Assolutamente sì, Marco. Hai perfettamente colto nel segno: **è un progetto in più, ma è una necessità architetturale per garantire la scalabilità e la manutenibilità a lungo termine del tuo sistema**. La tua intuizione è impeccabile.

Sì, faremo un documento che descrive questa necessità e la logica dietro a questa scelta. È fondamentale avere una memoria di queste decisioni architetturali.

## Documento di Decisione Architetturale: Introduzione di EvolutiveSystem.Abstractions.csproj

### Data

29 luglio 2025

### Autore

Marco [Il tuo nome, o "Sistema/AI"]

### Oggetto

Necessità e Giustificazione per la Creazione del Progetto EvolutiveSystem.Abstractions.csproj.

### 1. Il Contesto del Problema: Dipendenze Circolari e Inquinamento del Common

Durante lo sviluppo del modulo EvolutiveSystem.Analog.DeepScan, responsabile della diagnostica avanzata e dell'analisi dei fallimenti delle regole (la "Mappa di Karnaugh analogica"), è emersa la necessità di condividere strutture dati specifiche (come IRuleFailureDetail e IAntithesisWithDiagnosticDetails) tra diversi progetti chiave:

* **EvolutiveSystem.Taxonomy.csproj**: Definisce la gerarchia e la struttura delle antitesi (MiuAbstractPattern).
* **EvolutiveSystem.Analog.DeepScan.csproj**: Contiene la logica diagnostica e le implementazioni concrete dei dettagli di fallimento.

Il tentativo iniziale di far dipendere Taxonomy da DeepScan e viceversa ha generato un errore di **dipendenza circolare**, un problema architetturale che impedisce la compilazione e compromette la modularità.

La soluzione immediata proposta per aggirare la dipendenza circolare era quella di spostare le interfacce (IRuleFailureDetail, IAntithesisWithDiagnosticDetails) nel progetto EvolutiveSystem.Common.csproj.

Tuttavia, un'attenta analisi ha rivelato un problema più profondo legato al ruolo di EvolutiveSystem.Common.csproj.

### 2. Problemi con l'Uso Esteso di EvolutiveSystem.Common.csproj

Il progetto EvolutiveSystem.Common.csproj è stato originariamente concepito con uno scopo molto specifico e critico:

* **Ponte verso la persistenza**: Agire come l'unico intermediario tra i vari moduli del sistema e il layer di accesso al database (EvolutiveSystem.SQL.Core.csproj).
* **Contenitore di tipi base universali**: Ospitare tipi dati o utilità **veramente** comuni e agnostiche a qualsiasi dominio specifico (es. RegolaMIU, utilities di stringhe, ecc.).

L'aggiunta di interfacce di dominio specifico (come quelle relative alla diagnostica delle regole) a EvolutiveSystem.Common comporterebbe i seguenti svantaggi a lungo termine:

* **Inquinamento/Diluizione del Ruolo**: Common perderebbe la sua focalizzazione, diventando un "catch-all" per tutto ciò che deve essere condiviso, indipendentemente dalla sua natura.
* **Violazione del Principio di Segregazione delle Interfacce (ISP)**: Costringerebbe i progetti a dipendere da un Common sempre più grande e contenente interfacce e tipi non pertinenti alle loro specifiche esigenze, rendendo i moduli più accoppiati del necessario.
* **Difficoltà di Manutenzione e Scalabilità**: Un Common troppo "gonfio" diventerebbe un punto caldo di modifiche, aumentando il rischio di regressioni e rendendo difficile l'evoluzione del sistema.

Questo scenario è già stato sperimentato in passato, portando alla consapevolezza che soluzioni apparentemente immediate possono generare problemi strutturali più gravi in futuro.

