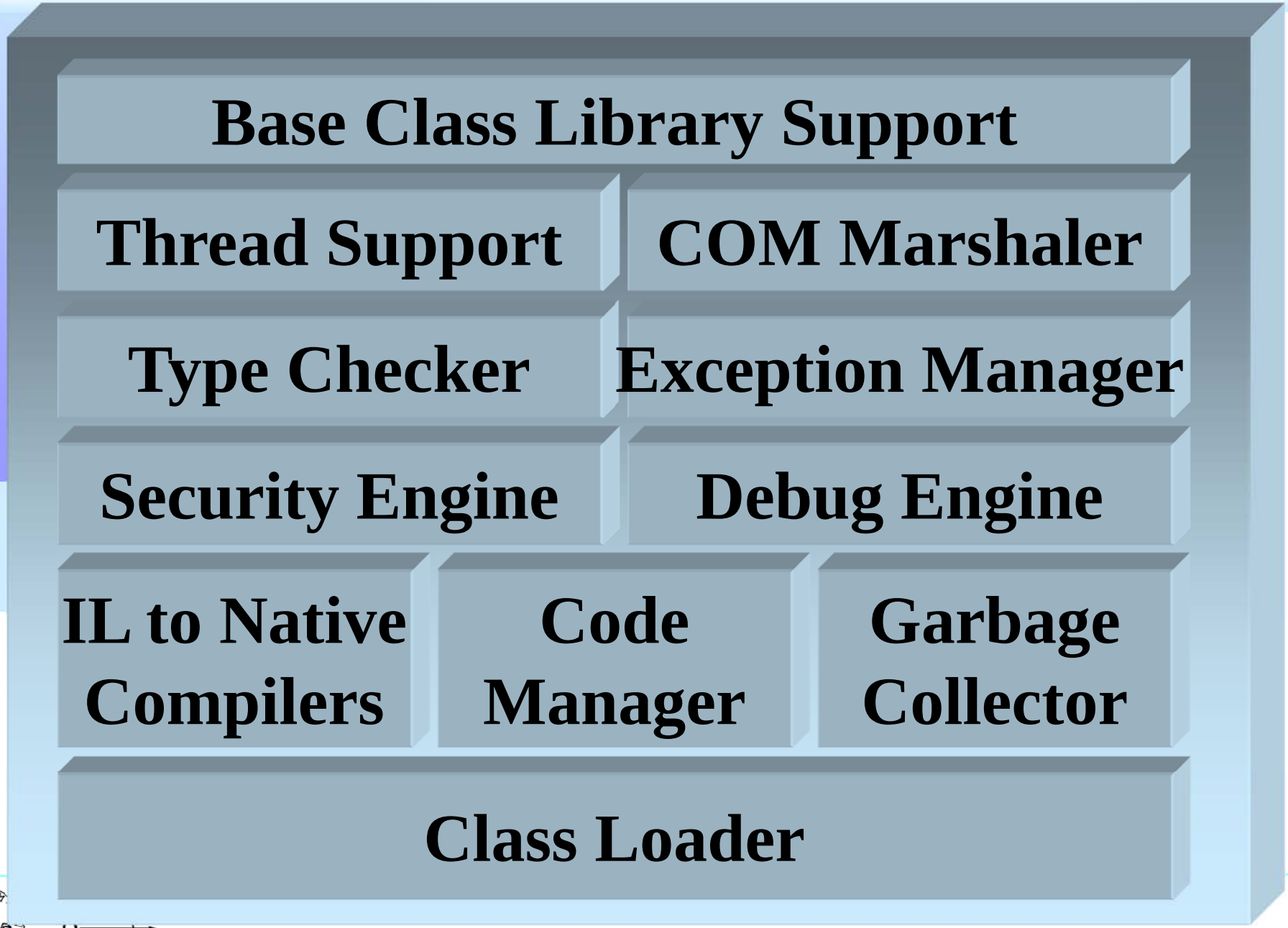
Intermediate Language

- IL e Metadati sono alla fine contenuti in uno o più file PE (Portable Executable) nella forma tradizionale:
  - .exe
    - Se è codice di programma eseguibile
  - .dll
    - Se è un insieme di librerie

Common Language Runtime

- Manages running code
  - Verifies type safety
  - Provides garbage collection, error handling
  - Provides code access security
- Common type system
  - Value types (integer, float, user, …)
  - Objects, Interfaces
  - Delegates, Events, Properties, Pointers
- Access to native system resources

Common Language Runtime



Common Type System

- Sistema di Tipi unificato ed interlinguaggio
- Un insieme standard di tipi di dato e di regole necessarie per la realizzazione di nuovi tipi
- Due Categorie di Tipi disponibili:
  - Value Type
  - Reference Type

Common Type System

- Value Type
  - Tipi atomici come integer e char
  - Divisi in *built-in* ed *user defined*
  - Descrivono valori che sono rappresentati come sequenze di bit
  - Allocati nello Stack del Thread
  - Non soggetti al Garbage Collector
- Reference Type
  - Entità autodefinite contententi sia metodi che variabili
  - Divisi in quattro sottocategorie:
    - Self Describing
    - Interface
    - Pointer
    - Built-in
  - Descrivono valori che sono rappresentati come la locazione di una sequenza di bit
- Reference Type
  - Allocati nell' Heap Gestito (Managed Heap)
  - Soggetti al Garbage Collector

Common Language Specification

- Il CLS definisce un sottoinsieme del Common Type System al quale tutti i fornitori di librerie di classi e progettisti di linguaggi che puntano al CLR, devono aderire.

- Se un componente scritto in un linguaggio (ad esempio C#) dovrà essere utilizzato da un altro linguaggio (ad esempio VB.NET), allora chi scrive il componente dovrà aderire ai tipi e alle strutture definite dal CLS.
  - Ad esempio, il tipo Int32 è compatibile con il CLS ed i linguaggi e gli strumenti possono aspettarsi che altri linguaggi e strumenti conformi al CLS sappiano come utilizzarlo correttamente

JIT – Just in Time Compiler

- Il codice non viene caricato tutto in memoria
- il compilatore JIT compila solo il codice necessario, quindi memorizza nella cache il codice nativo compilato per riutilizzarlo
- L'overhead è una lieve differenza che, nella maggior parte dei casi, non verrà rilevata

- Quando viene caricata una classe, il caricatore aggiunge uno stub a ogni metodo della classe
- La prima volta che viene chiamato il metodo, il codice stub cede il controllo al compilatore JIT, che compila MSIL nel codice nativo.
- Lo stub viene quindi modificato per puntare al codice nativo appena creato, affinché le chiamate successive passino direttamente al codice nativo

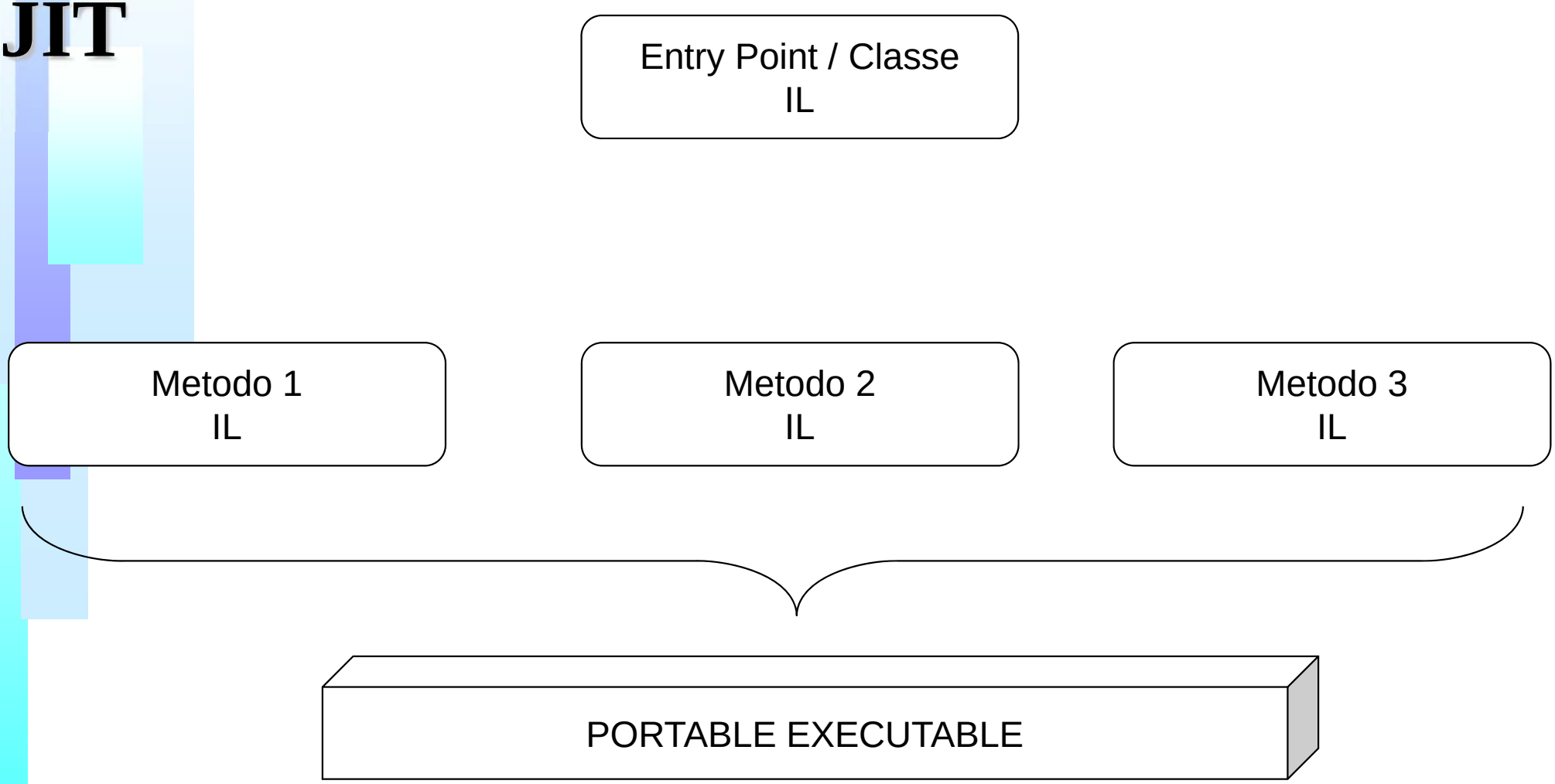
Just In Time Compiler

- Compilatore al volo basato sul concetto JIT:
  - Non tutto l'IL di un PE viene eseguito durante un programma, solo la parte necessaria viene compilata un istante prima della sua esecuzione.
  - Il codice compilato viene memorizzato per successive esecuzioni
  - Tutto il codice .NET è compilato JIT, anche linguaggi di scripting come VB Script, J Script, JavaScript ecc.

