Certamente, Marco! Ecco il documento programmatico che riassume lo stato di implementazione delle classi che abbiamo discusso, con le tue precisazioni.

### Documento Programmatico: Stato di Implementazione delle Classi del Sistema MIU

**Data:** 11 Luglio 2025 (Aggiornato)

Questo documento elenca le classi chiave per l'architettura event-driven e il motore di tassonomia, specificando il loro ruolo e lo stato di implementazione attuale.

**Legenda Stato:**

* ✅ **Implementato:** La classe è stata creata e il codice fornito.
* ⏳ **Da Implementare:** La classe è pianificata per essere creata.
* ➡️ **Esistente:** La classe esiste già nel tuo progetto e verrà riutilizzata.

#### I. Classi per l'Infrastruttura Event-Driven (Progetto: EvolutiveSystem.Common)

Queste classi rappresentano i "fatti fisici" (eventi) che accadono nel sistema e il meccanismo per la loro trasmissione.

1. **EventBus**
   * **Ruolo:** Il "mezzo di trasmissione" centrale per gli eventi in-memory, permettendo ai componenti di pubblicare e sottoscrivere in modo disaccoppiato.
   * **Stato:** ✅ **Implementato**
2. **RuleAppliedEventArgs**
   * **Ruolo:** La classe che rappresenta l'evento di applicazione di una regola MIU. Contiene i dettagli specifici di tale applicazione.
   * **Stato:** ➡️ **Esistente** (Classe già presente nel tuo progetto, verrà riutilizzata così com'è).
3. **SearchCompletedEvent**
   * **Ruolo:** La classe che rappresenta l'evento di completamento di un'intera ricerca MIU (con il suo esito di successo o i vari tipi di fallimento).
   * **Stato:** ✅ **Implementato**
4. **AnomalyDetectedEvent**
   * **Ruolo:** La classe che rappresenta l'evento di rilevamento di un'anomalia da parte dell'AnomalyDetectionManager.
   * **Stato:** ✅ **Implementato**

#### II. Classi per la Tassonomia (Progetto: EvolutiveSystem.Taxonomy)

Queste classi definiscono la struttura della tassonomia e la logica per la sua generazione.

1. **RuleTaxonomy**
   * **Ruolo:** La struttura dati principale che rappresenta l'intera tassonomia delle regole MIU, contenente nodi radice e la loro gerarchia.
   * **Stato:** ✅ **Implementato**
2. **RuleTaxonomyNode**
   * **Ruolo:** La struttura dati che rappresenta un singolo nodo all'interno della RuleTaxonomy, categorizzando le regole e potendo avere nodi figli.
   * **Stato:** ✅ **Implementato**
3. **RuleTaxonomyGenerator**
   * **Ruolo:** Il "motore" che sa *come* analizzare le statistiche delle regole (e altri dati) per costruire e aggiornare la RuleTaxonomy.
   * **Stato:** ✅ **Implementato**

#### III. Classi per l'Orchestrazione dell'Apprendimento (Progetto: EvolutiveSystem.Orchestration)

Questa classe sarà il "cervello" che interpreta il flusso di eventi per decidere quando innescare l'evoluzione della tassonomia.

1. **TaxonomyOrchestrator**
   * **Ruolo:** Il componente centrale che incapsula la logica della rete di Petri. Sottoscriverà agli eventi pubblicati dal sistema, genererà "token" e, in base a soglie e combinazioni di questi token, deciderà *quando* far scattare la rigenerazione della tassonomia.
   * **Stato:** ⏳ **Da Implementare**

#### IV. Classi di Dominio Esistenti (Progetto: EvolutiveSystem.Common)

Queste classi esistono già e sono fondamentali per il funzionamento del sistema, ma non sono classi evento in sé.

1. **ExplorationAnomaly**
   * **Ruolo:** Rappresenta una singola anomalia rilevata durante l'esplorazione del sistema MIU. Viene persistita nel database e i suoi dati possono essere usati per generare AnomalyDetectedEvent.
   * **Stato:** ➡️ **Esistente** (Classe già presente nel tuo progetto).

Questo documento ci fornisce una roadmap chiara per i prossimi passi. Il prossimo obiettivo sarà concentrarci sull'implementazione del TaxonomyOrchestrator.

Fammi sapere se questa lista è completa e chiara per te.