Università degli Studi di Firenze

Dipartimento di Ingegneria dell'Informazione
Tesi di Laurea Triennale in Ingegneria Informatica

Analisi e Sviluppo di un Componente Java per la Simulazione Interattiva di Reti di Petri Stocastiche

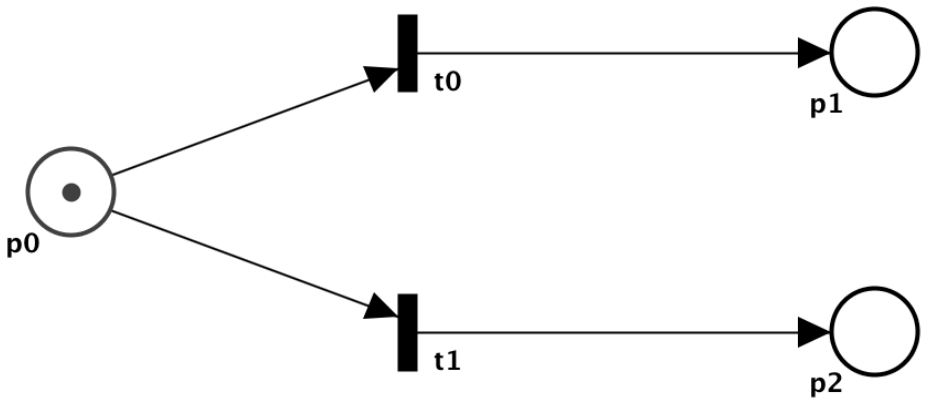
Candidato
Tommaso Scarlatti

Relatore
Prof. Enrico Vicario

Correlatore
Ing. Marco Biagi

Anno Accademico 2016-2017

- Analisi e sviluppo di un simulatore interattivo di reti di Petri stocastiche
- Componente Java integrato nel tool ORIS



Rete di Petri

- Formalismo grafico/matematico per la modellazione di sistemi dinamici ad eventi discreti
- E' un grafo bipartito: posti e transizioni

Aree di applicazione

- Sviluppo di sistemi concorrenti
- Business Process Modeling
- ...

Semantica del formalismo

- **Marking:** distribuzione dei token nei posti
- **Firing rule:** regola che governa la rete. Determina quando una transizione è abilitata
- Transizioni abilitate → una transizione scatta → si modifica il marking

Estensioni

Esistono molte estensioni del formalismo base posto transizione (P/T):

- Reti di Petri Temporizzate
- Reti di Petri Stocastiche
- ...

place

transition

arc

Analisi dei requisiti

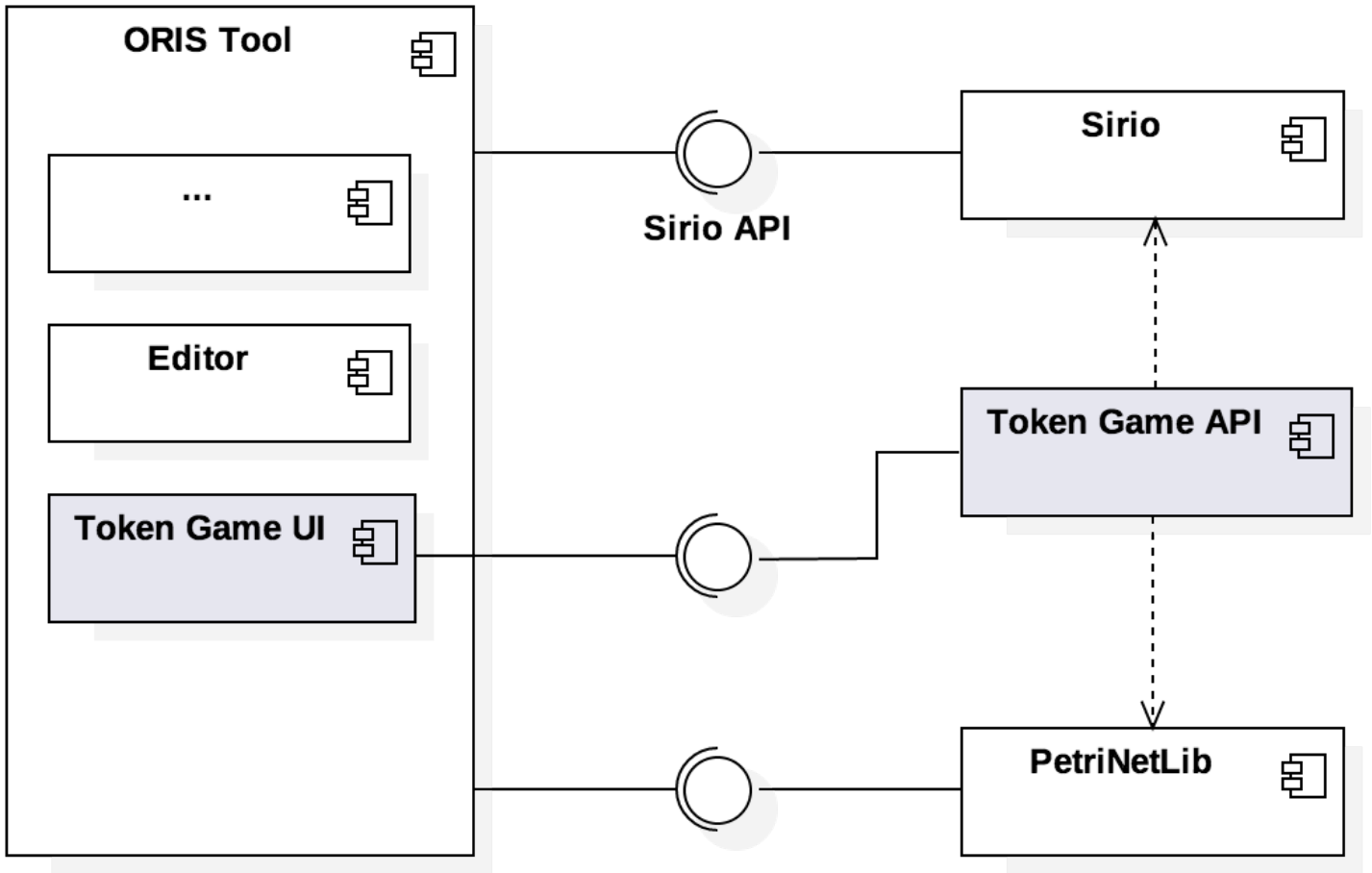
Requisiti funzionali

Sviluppare un **token game**, ossia un componente Java per la simulazione interattiva di reti di Petri stocastiche