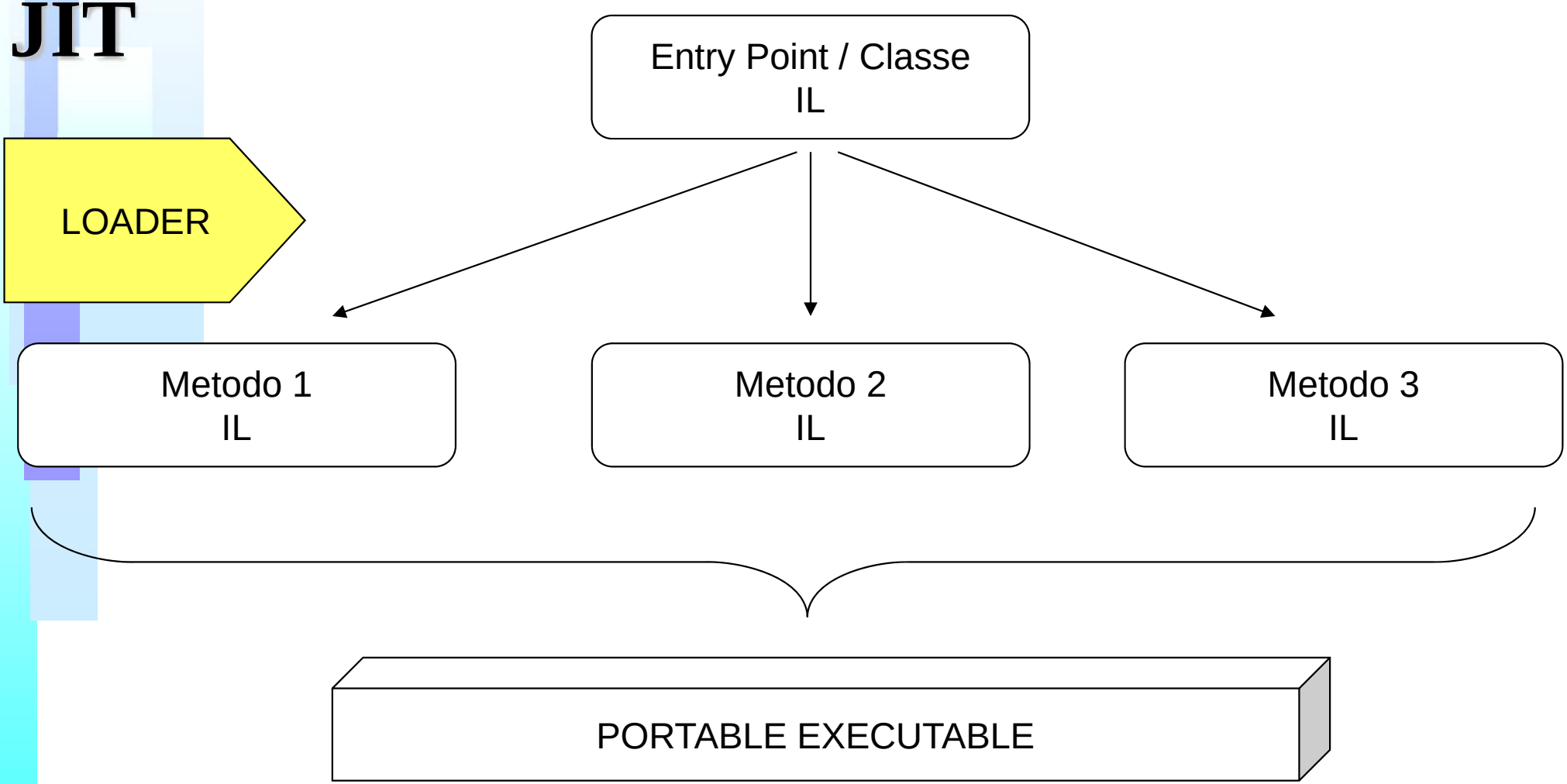
Just In Time Compiler

- Solo il codice usato verrà compilato
- Minore occupazione di memoria
- Facile rimozione del codice inutilizzato da tempo
- Controlli sull'IL in fase di compilazione
- Dati per la compilazione contenuti nello stesso file del codice (metadati)
- Compilazione ottimizzante perché conosce lo stato preciso dell'ambiente di esecuzione

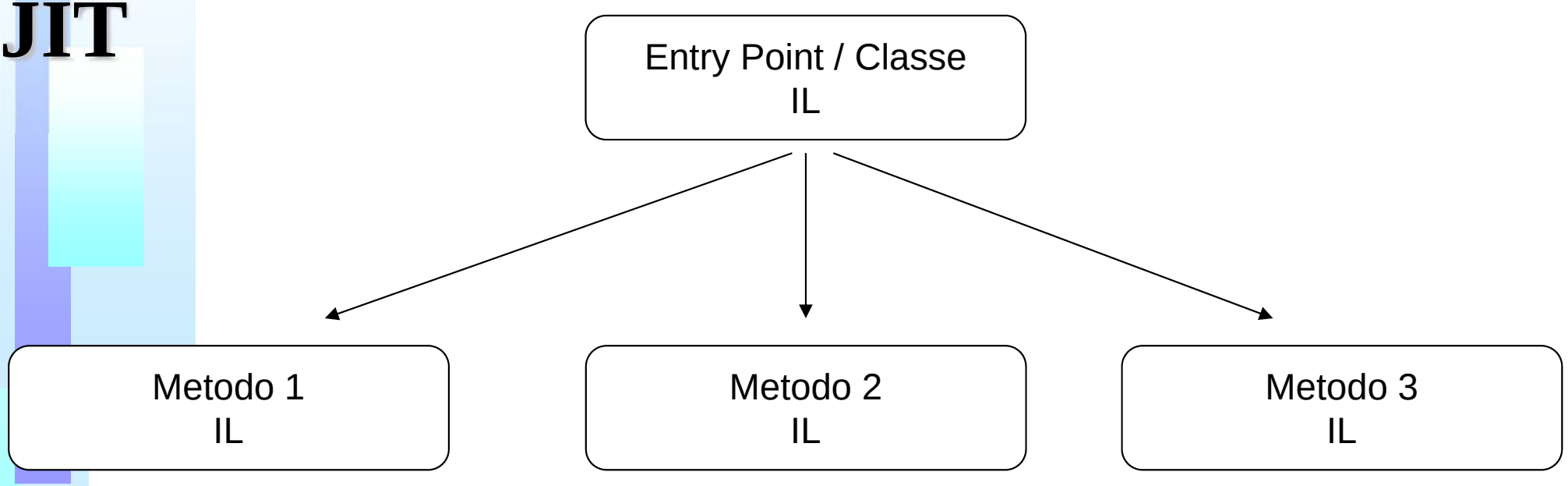
JIT


Entry Point / Classe
IL
Metodo 1
IL
Metodo 2
IL
Metodo 3
IL
PORTABLE EXECUTABLE

JIT

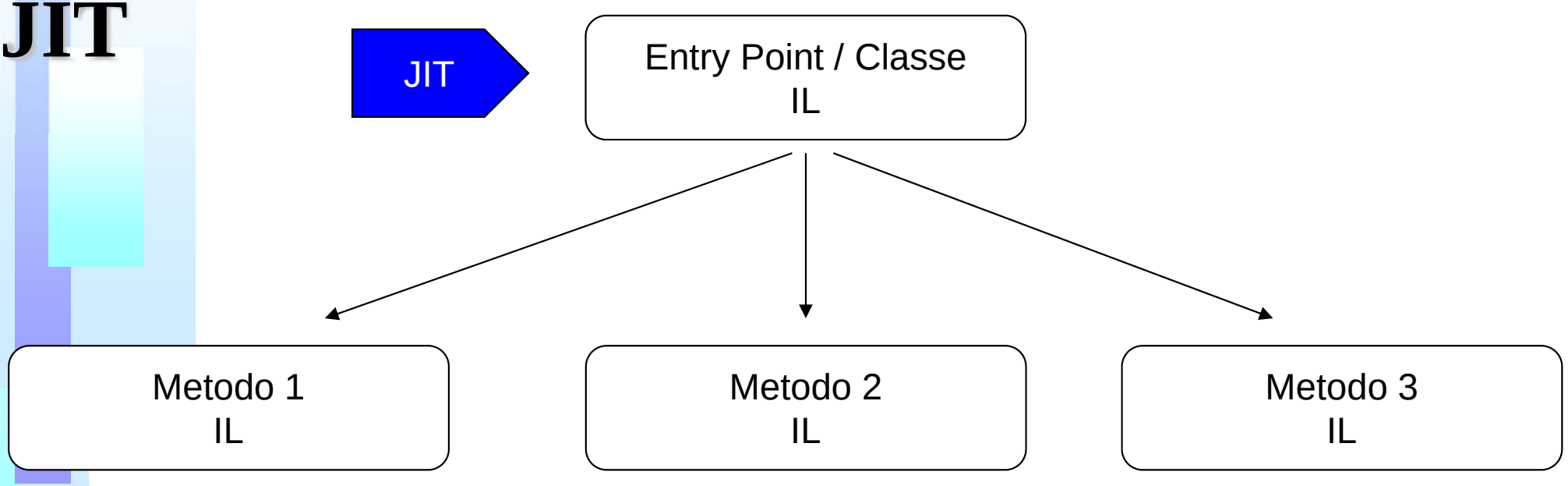

Entry Point / Classe
IL
LOADER
Metodo 1
IL
Metodo 2
IL
Metodo 3
IL
PORTABLE EXECUTABLE

