

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

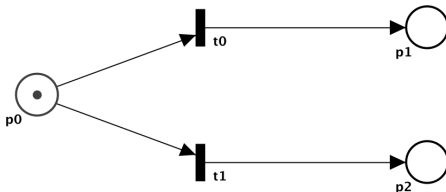


Relatore
Prof. Enrico Vicario

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

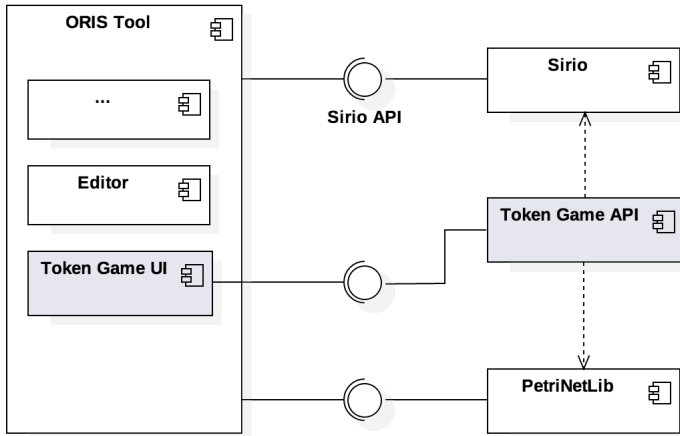
Requisiti funzionali

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

Requisiti architetturali

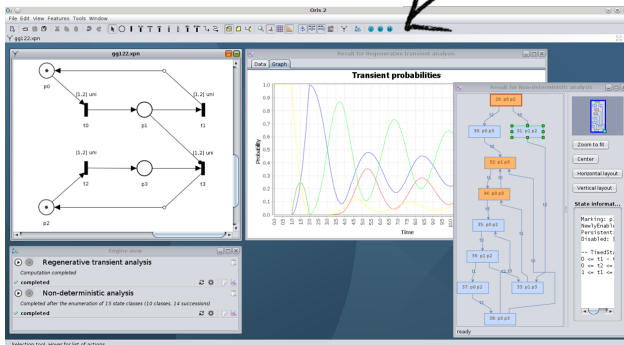
Integrare il componente all'interno del **tool ORIS**
(*www.oristool.org*)

- **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

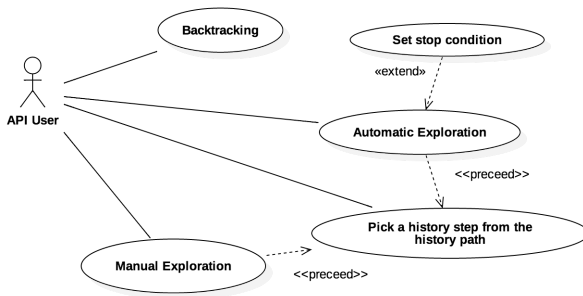
- ▶ Due viste: Editor View, Engine View
- ▶ **Token Game View** integrata nel tool Oris