Requisiti architetturali

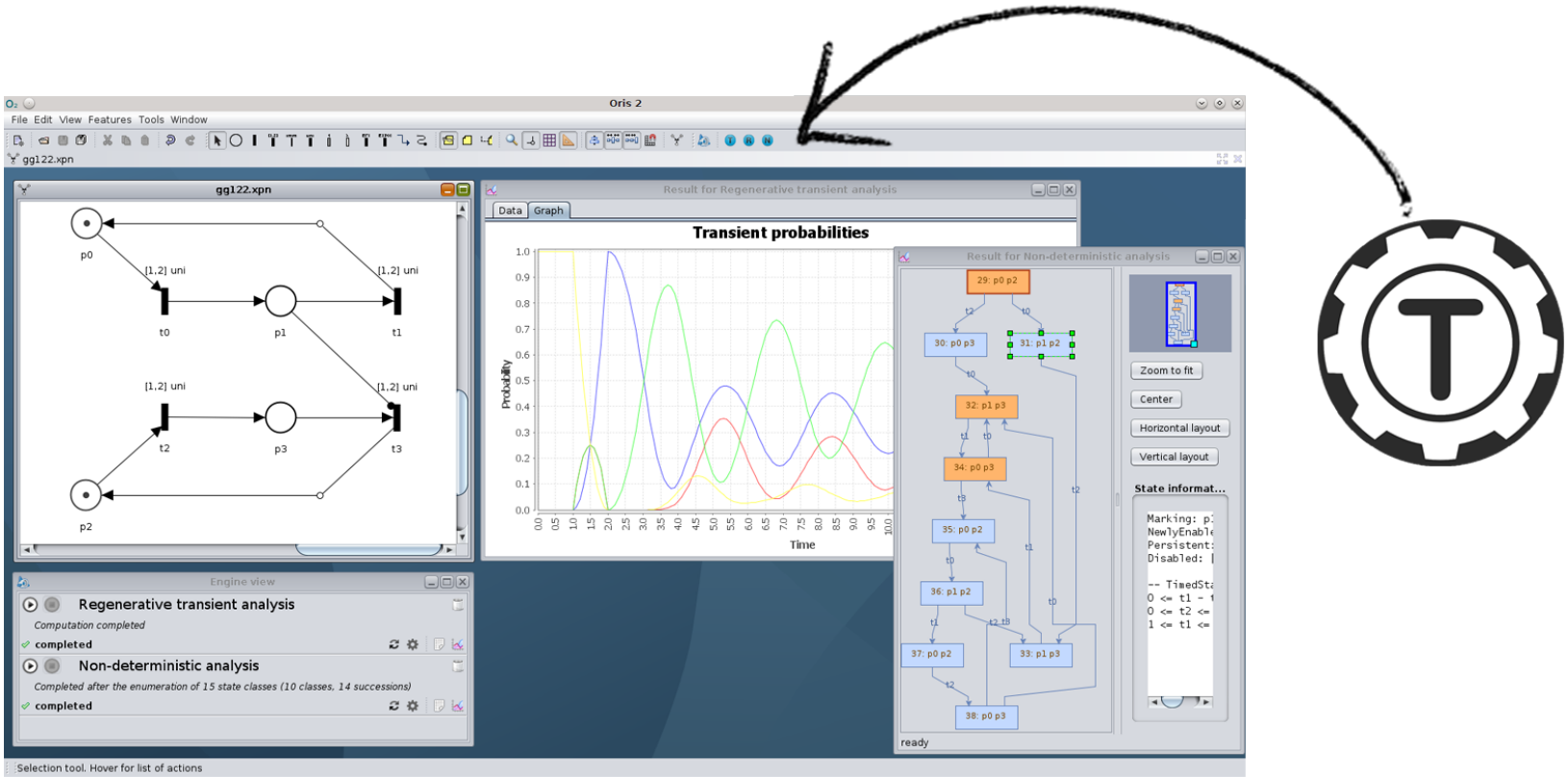
Integrare il componente all'interno del **tool ORIS** (*www.oristool.org*)

- **ORIS** è un tool sviluppato dal *Software Technologies Lab* (STLAB) che permette la modellazione e l'analisi di sistemi reattivi temporizzati basati su varie classi di reti di Petri.



Interfaccia di ORIS

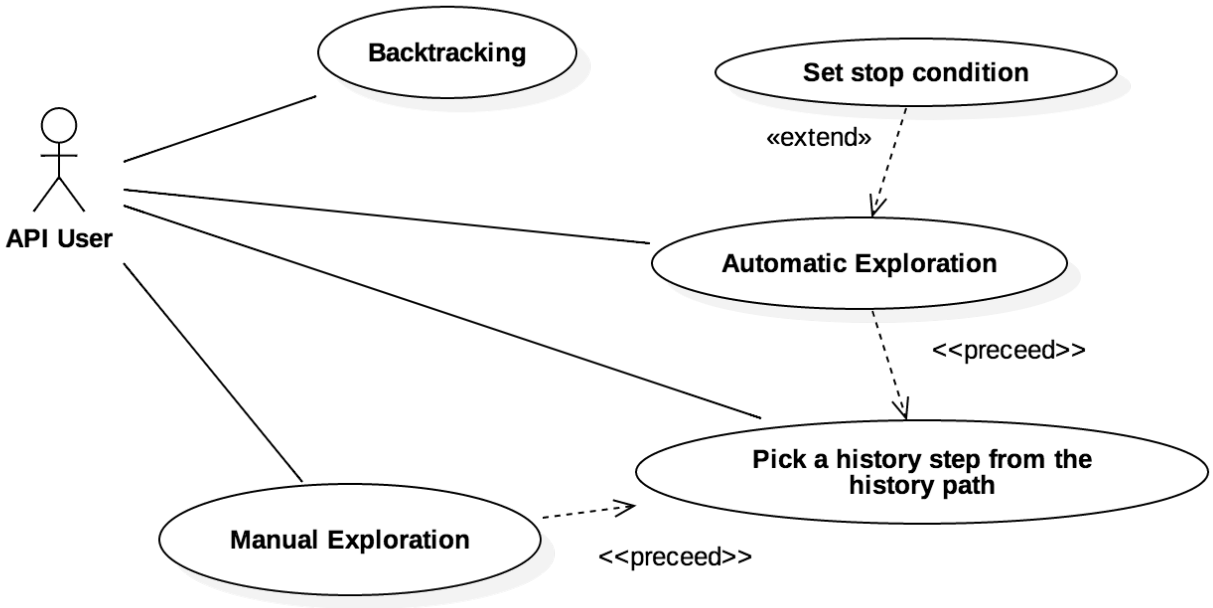
- Due viste: Editor View, Engine View
- **Token Game View** integrata nel tool Oris