JIT



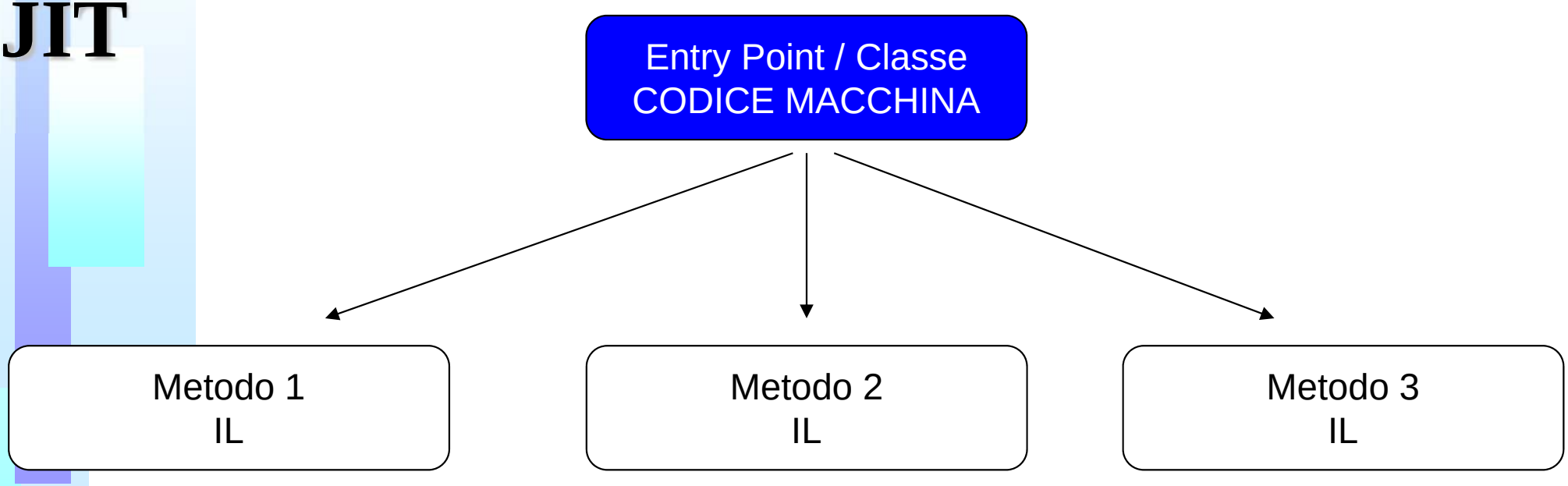
Esecuzione

JIT



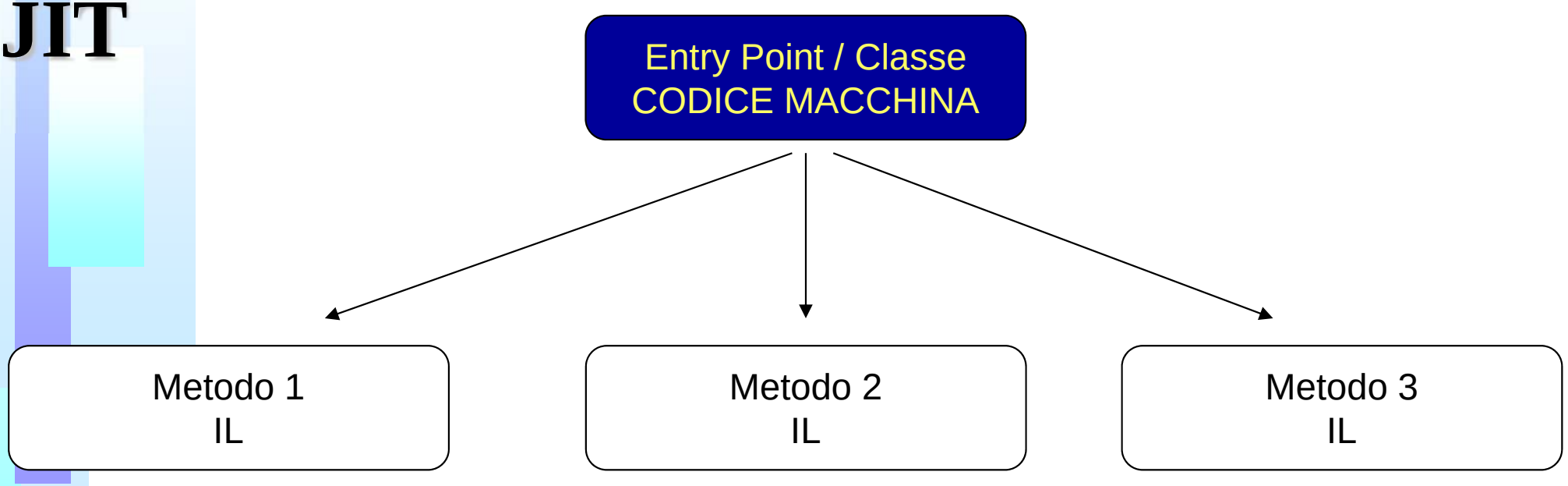
Esecuzione

JIT



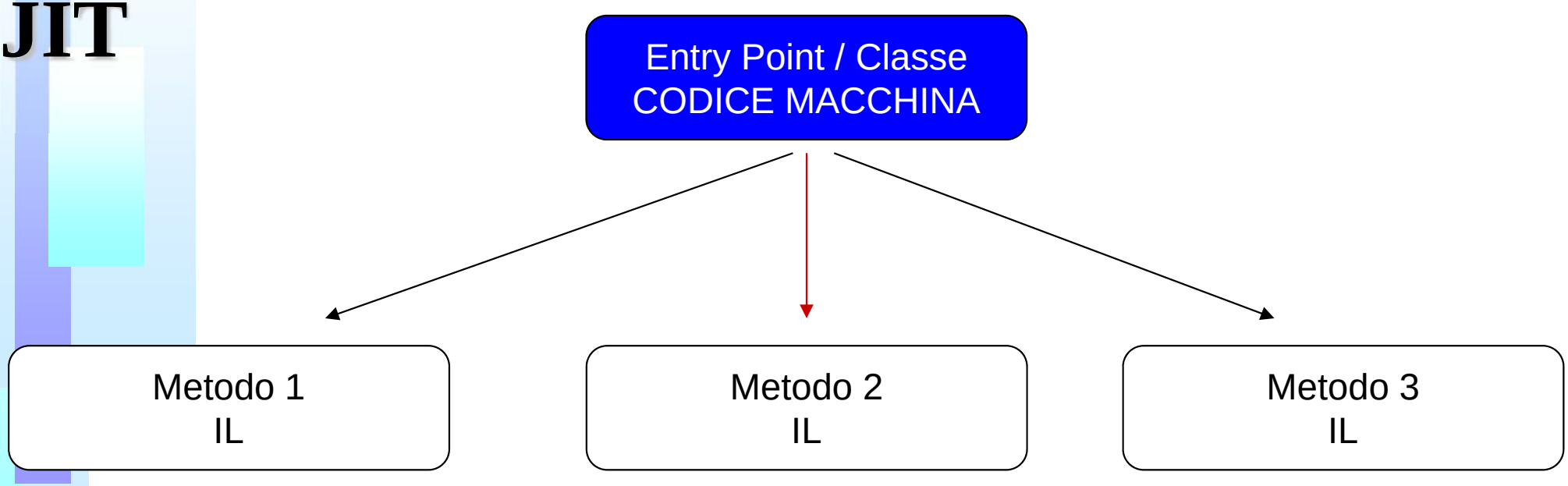
Esecuzione

JIT

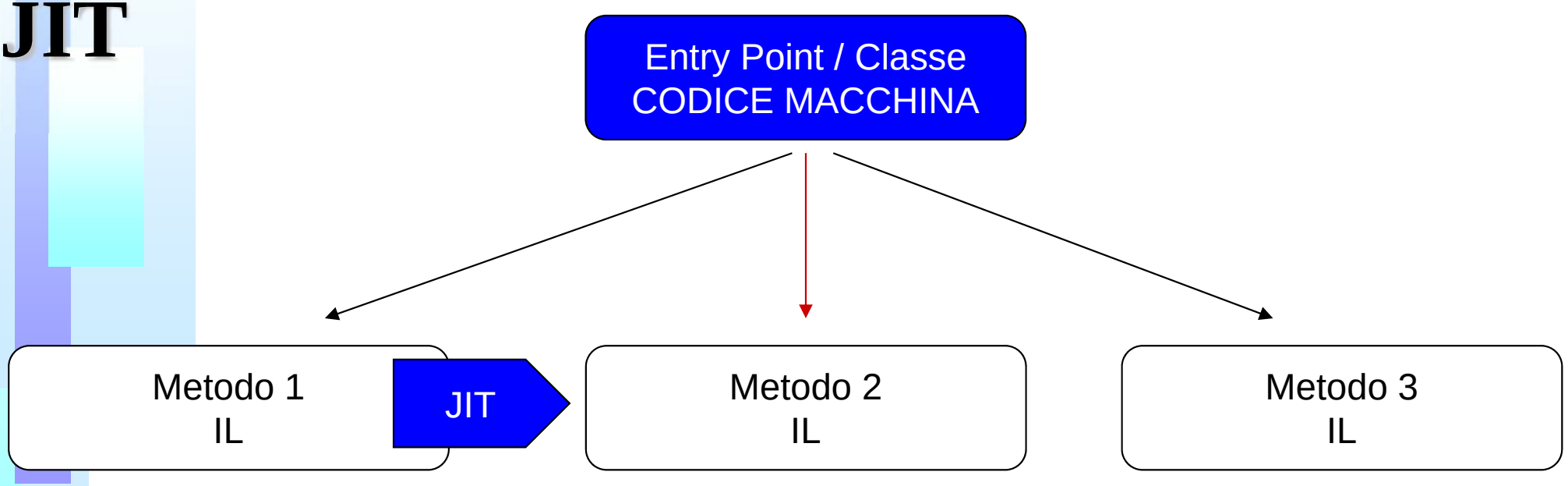


Esecuzione

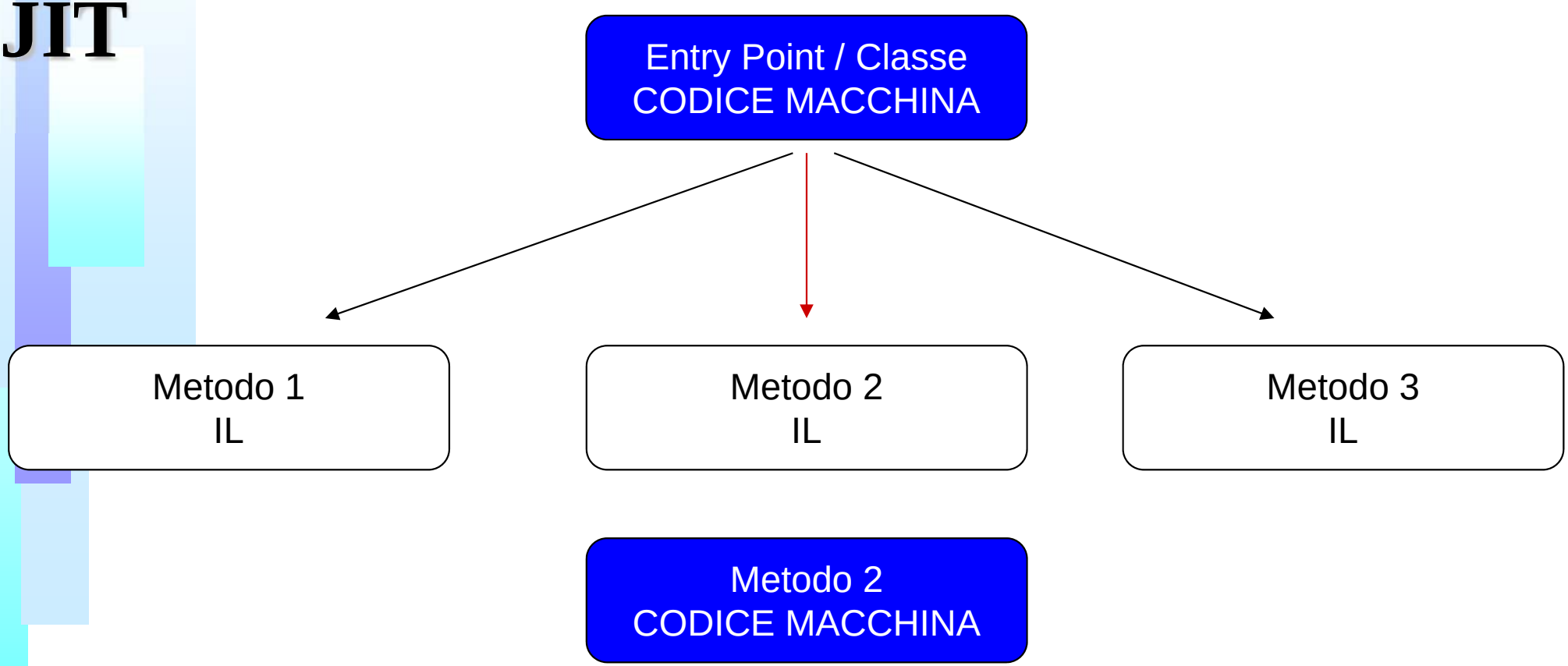
JIT


Entry Point / Classe
CODICE MACCHINA
Metodo 1
IL
Metodo 2
IL
Metodo 3
IL

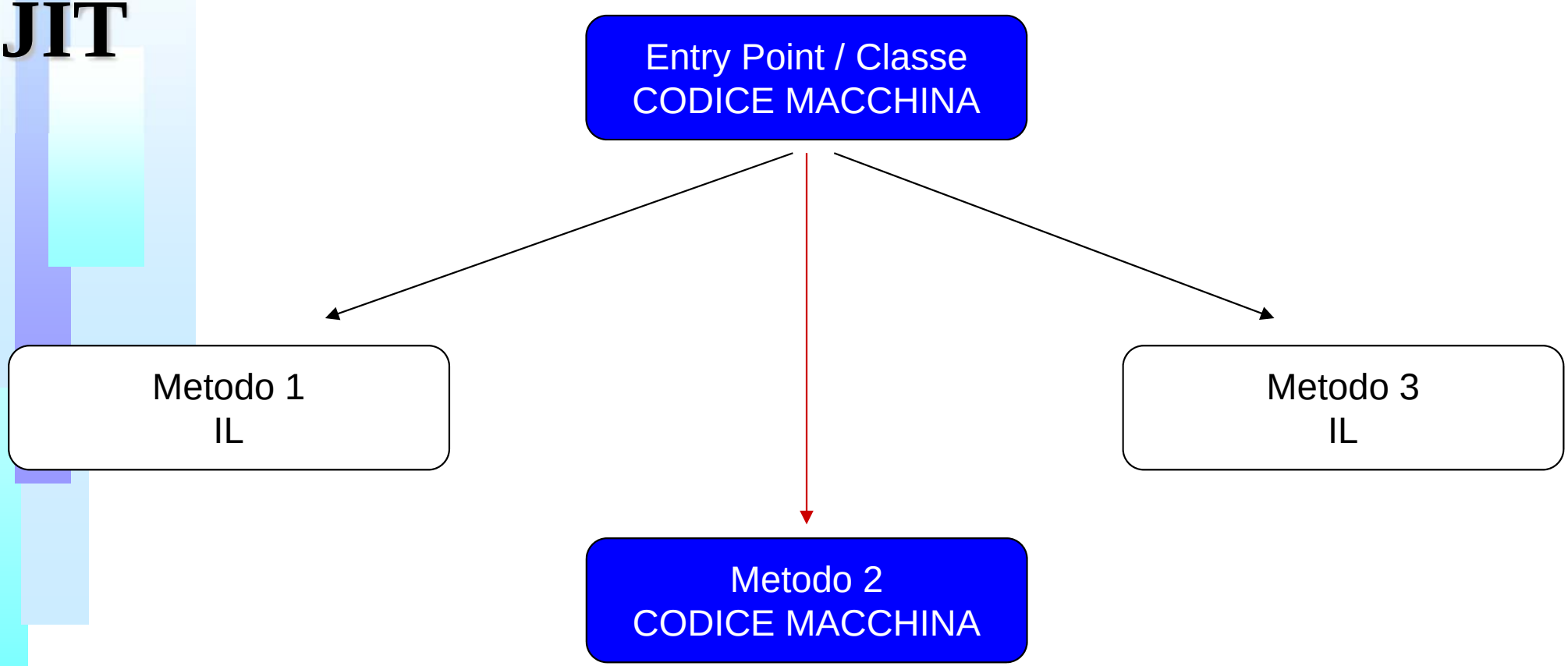
JIT


Entry Point / Classe
CODICE MACCHINA
Metodo 1
IL
JIT
Metodo 2
IL
Metodo 3
IL

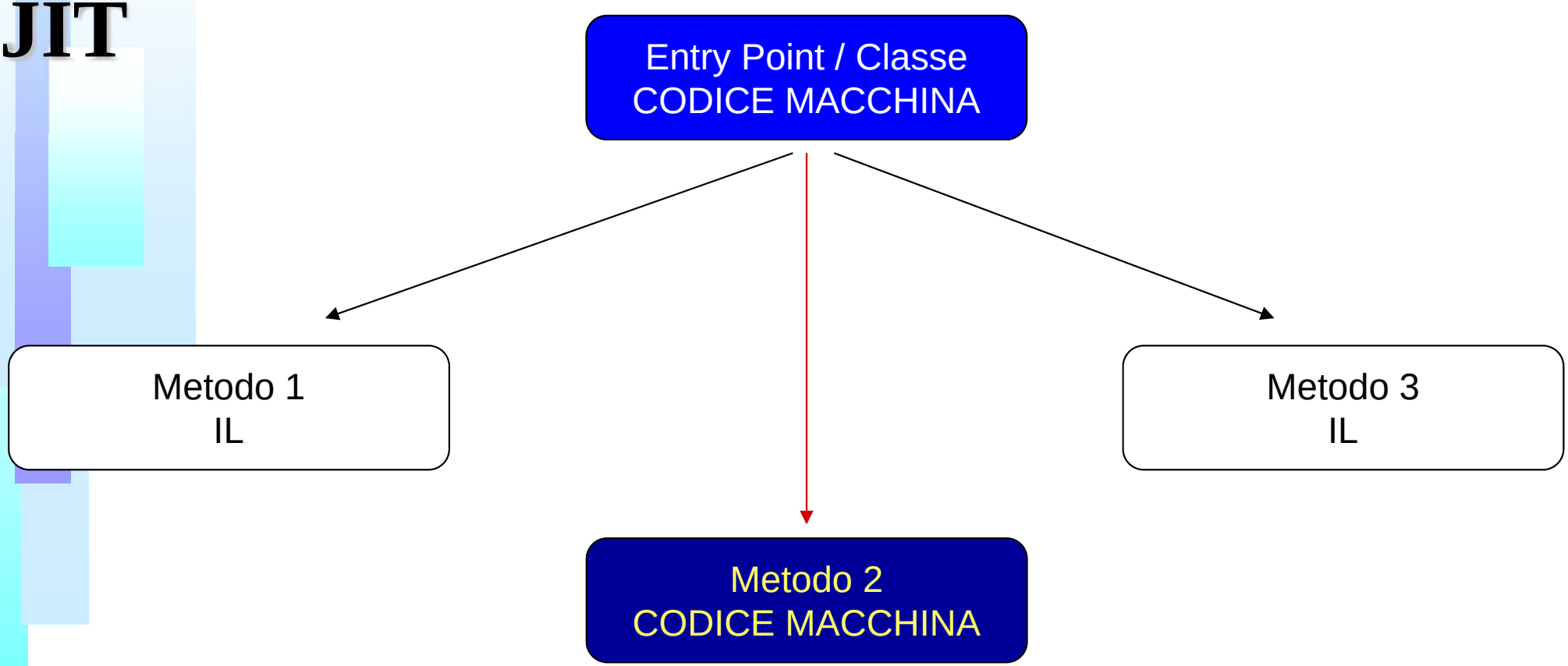
JIT


Entry Point / Classe
CODICE MACCHINA
Metodo 1
IL
Metodo 2
IL
Metodo 3
IL
Metodo 2
CODICE MACCHINA