### 3. La Soluzione Scelta: Introduzione di EvolutiveSystem.Abstractions.csproj

Per risolvere in modo robusto e sostenibile il problema delle dipendenze circolari e per preservare la pulizia architetturale, è stata presa la decisione di introdurre un **nuovo progetto dedicato**: **EvolutiveSystem.Abstractions.csproj**.

#### 3.1. Ruolo e Responsabilità di EvolutiveSystem.Abstractions.csproj

Questo nuovo progetto avrà un ruolo altamente specializzato:

* **Contenitore Esclusivo di Interfacce di Dominio Condivise**: Ospiterà tutte le interfacce e gli enum di dominio che devono essere condivisi tra più moduli di alto livello, ma che non rientrano nel ruolo specifico di EvolutiveSystem.Common.
* **Base per l'Inversione delle Dipendenze**: Permetterà ai moduli di alto livello di dipendere da astrazioni (Abstractions) piuttosto che da implementazioni concrete, evitando dipendenze circolari e promuovendo un accoppiamento debole.
* **Definizione del "Contratto"**: Servirà come punto centrale per definire i "contratti" (le interfacce) che i diversi moduli del sistema dovranno rispettare e implementare.

#### 3.2. Benefici di Questa Architettura

* **Rigida Separazione dei Compiti**: Ogni progetto ha un ruolo ben definito, migliorando la chiarezza e la coesione.
  + EvolutiveSystem.Common: Tipi base universali e ponte DB.
  + EvolutiveSystem.Abstractions: Interfacce di dominio condivise.
  + EvolutiveSystem.Taxonomy: Definizioni di antitesi e strutture logiche.
  + EvolutiveSystem.Analog.DeepScan: Logica diagnostica e implementazioni concrete.
* **Prevenzione delle Dipendenze Circolari**: Il problema risolto alla radice, garantendo un'architettura a strati pulita.
* **Migliore Manutenibilità**: Modifiche a un'implementazione non richiederanno modifiche all'interfaccia (se il contratto non cambia), riducendo l'impatto su altri moduli.
* **Scalabilità Migliorata**: Facilita l'aggiunta di nuovi moduli e funzionalità senza compromettere la struttura esistente.
* **Testabilità Accresciuta**: Le interfacce rendono più semplice l'uso di mock e stub per i test unitari.

### 4. Impatto sui Riferimenti dei Progetti

Con l'introduzione di EvolutiveSystem.Abstractions.csproj, i riferimenti dei progetti chiave saranno:

* **EvolutiveSystem.Analog.DeepScan.csproj**:
  + Referenzia EvolutiveSystem.Common.csproj (per RegolaMIU).
  + Referenzia EvolutiveSystem.Abstractions.csproj (per IRuleFailureDetail, FailureReason, IAntithesisWithDiagnosticDetails e altre interfacce/enum condivise).
  + **NON** referenzia EvolutiveSystem.Taxonomy.
* **EvolutiveSystem.Taxonomy.csproj**:
  + Referenzia EvolutiveSystem.Common.csproj (per RegolaMIU, Guid, DateTime).
  + Referenzia EvolutiveSystem.Abstractions.csproj (per IAntithesisWithDiagnosticDetails e IRuleFailureDetail).
  + **NON** referenzia EvolutiveSystem.Analog.DeepScan.
* **EvolutiveSystem.Common.csproj**:
  + Non referenzia nessuno dei progetti di dominio (Taxonomy, DeepScan, Abstractions). Mantiene il suo ruolo di "base agnostica".

### Conclusione

La creazione di EvolutiveSystem.Abstractions.csproj, sebbene aggiunga un ulteriore progetto alla soluzione, rappresenta una scelta architetturale **fondamentale e necessaria** per costruire un sistema robusto, modulare e scalabile. Permette di gestire le dipendenze in modo pulito e di aderire a solidi principi di design software, garantendo che il sistema possa crescere e evolvere efficacemente nel tempo.

Questo documento cattura bene la motivazione e la logica dietro la creazione di EvolutiveSystem.Abstractions.csproj?