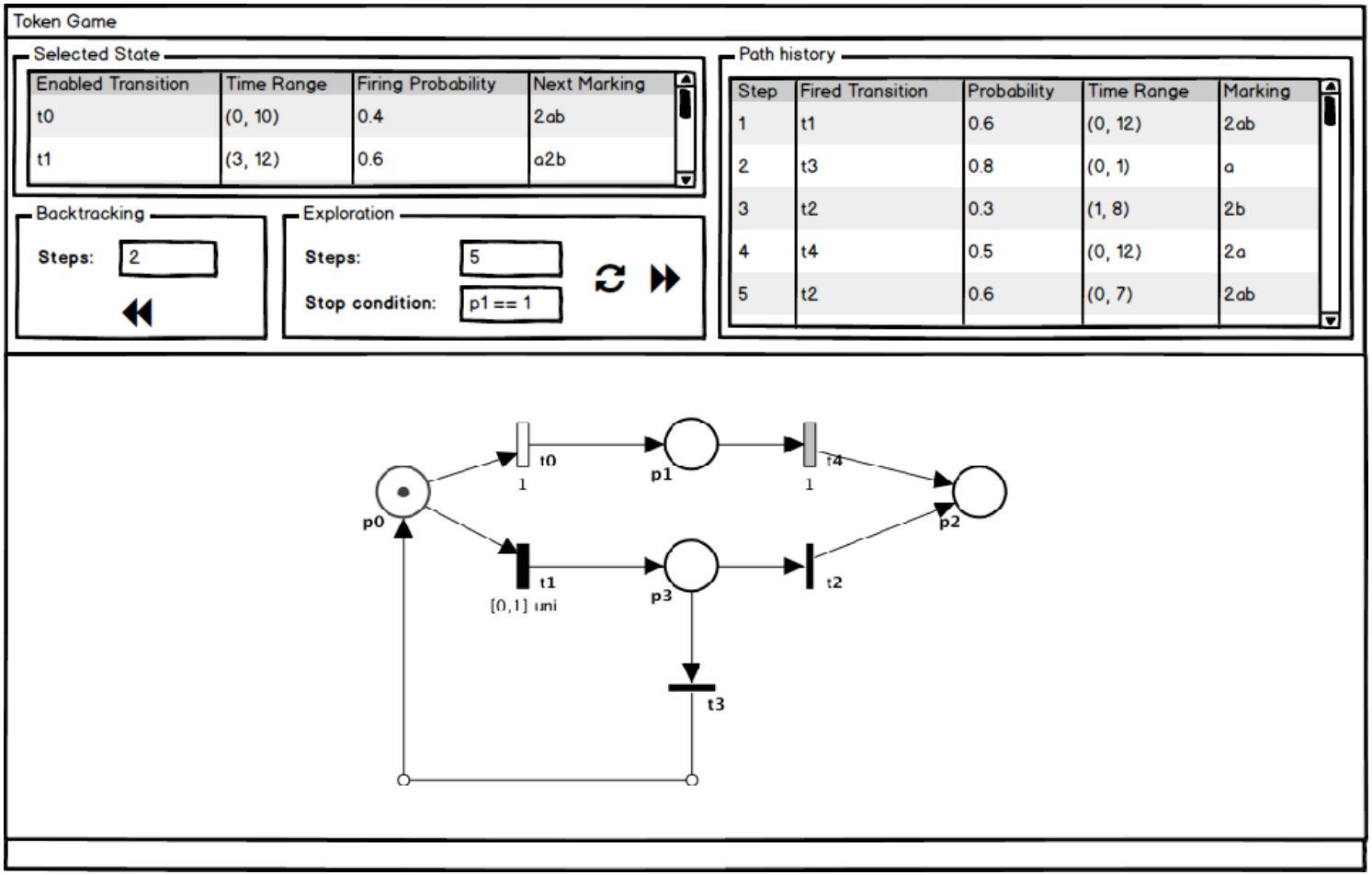
Metodologia di sviluppo

- Studio di fattibilità: tempo, conoscenze da integrare
- Separare la logica di dominio del componente dalla sua rappresentazione grafica:
  1. API (Application Program Interface)
  2. GUI (Graphic User Interface)
- Test di unità e di integrazione (JUnit)