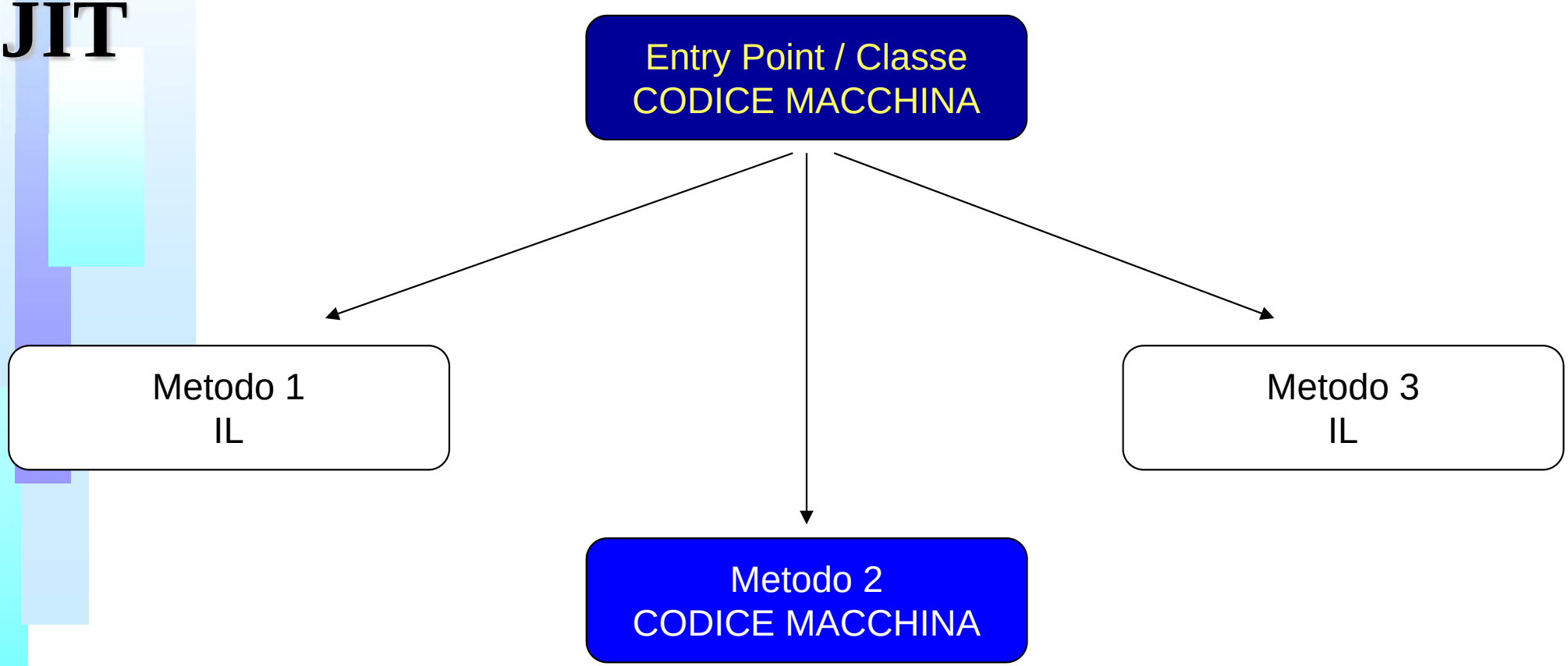
JIT


Entry Point / Classe
CODICE MACCHINA
Metodo 1
IL
Metodo 3
IL
Metodo 2
CODICE MACCHINA

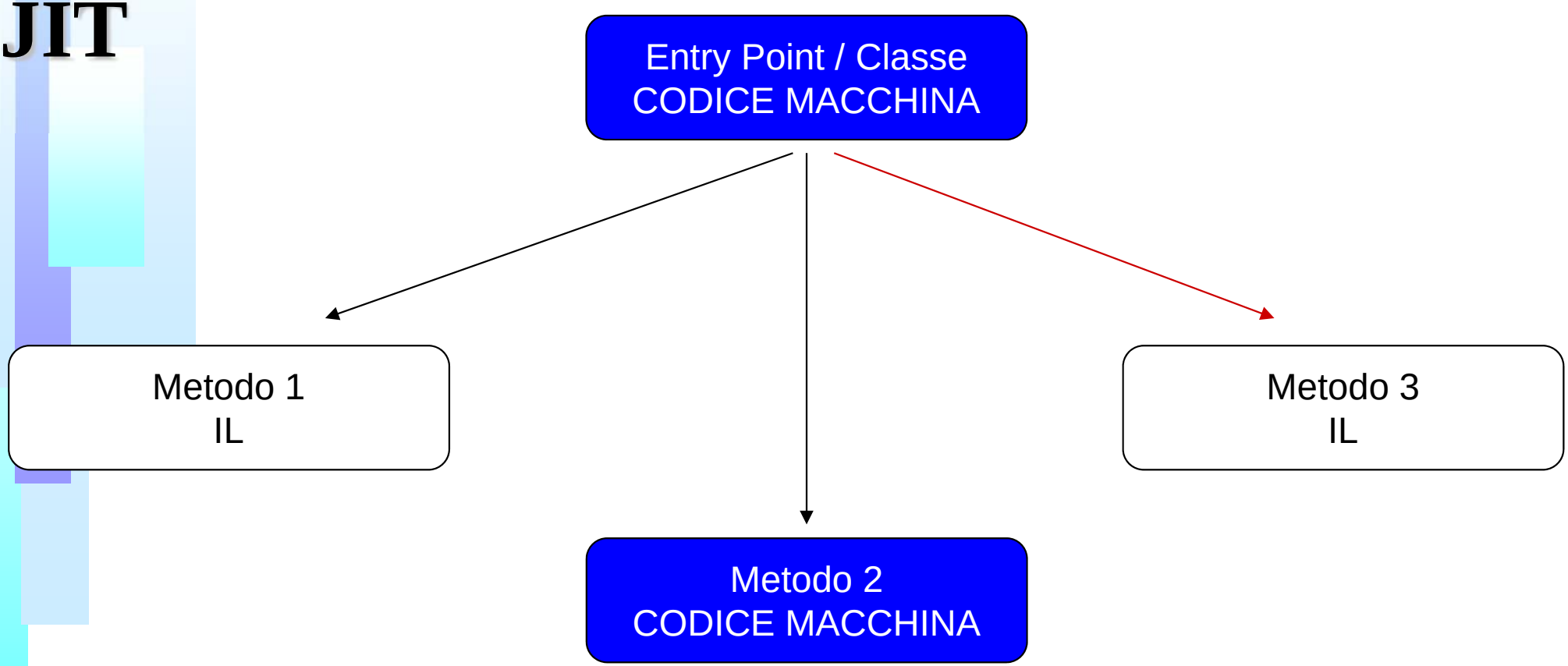
JIT


Entry Point / Classe
CODICE MACCHINA
Metodo 1
IL
Metodo 3
IL
Metodo 2
CODICE MACCHINA

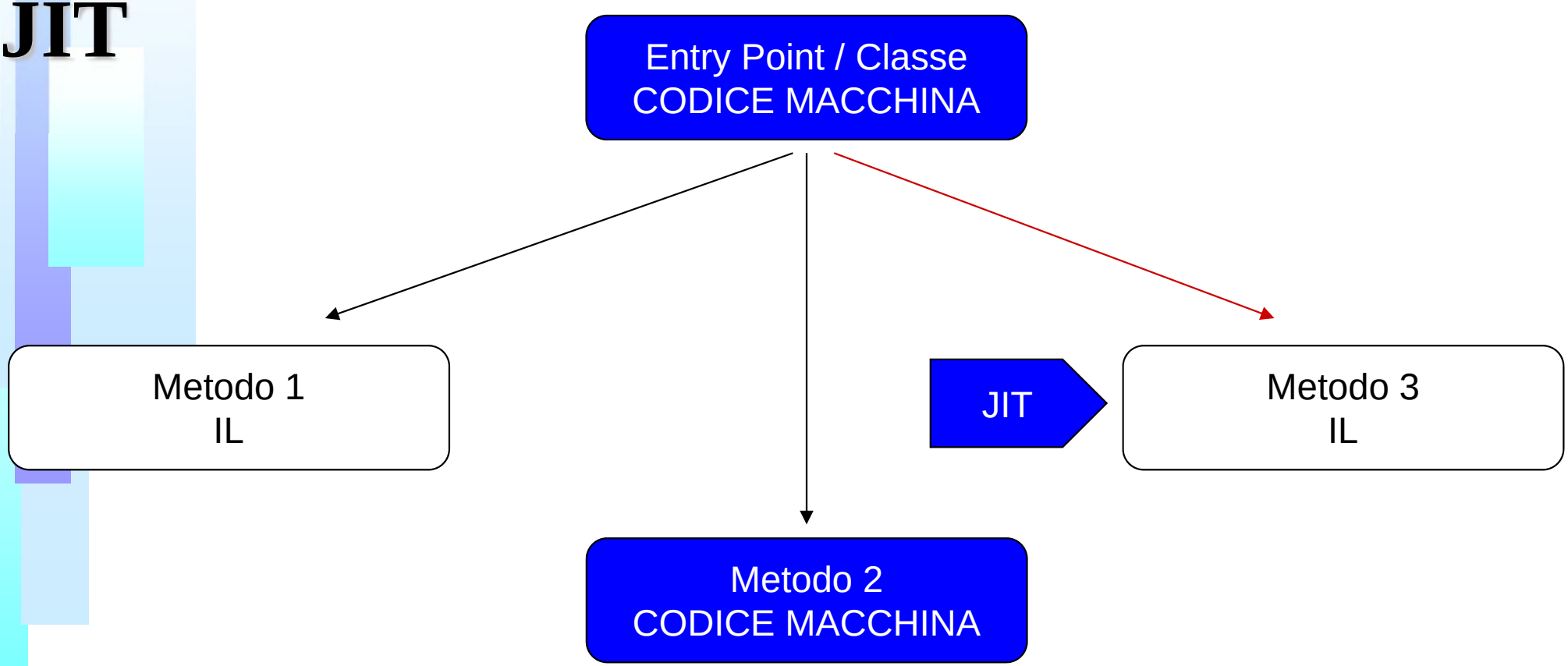
JIT


Entry Point / Classe
CODICE MACCHINA
Metodo 1
IL
Metodo 3
IL
Metodo 2
CODICE MACCHINA

