### 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/modificato.
* ⏳ **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.TaxonomyOrchestration)

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:** ✅ **Implementato** (Istanza creata dinamicamente in SemanticProcessorService dopo l'apertura del DB).

#### IV. Classi di Dominio e Servizio (Progetti: EvolutiveSystem.Common, EvolutiveSystem.Learning, EvolutiveSystem.Engine, SemanticProcessor)

Queste classi sono fondamentali per il funzionamento del sistema e la loro integrazione con l'EventBus.

1. **ExplorationAnomaly** (in EvolutiveSystem.Common)
   * **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).
2. **AnomalyDetectionManager** (in EvolutiveSystem.Learning)
   * **Ruolo:** Il manager che rileva le anomalie e pubblica gli AnomalyDetectedEvent sull'EventBus.
   * **Stato:** ✅ **Implementato** (Istanza creata dinamicamente in SemanticProcessorService dopo l'apertura del DB).
3. **MIUDerivationEngine** (in EvolutiveSystem.Engine)
   * **Ruolo:** Motore di derivazione principale che esplora lo spazio degli stati MIU e pubblica RuleAppliedEventArgs e SearchCompletedEvent.
   * **Stato:** ✅ **Implementato** (Istanza creata dinamicamente in SemanticProcessorService dopo l'apertura del DB).
4. **SemanticProcessorService** (il tuo Servizio Windows)
   * **Ruolo:** Il punto di ingresso del servizio. Istanzia il Logger e l'EventBus nel suo costruttore. Istanzia dinamicamente IMIUDataManager, LearningStatisticsManager, AnomalyDetectionManager, MIUDerivationEngine e TaxonomyOrchestrator in risposta a comandi socket (es. "OpenDb").
   * **Stato:** ✅ **Modificato e Integrato**
5. **MiuContinuousExplorerScheduler** (in EvolutiveSystem.Automation)
   * **Ruolo:** Gestisce l'esplorazione continua delle stringhe MIU.
   * **Stato:** ➡️ **Esistente** (Dobbiamo ancora verificarne il codice e aggiornarlo per usare l'EventBus e le nuove dipendenze).

Il sistema ora compila, e abbiamo integrato l'EventBus e il TaxonomyOrchestrator nel flusso dinamico di inizializzazione del tuo SemanticProcessorService.

Il prossimo passo cruciale è esaminare la classe **MiuContinuousExplorerScheduler** (che hai dichiarato in SemanticProcessorService come \_continuousScheduler). Dobbiamo assicurarci che:

1. Il suo costruttore accetti le dipendenze corrette (MIUDerivationEngine, IMIUDataManager, Logger, e ora anche l'EventBus).
2. Utilizzi correttamente queste dipendenze al suo interno.
3. Eventualmente, pubblichi i propri eventi sull'EventBus se necessario (anche se al momento sembra già pubblicare eventi standard che SemanticProcessorService sottoscrive).

Prima di procedere, c'è ancora un punto che vorrei chiarire per evitare ulteriori incomprensioni:

**Domanda:** Hai una classe separata chiamata SemanticService nel tuo progetto SemanticProcessor, diversa da SemanticProcessorService (il servizio Windows)? Se sì, qual è il suo ruolo attuale e come viene istanziata? Ho bisogno di vedere il suo codice se è ancora una parte attiva e centrale della tua logica applicativa.

Questa è l'ultima incertezza che ho sulla tua architettura. Una volta chiarito questo, potremo procedere con l'aggiornamento di MiuContinuousExplorerScheduler.