API Use Case Diagram

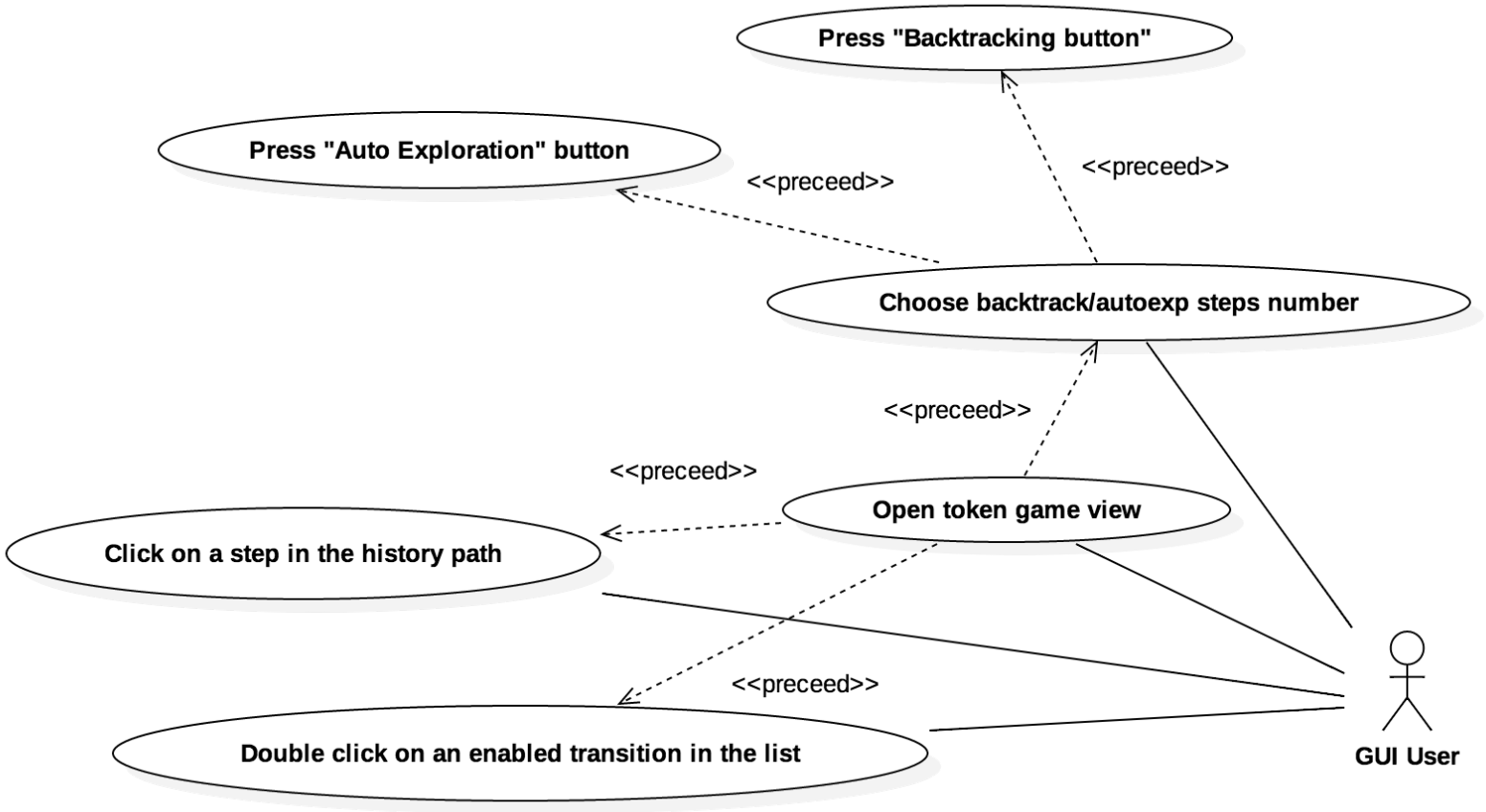


- Esplorazione Manuale/Automatica
- Backtracking
- Selezione di un passo del cammino