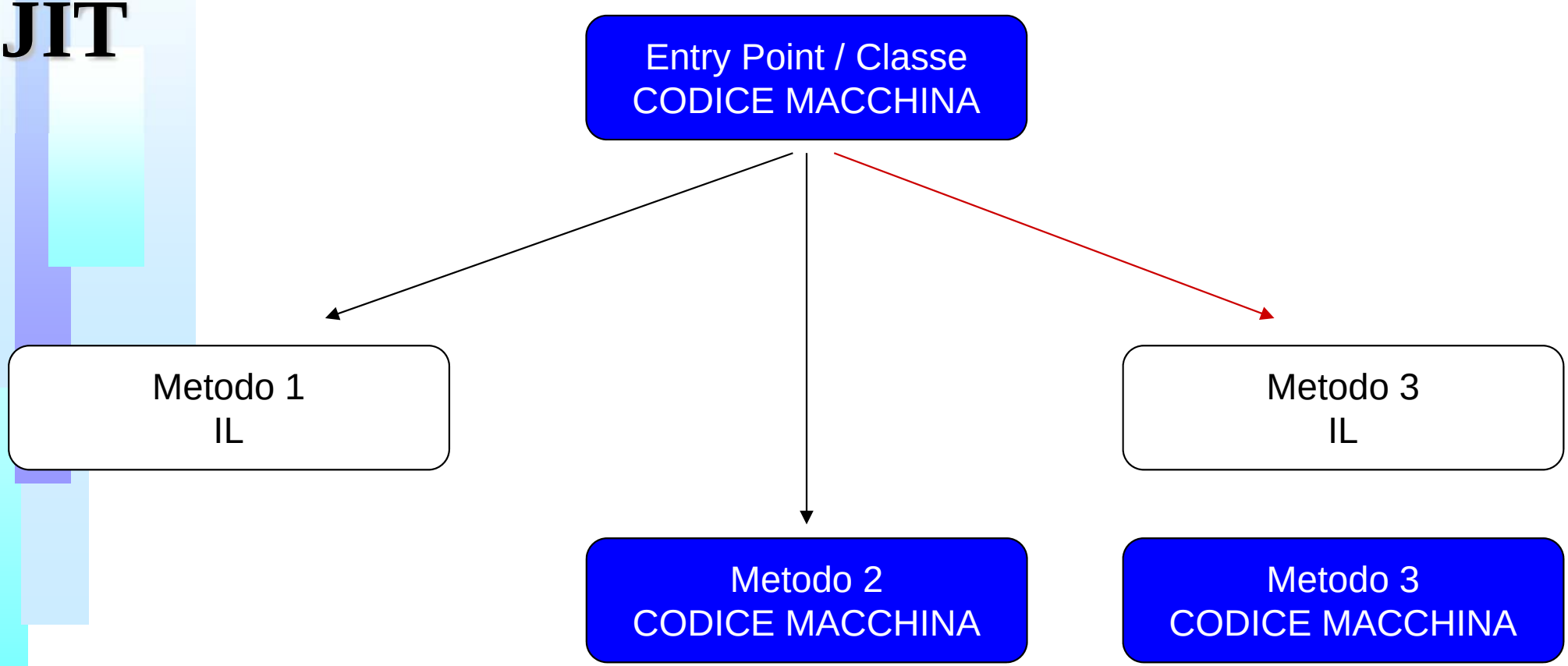
JIT


Entry Point / Classe
CODICE MACCHINA
Metodo 1
IL
Metodo 3
IL
Metodo 2
CODICE MACCHINA

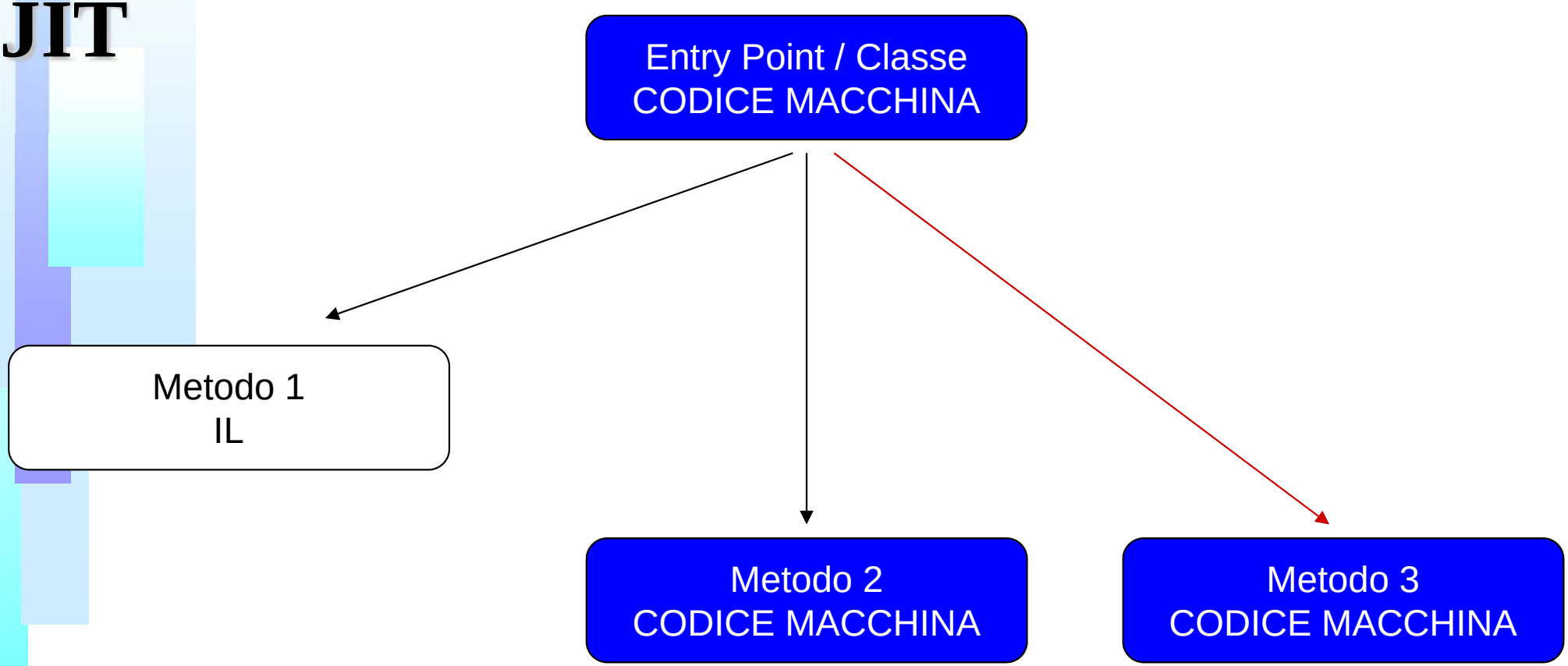
JIT


Entry Point / Classe
CODICE MACCHINA
Metodo 1
IL
JIT
Metodo 3
IL
Metodo 2
CODICE MACCHINA

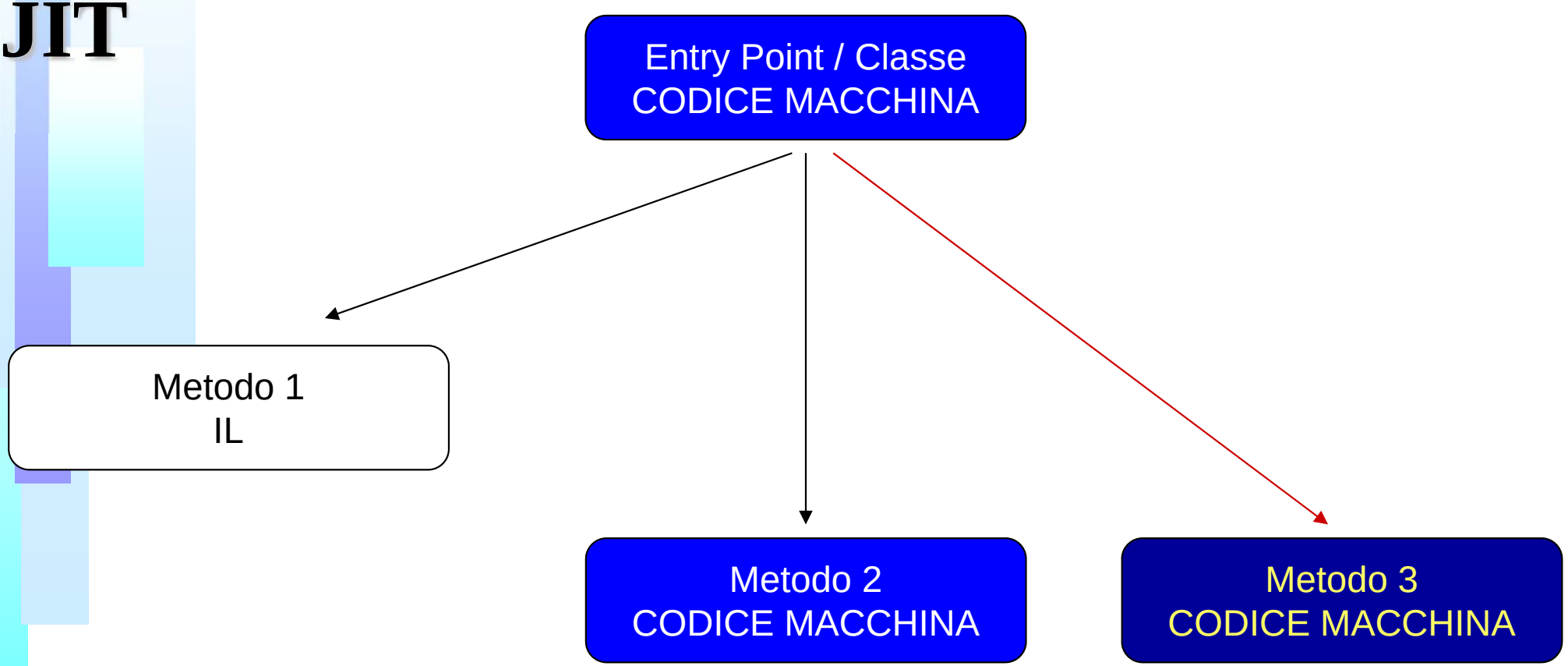
JIT


Entry Point / Classe
CODICE MACCHINA
Metodo 1
IL
Metodo 3
IL
Metodo 2
CODICE MACCHINA
Metodo 3
CODICE MACCHINA

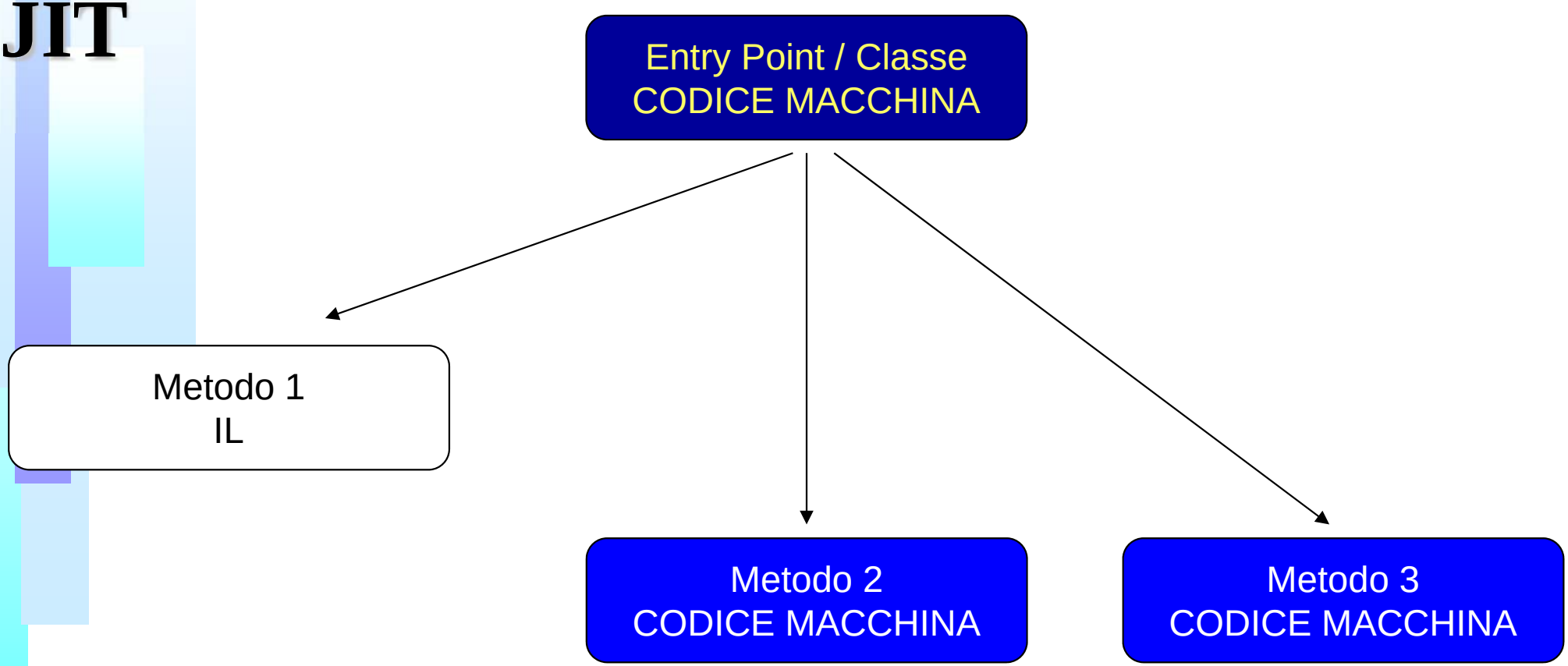
JIT


Entry Point / Classe
CODICE MACCHINA
Metodo 1
IL
Metodo 2
CODICE MACCHINA
Metodo 3
CODICE MACCHINA

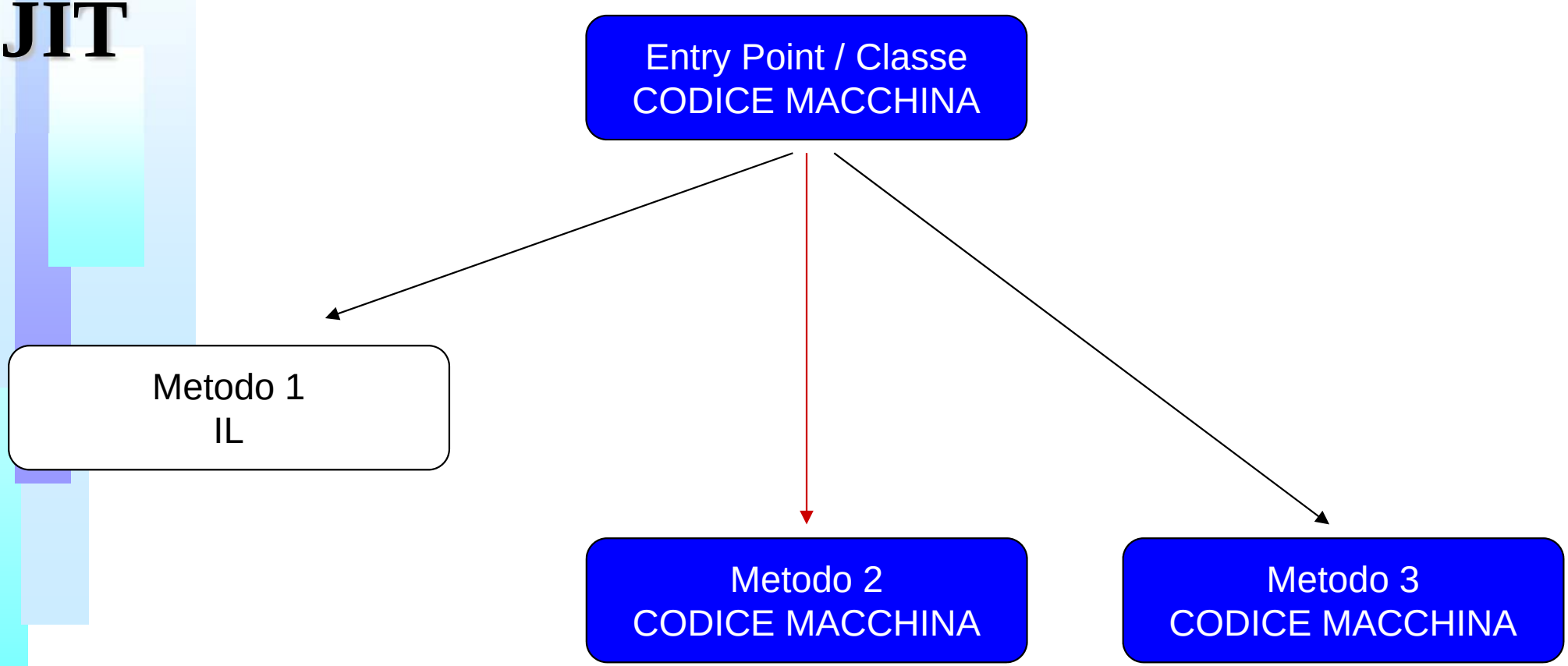
JIT


Entry Point / Classe
CODICE MACCHINA
Metodo 1
IL
Metodo 2
CODICE MACCHINA
Metodo 3
CODICE MACCHINA

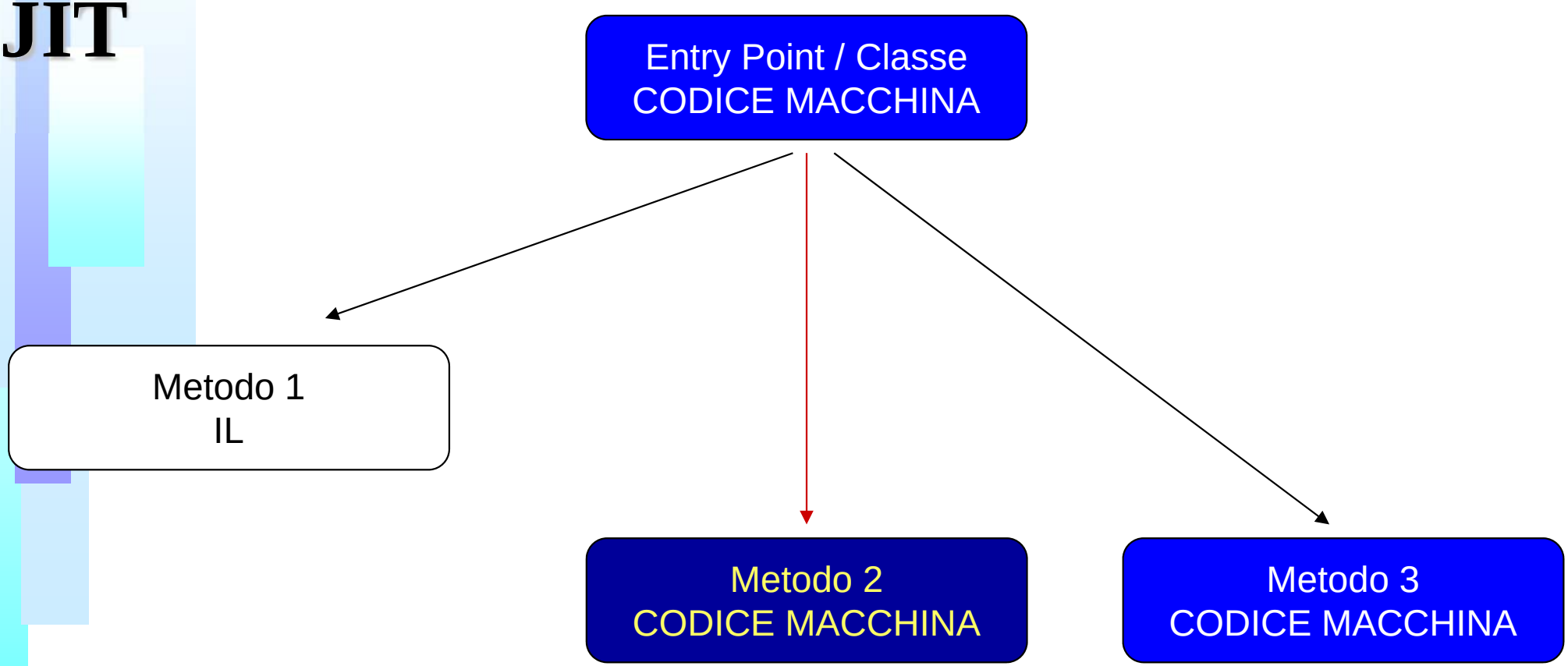
JIT


Entry Point / Classe
CODICE MACCHINA
Metodo 1
IL
Metodo 2
CODICE MACCHINA
Metodo 3
CODICE MACCHINA

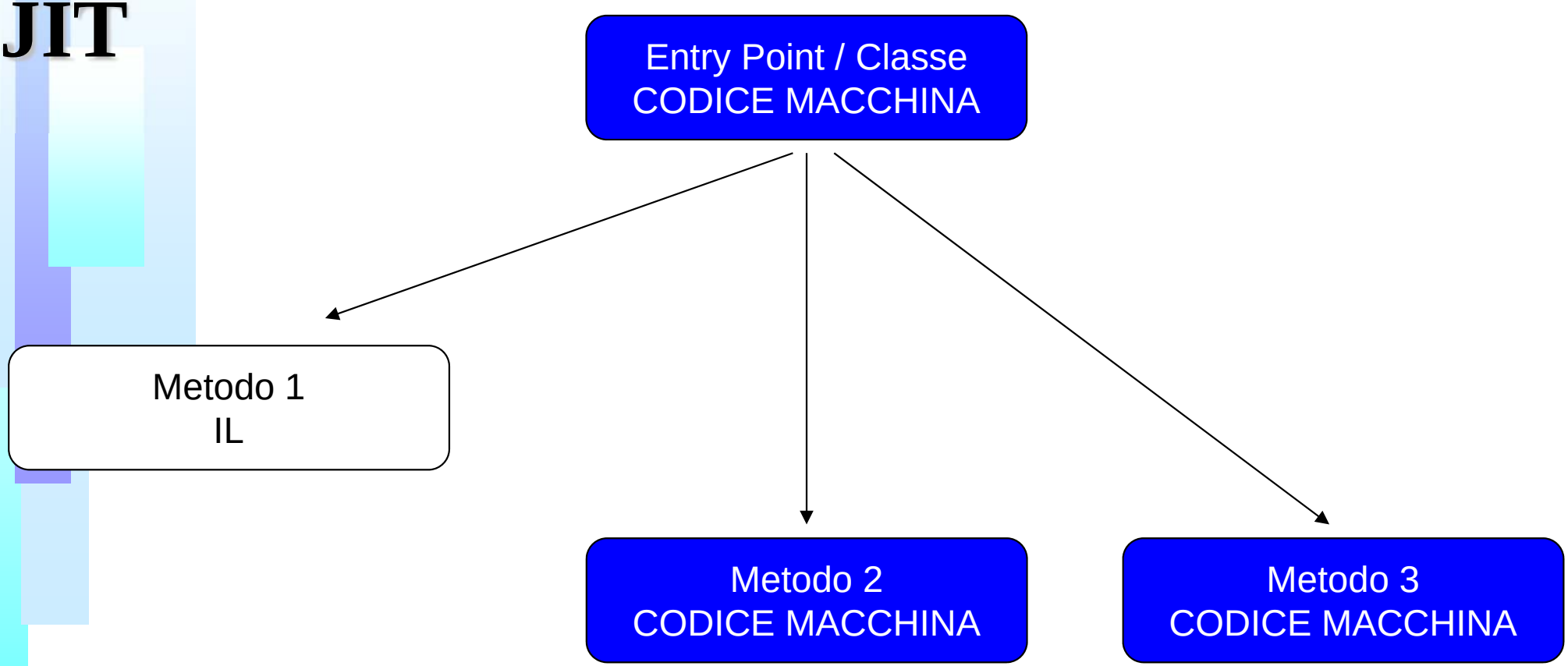
JIT


Entry Point / Classe
CODICE MACCHINA
Metodo 1
IL
Metodo 2
CODICE MACCHINA
Metodo 3
CODICE MACCHINA

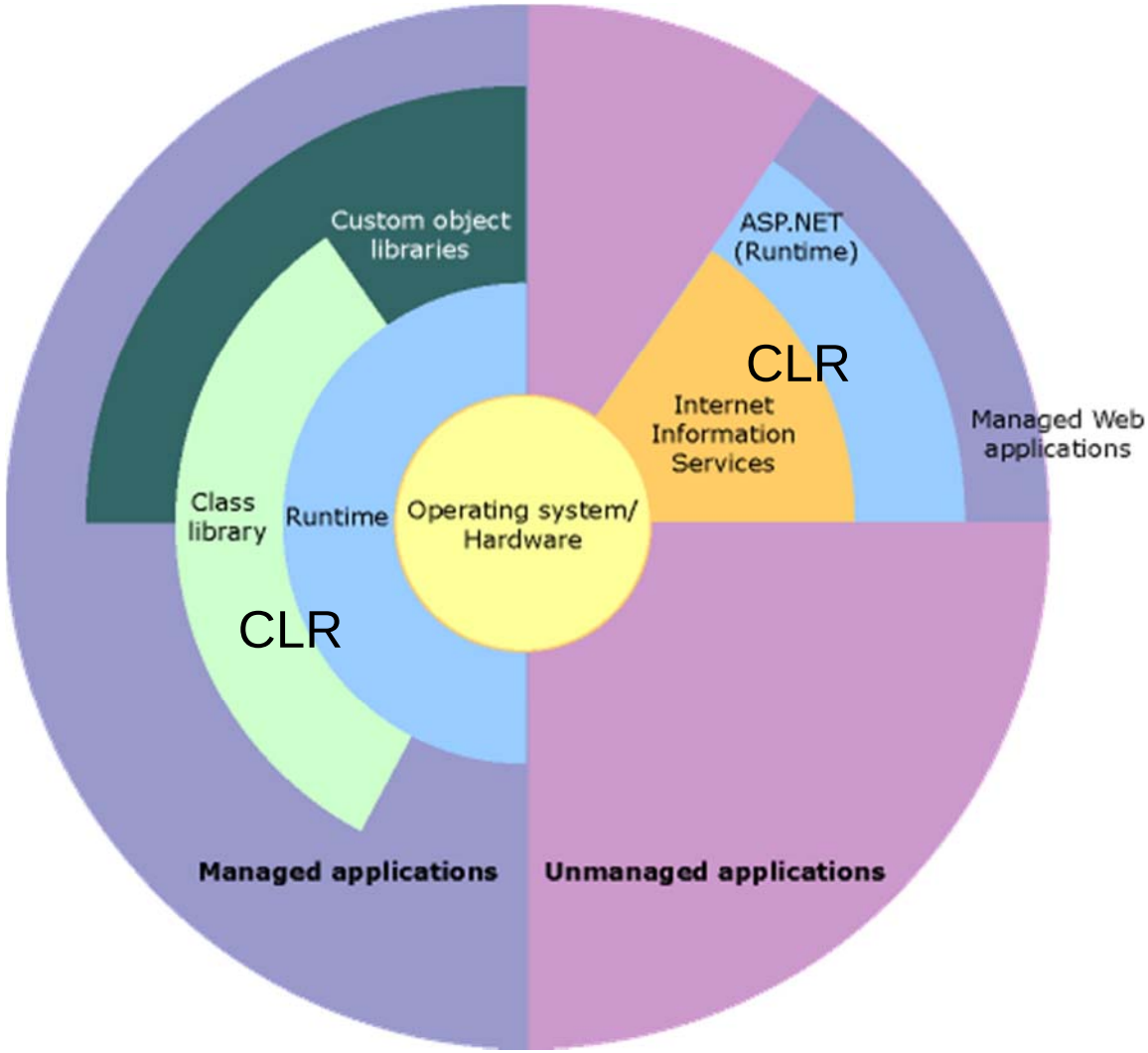
JIT


Entry Point / Classe
CODICE MACCHINA
Metodo 1
IL
Metodo 2
CODICE MACCHINA
Metodo 3
CODICE MACCHINA