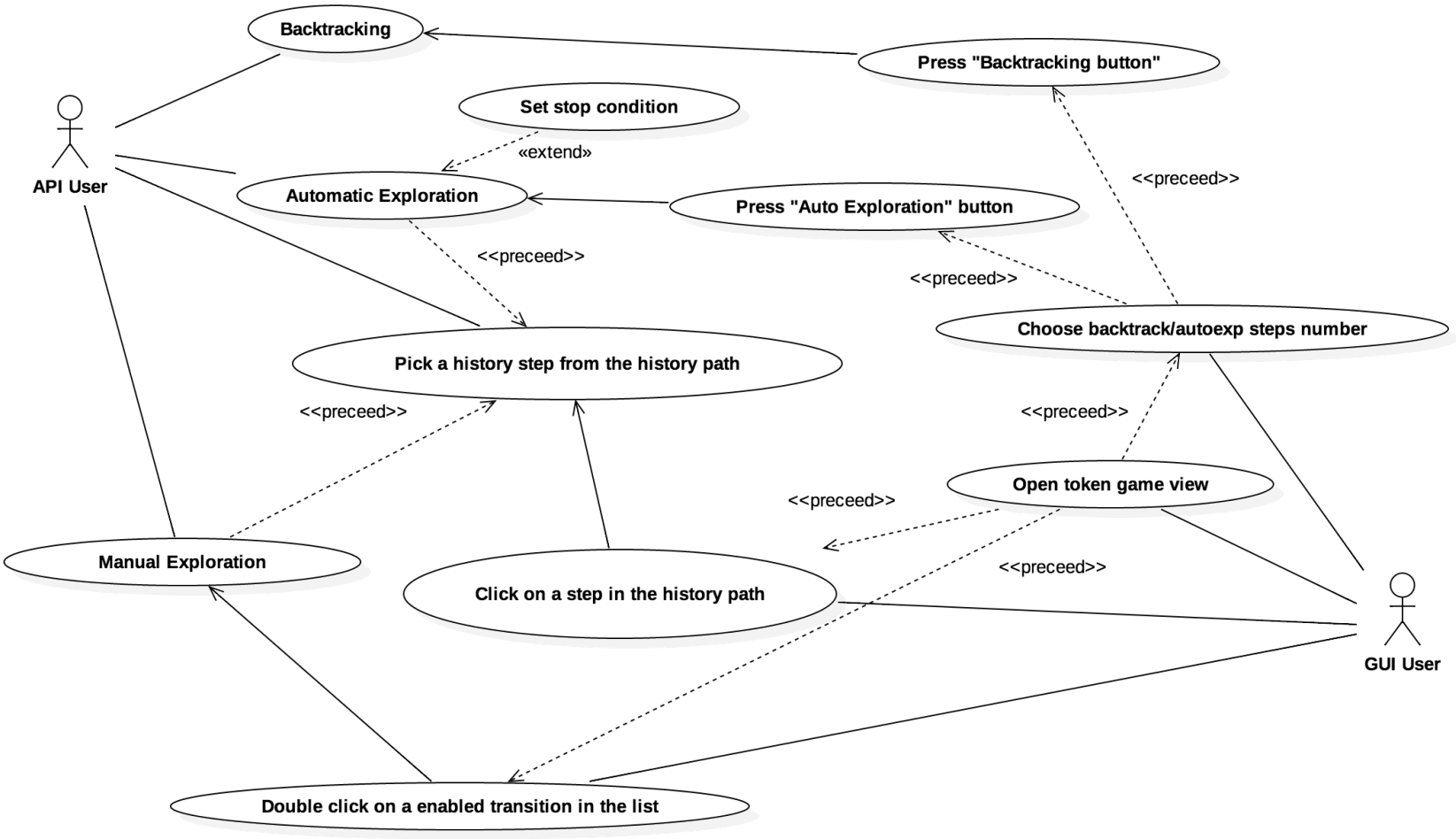


- Exploration
- Path History
- Petri Net

GUI Use Case Diagram

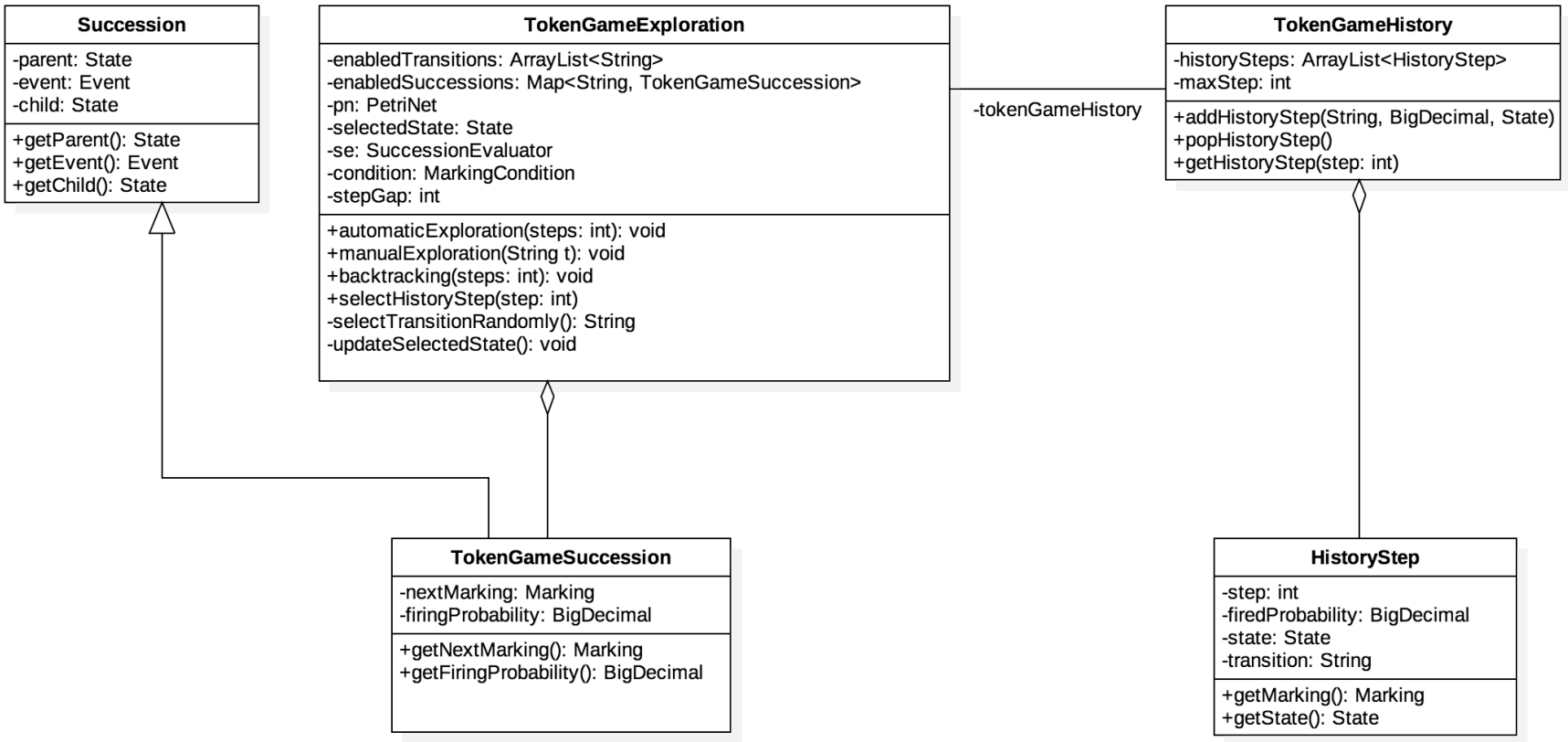


Tying pieces togheter..



Metodologia

- Implementare tutti i requisiti emersi nella fase di analisi
- Class diagram per catturare l'organizzazione delle classi mediante l'allocazione delle responsabilità tra esse