JIT


Entry Point / Classe
CODICE MACCHINA
Metodo 1
IL
Metodo 2
CODICE MACCHINA
Metodo 3
CODICE MACCHINA

.NET – Hierarchy, Another View



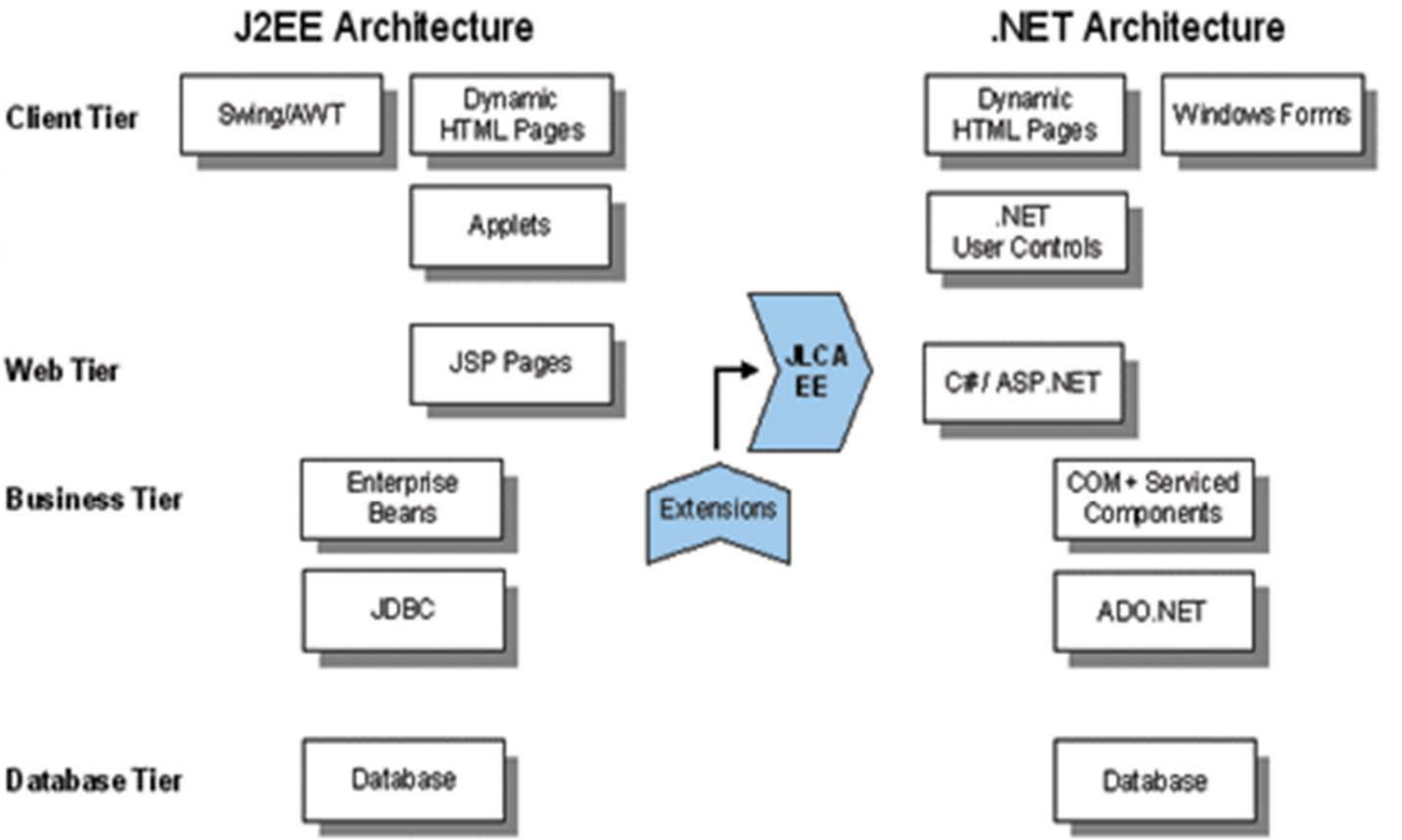
Indipendenza dalla piattaforma

- .NET è un'implementazione di CLI
  - Common Language Infrastructure
- CLI è uno standard ECMA
  - ECMA-334, ECMA-335
- Esistono già altre implementazioni di CLI:
  - SSCLI (Microsoft, per Windows, FreeBSD e Macintosh)
  - Mono (per Linux)
  - DotGNU
  - Intel OCL (Open CLI Library)
  - …

Comparison

- Java
  - One language
  - Multiple platforms
- .NET
  - Multiple languages
  - Multiple platforms
    - Windows
    - FreeBSD (Rotor)
    - Linux, GNU (in progress)
    - Mono Project (Ximian)

Comparison between J2EE and .NET Architectures



J2EE and .NET

Execution Engine

- J2EE

  Java source code compiles into machine-independent byte code

  Runtime Environment : JVM

- .NET

  Any compliant language compiles into MSIL

  Runtime environment : CLR

Both JVM and CLR ,support services, such as code verification, memory management via garbage collection, and code security

J2EE and .NET

Cross Platform Portability

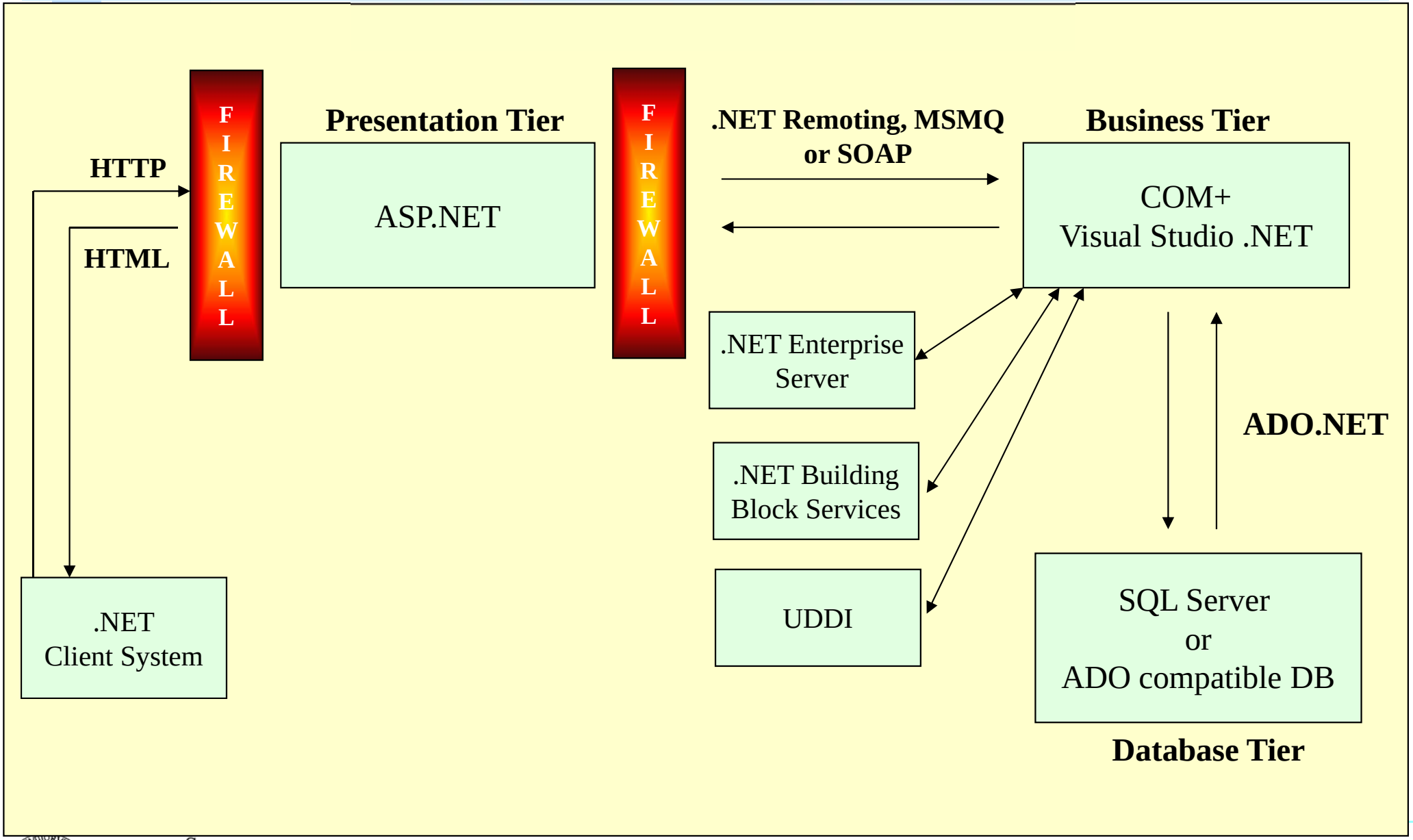
- J2EE
  - Platform Independent
  - JDK should exist on target machine
- .NET
  - Supports Windows platform
  - CLR should exist on target machine
  - Can support other platforms provided it has its own JIT complier

J2EE and .NET

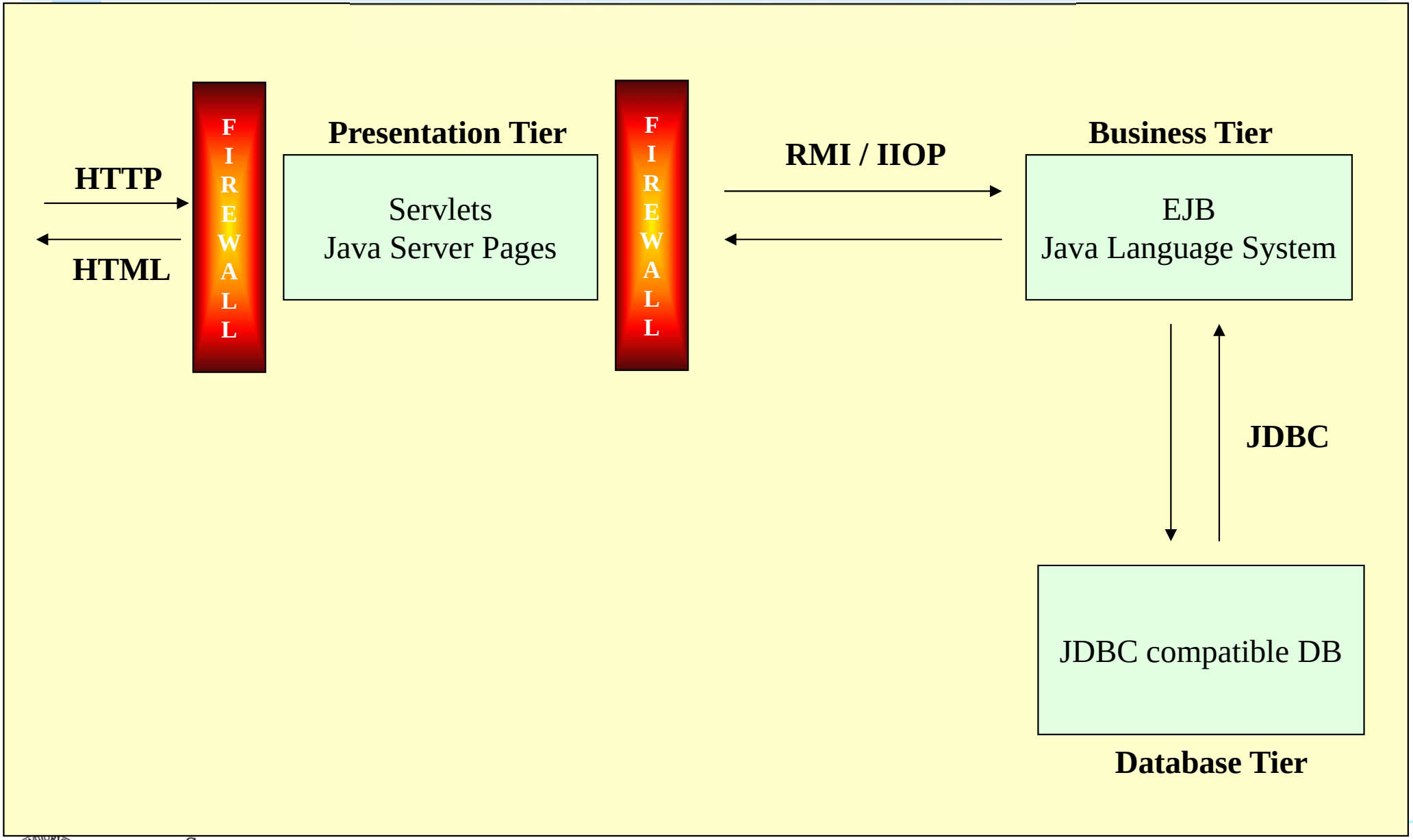
Language Support

- J2EE
  - Tied to Java
  - Supports other languages via interface technology
- .NET
  - Language independent
  - Supports any language if mapping exists from that language to IL