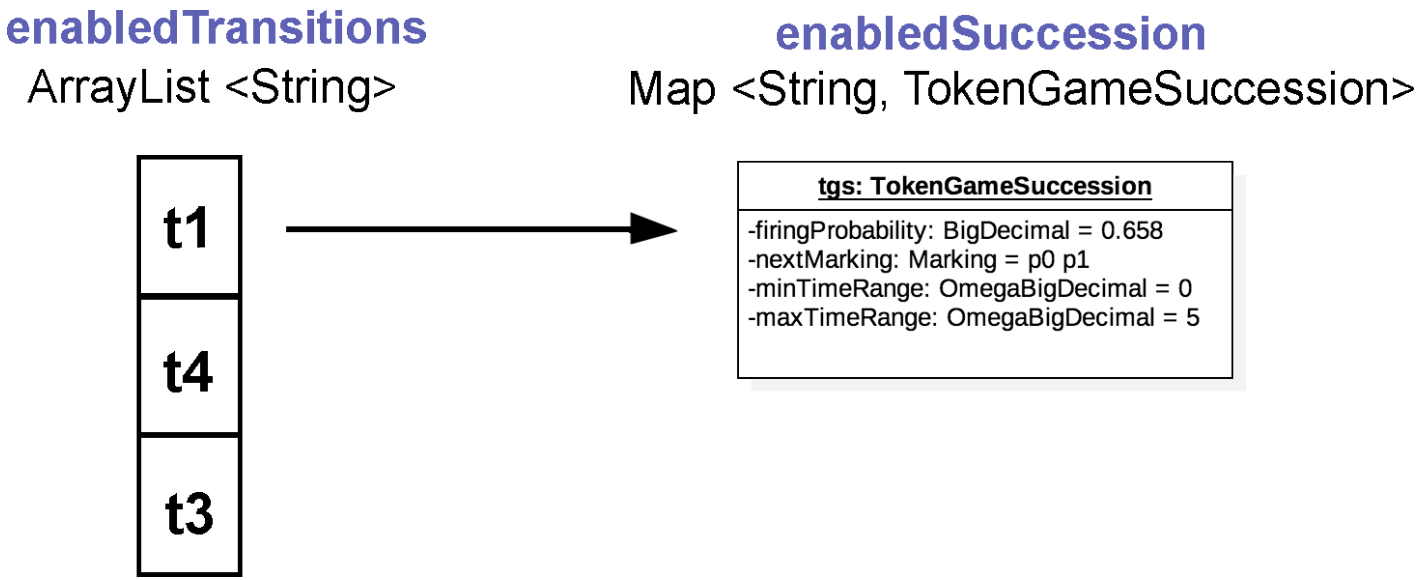


TokenGameExploration

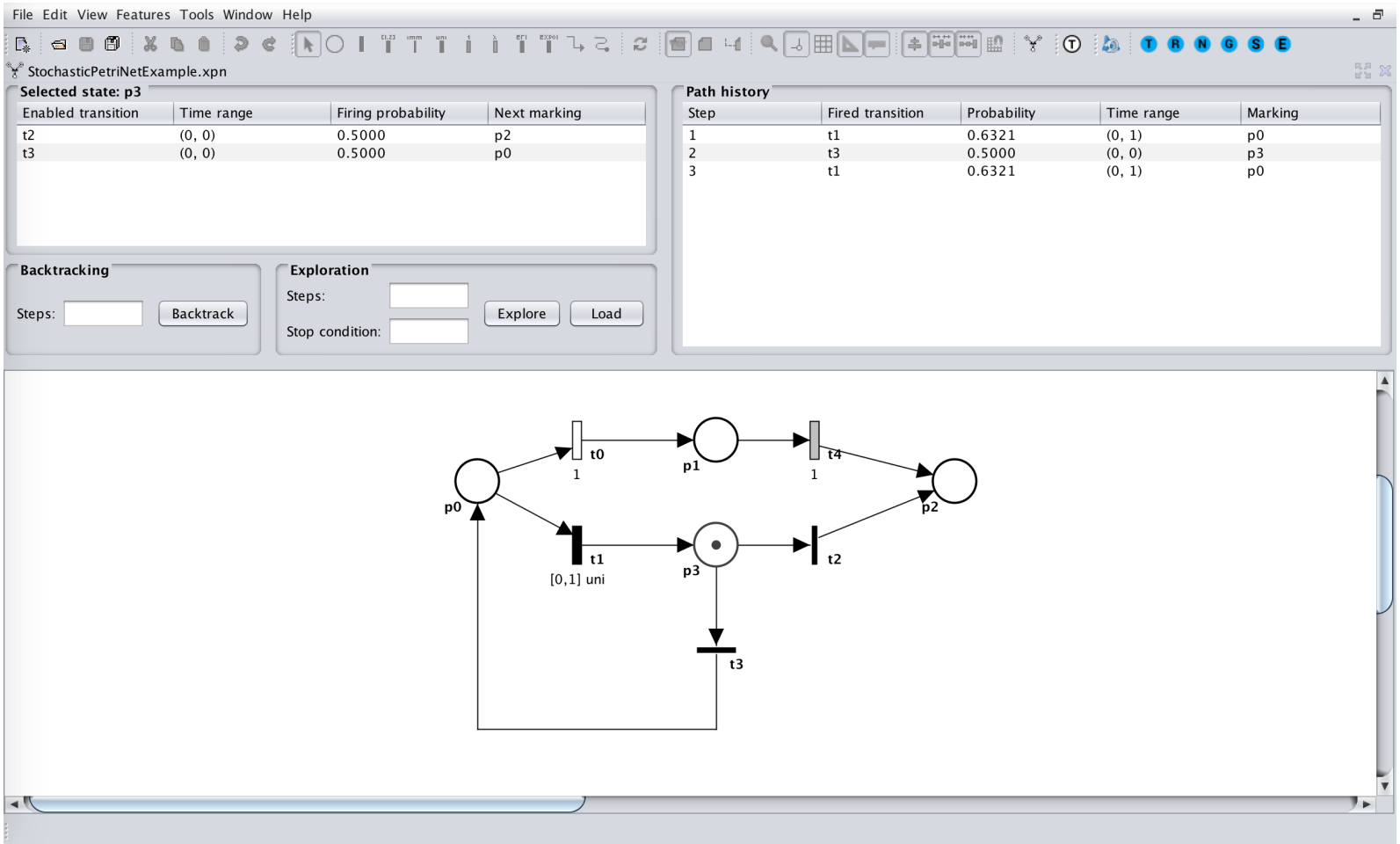
- Classe centrale del progetto che espone metodi pubblici che soddisfano i requisiti funzionali