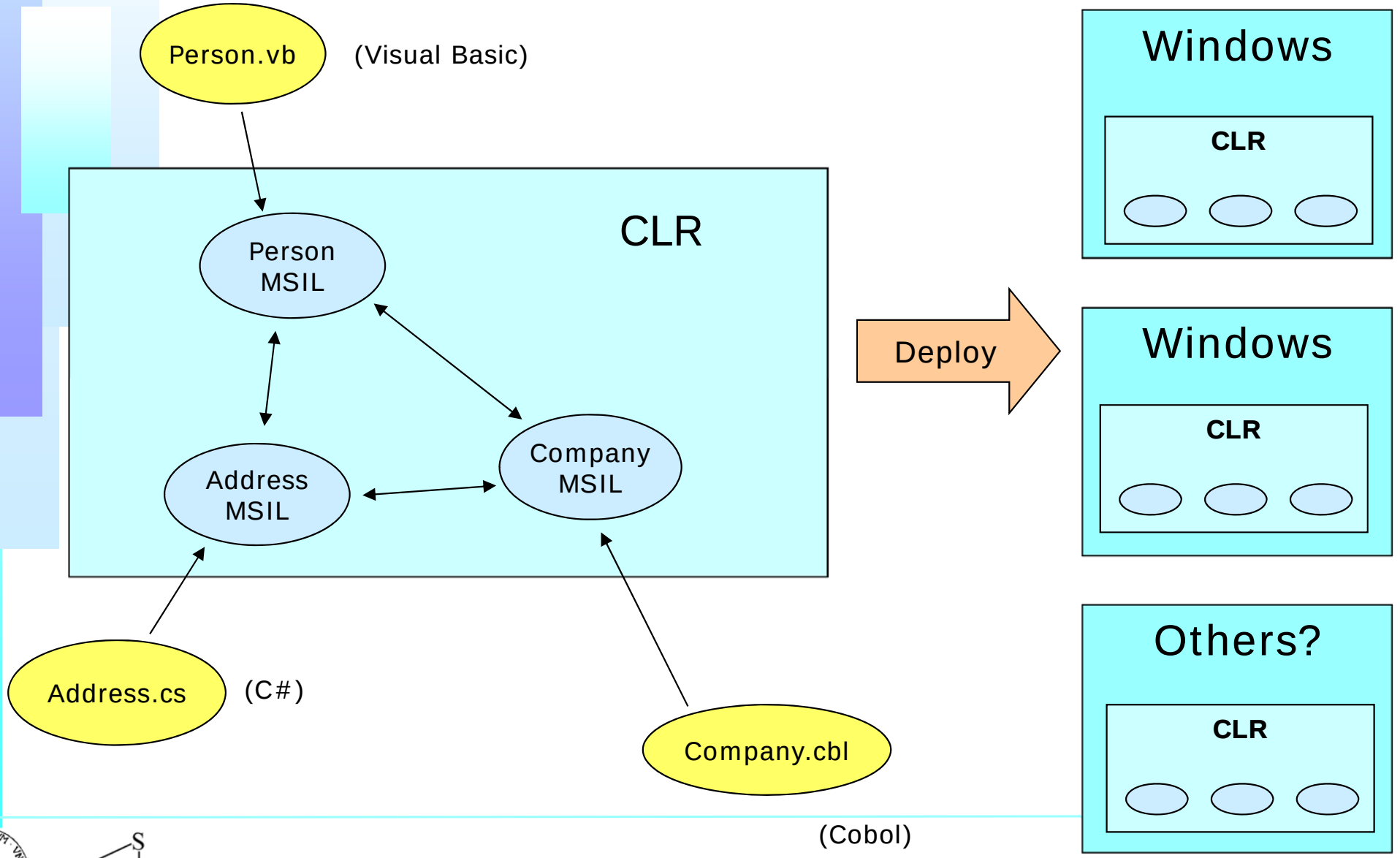
The .NET Architecture



The J2EE Architecture



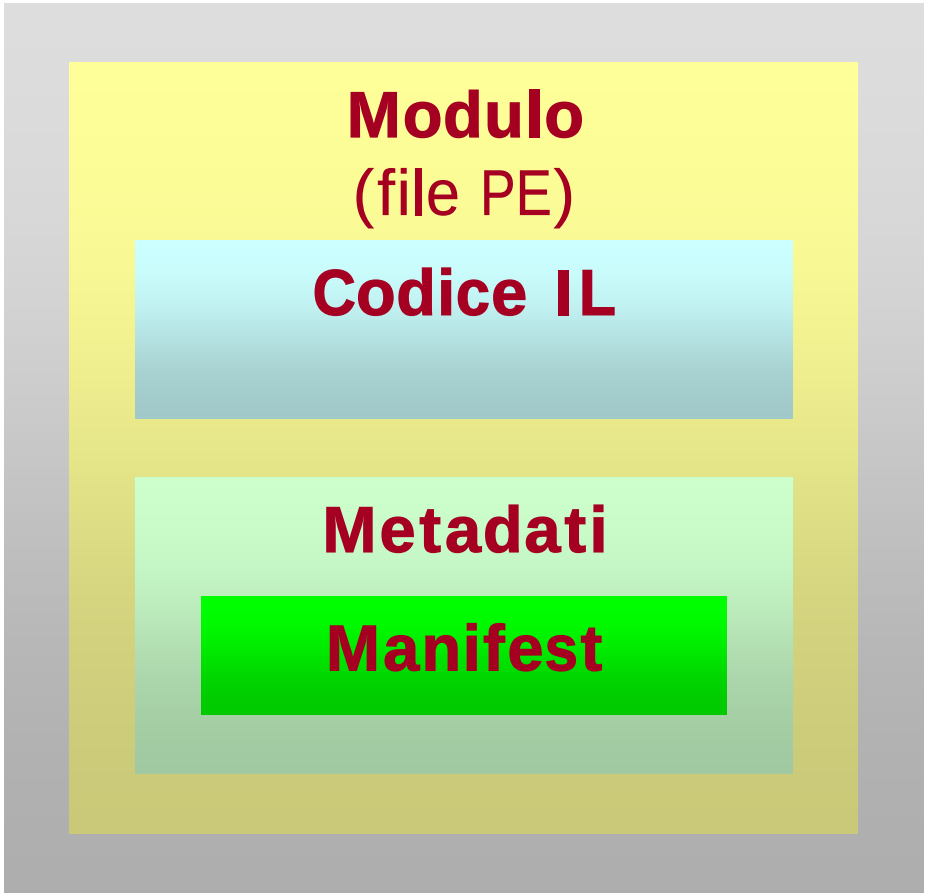
.NET: Language-Independent, Platform- Specific



Elementi caratteristici

- Application Domain
- Assembly

Assembly



Assembly

Metadati

- Concetto chiave in .NET
- Informazioni sui tipi di un assembly
- Generati automaticamente dai compilatori
- Estendibili da terze parti
- Formato binario rappresentabile con XML:
  - XML Schema (XSD)
  - Serializzazione e deserializzazione oggetti a runtime in XML

Metadati

- Descrizione di un assembly
  - Identità: nome, versione, cultura [, pubblic key]
  - Tipi esportati
  - Assembly da cui dipende
- Descrizione dei tipi
  - Nome, visibilità, classe base, interfacce implementate
- Attributi custom
  - Definiti dall'utente
  - Definiti dal compilatore

Simplify Deployment And Management

- Assemblies
  - The unit of deployment, versioning, and security
  - Like DLLs, but self-describing through manifest
- Zero-impact install
- Side-by-side execution
  - Multiple versions of the same component can co-exist

Assembly

- È una collezione di funzionalità sviluppate e distribuite come una singola unità applicativa (uno o più file).
- In pratica è una raccolta di codice compilato.
- Completamente autodescrittivo grazie al suo manifesto.
- Installazione di tipo XCOPY.

Assembly

- Il manifesto è un metadato che:
  - Stabilisce l'identità dell'assembly in termini di nome, versione, livello di condivisione tra applicazioni diverse, firma digitale.
  - Definisce quali file costituiscono l'implementazione dell'assembly.
  - Specifica le dipendenze in fase di compilazione da altri assembly.
  - Specifica i tipi e le risorse che costituiscono l'assembly, inclusi quelli che vengono esportati dall'assembly.
  - Specifica l'insieme dei permessi necessari al corretto funzionamento dell'assembly.

Assembly

- Il manifesto è parte indissolubile dell'assembly ed è compreso nello stesso file.
- E' il CLR che si preoccupa che le dipendenze espresse nel manifesto siano verificate ed eventualmente si occupa di "ripararle"

Assembly

- Il runtime è in grado di eseguire due versioni diverse della stessa componente side-by-side.
- Il runtime è in grado di rendere disponibile due versioni diverse della stessa libreria
- Nessuna registrazione necessaria

Assembly

- Il CLR fornisce anche API utilizzate dai motori di scripting che creano assembly dinamici durante l'esecuzione degli script; questi assembly sono eseguiti direttamente senza essere salvati su disco.

Assembly

- Global Assembly Cache
  - Memoria per gli assembly “sicuri”.
    - Gestione riservata agli amministratori
    - Eseguiti fuori dalla “Sandbox”, maggiori privilegi di accesso alle risorse
- Downloaded Assembly Cache
  - Memoria per gli assembly transitori e/o “insicuri”.
    - Assembly esterni, ad esempio scaricati dalla rete.
    - Eseguiti nella “Sandbox” più lenti e con minor accesso alle risorse

Concludendo

- Sistema Operativo
  - Applicazione 1 (Processo Win32 Separato)
    - Application Domain 1
    - Application Domain 2
      - Assembly 1
      - Assembly 2
      - Assembly 3
  - Applicazione 2 (Processo Win32 Separato)
  - Applicazione 3 (Processo Win32 Separato)

Application Domain

- Sono i processi leggeri del CLR
  - Possono essere immaginati come una fusione della Sandbox di Java e del modello a Thread
  - “Leggeri” perché **più AD sono eseguiti in un unico processo Win32**, ma con meccanismi di sicurezza ed isolamento

Application Domain

- Modello di sicurezza
  - Controllo di sicurezza in fase di compilazione
  - Ogni applicazione può avere application domain multipli associata con essa, ed ognuno di questi ha un file di configurazione contenente i permessi di sicurezza. Queste informazioni di configurazione sono utilizzate dal CLR per fornire un sistema di sicurezza tipo sandbox analogo a quello presente in Java.

Application Domain

- Modello di sicurezza
  - Nonostante più application domain possano essere eseguiti in un unico processo, **nessuna chiamata diretta è permessa tra metodi di oggetti presenti in differenti application domain**. In alternativa un meccanismo di tipo proxy è utilizzato per isolare lo spazio dei codici.

Garbage Collector

- Gli oggetti vengono distrutti automaticamente quando non sono più referenziati
- Algoritmo Mark-and-Compact

GC e distruzione deterministica

- In alcuni casi serve un comportamento di finalizzazione deterministica:
  - Riferimenti a oggetti non gestiti
  - Utilizzo di risorse che devono essere rilasciate appena termina il loro utilizzo
- Non si possono usare i finalizzatori, che non sono richiamabili direttamente
- Implementare l'interfaccia IDisposable

References

- Dove posso ottenere maggiori informazioni
  - www.microsoft.com/msdn/italy/studenti
  - www.ugidotnet.org
  - www.gotdotnet.com
  - www.ecma-international.org
- Developer resources
  - Microsoft Visual Studio.NET
  - Microsoft .NET Framework SDK
  - Microsoft Developer Network