TokenGameHistory

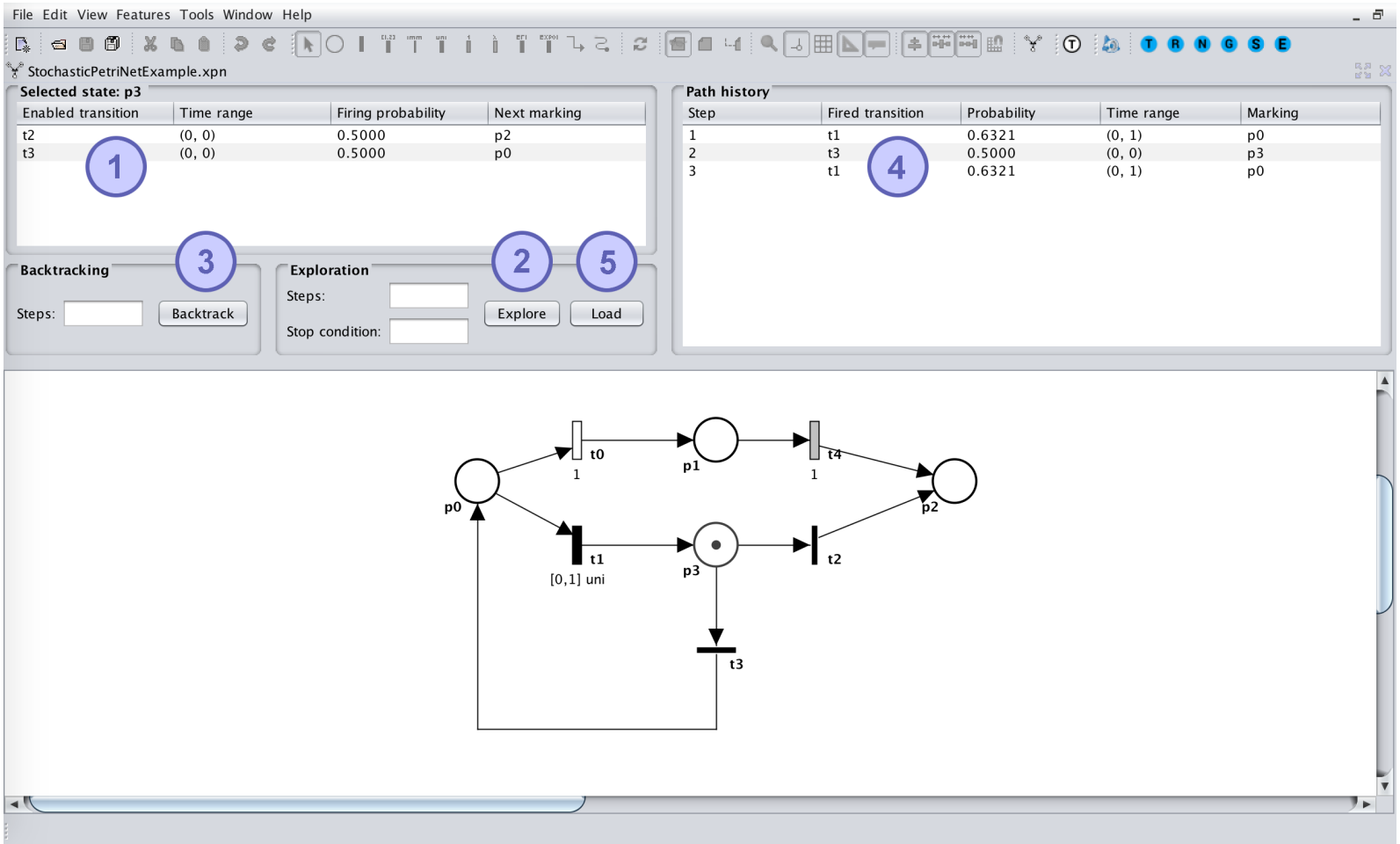
- Rappresenta il container dei passi del cammino di esplorazione
- Modellata come una pila a stack (LIFO)



- Java Swing + Window Builder
- *Listener* in ascolto degli eventi generati dall'utente
- Cinque possibili interazioni



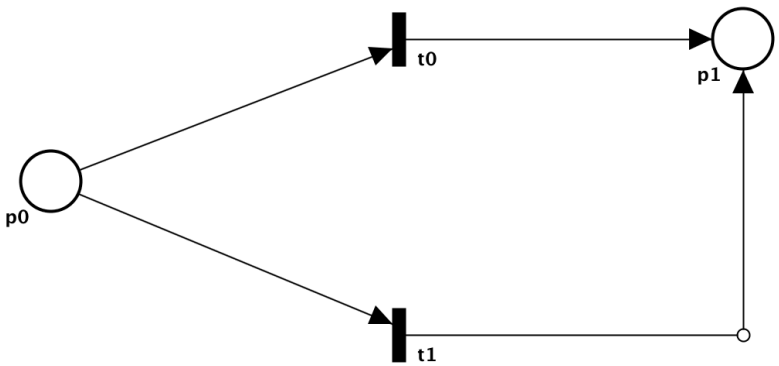
- Java Swing + Window Builder
- *Listener* in ascolto degli eventi generati dall'utente
- Cinque possibili interazioni

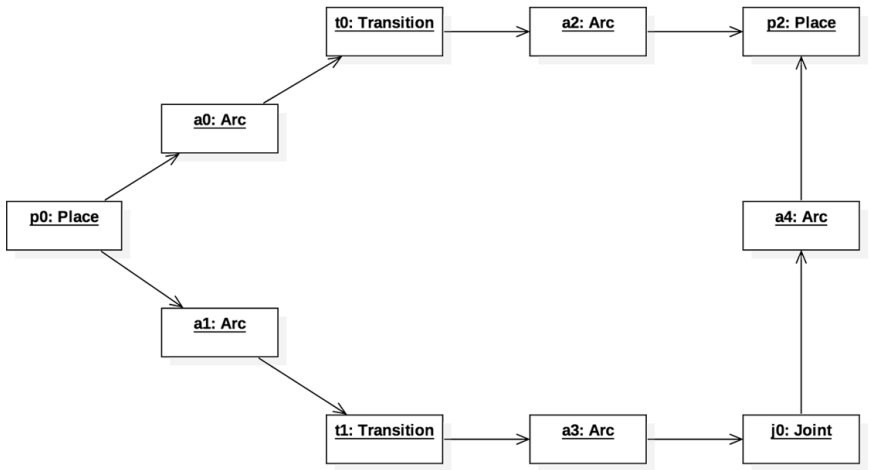


Editor View VS Token Game View

- Permettere alle reti delle due viste di evolvere in maniera indipendente l'una dall'altra
- **Deep copy** delle entità che formano la rete di Petri
  - *Shallow copy:* copia della struttura dati
  - *Deep copy:* copia dei singoli elementi della struttura

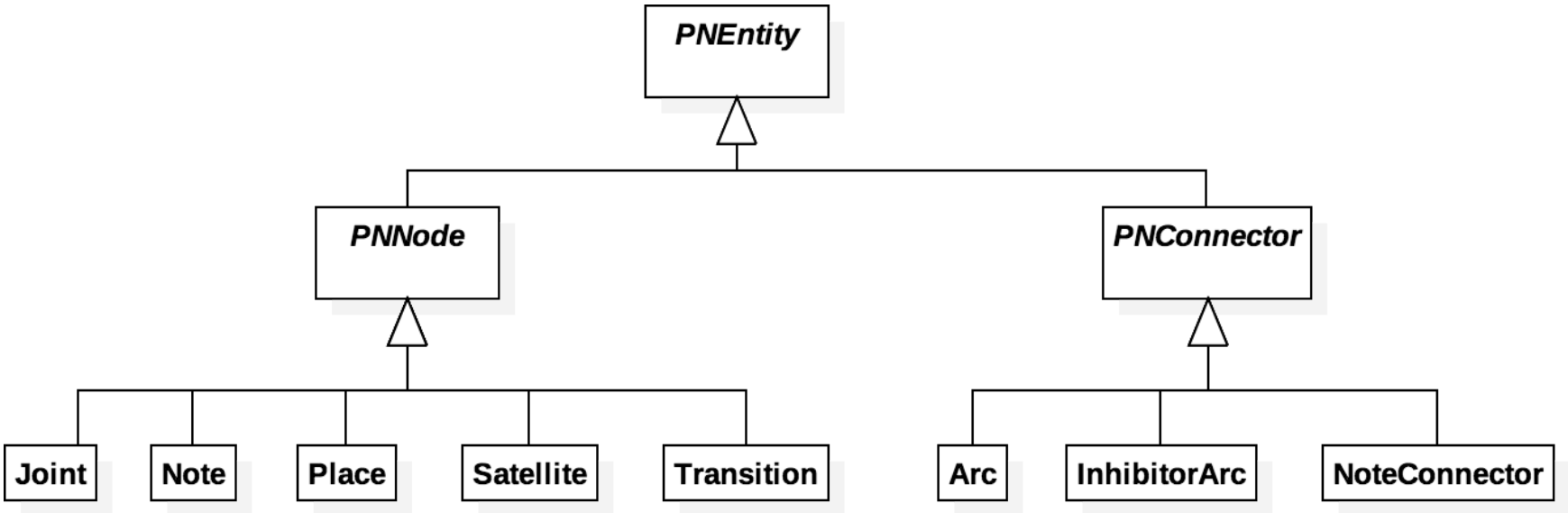
```
private Map<String, PNEntity> entities = new LinkedHashMap<>();
```





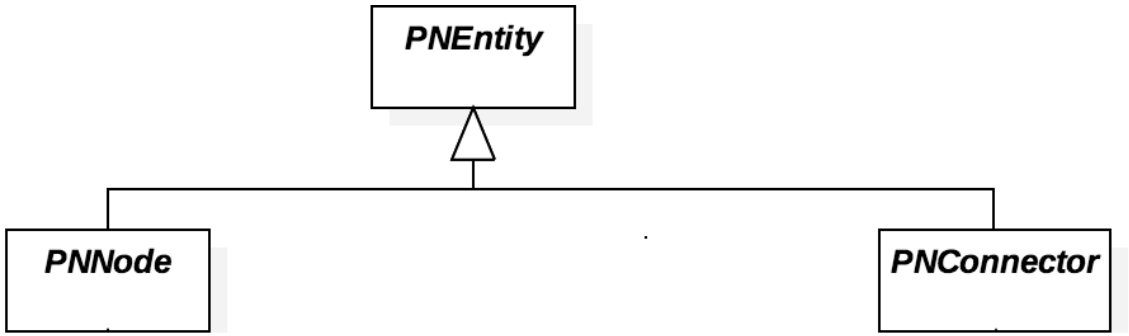
Problema

- Si ha a disposizione una collezione di **entità astratte**.
  Non si può sapere a priori il tipo concreto
- Le entità formano complessivamente un **grafo orientato**.
  Non se ne ha una rappresentazione esplicita



Problema

- Si ha a disposizione una collezione di **entità astratte**.
  Non si può sapere a priori il tipo concreto
- Le entità formano complessivamente un **grafo orientato**.
  Non se ne ha una rappresentazione esplicita

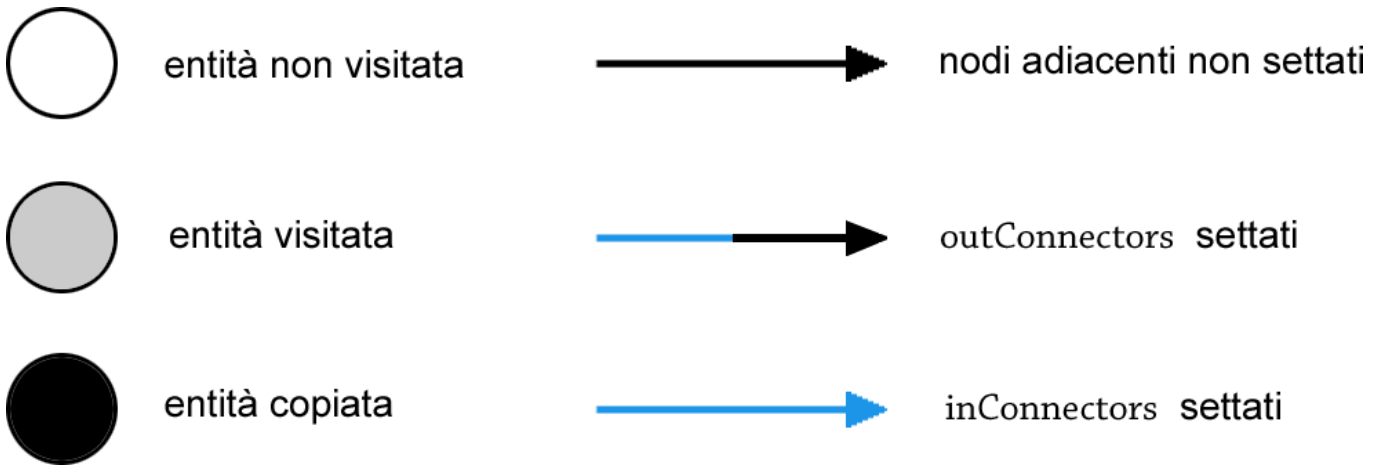



```
– Set<PNConnector> inConnectors = new HashSet<>()

– Set<PNConnector> outConnectors = new HashSet<>()
```

```
– PNNode fromNode

– PNNode toNode
```

Soluzione

- Utilizzo del pattern **Visitor** per separare l'operazione di copia dalla struttura dati
- Utilizzare una **DFS** (Depth First Search) per attraversare il grafo chiamando ricorsivamente l'operazione di visita sui connettori di uscita/nodi terminali



Riepilogo

- Analisi e sviluppo di un **simulatore interattivo** per reti di Petri stocastiche
  - API
  - GUI
- Integrato nel **tool ORIS**:
  - 25 classes
  - 6 different packages
  - 2613 code lines
- Editor View VS Token Game View:
  - Visitor (Deep copy)
  - DFS