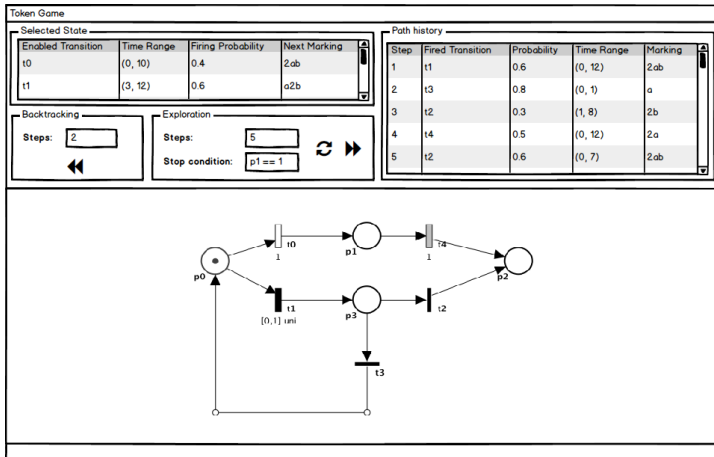
Metodologia di sviluppo

- ▶ 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

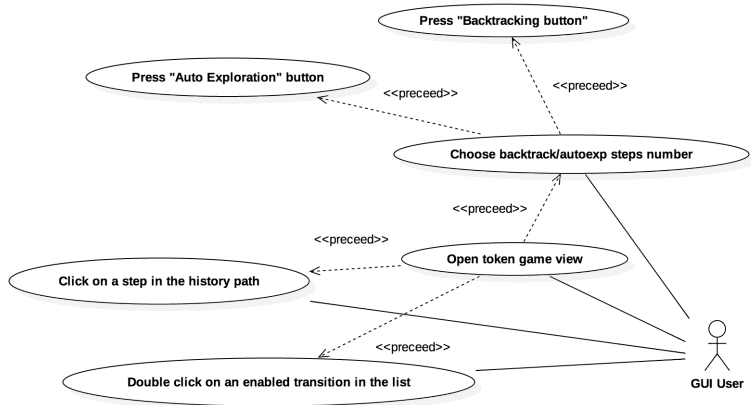


- ▶ Esplorazione Manuale/Automatica
- ▶ Backtracking
- ▶ Selezione di un passo del cammino

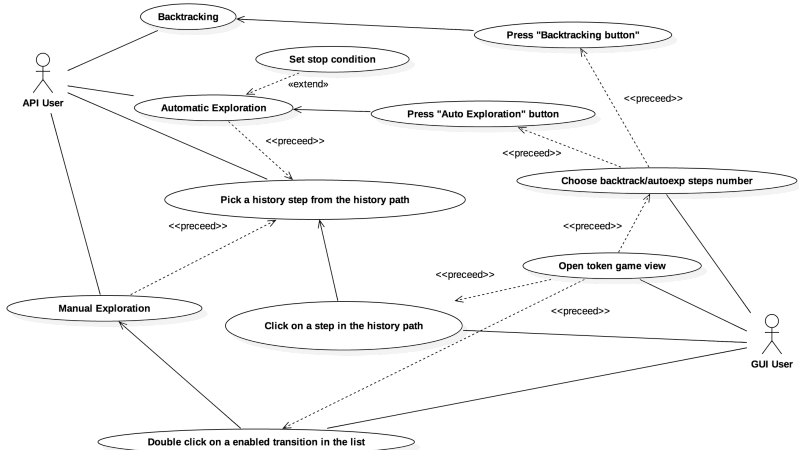


- Exploration
- Path History
- Petri Net

GUI Use Case Diagram

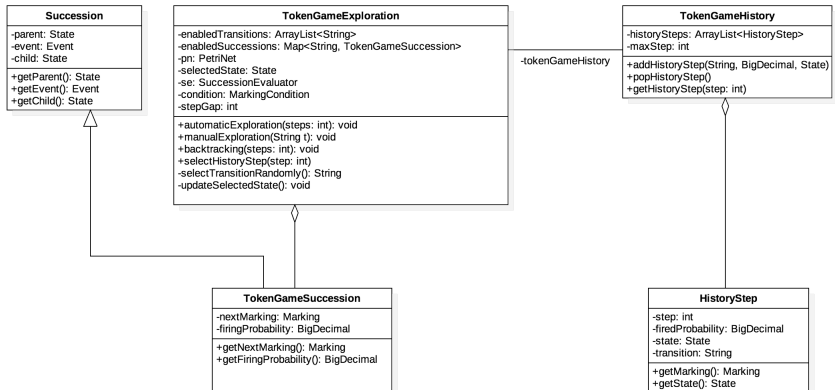


Tying pieces together..



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

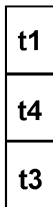
- ▶ Classe centrale del progetto che espone metodi pubblici che soddisfano i requisiti funzionali

TokenGameHistory

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

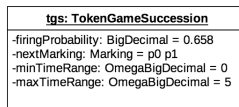
enabledTransitions

ArrayList <String>

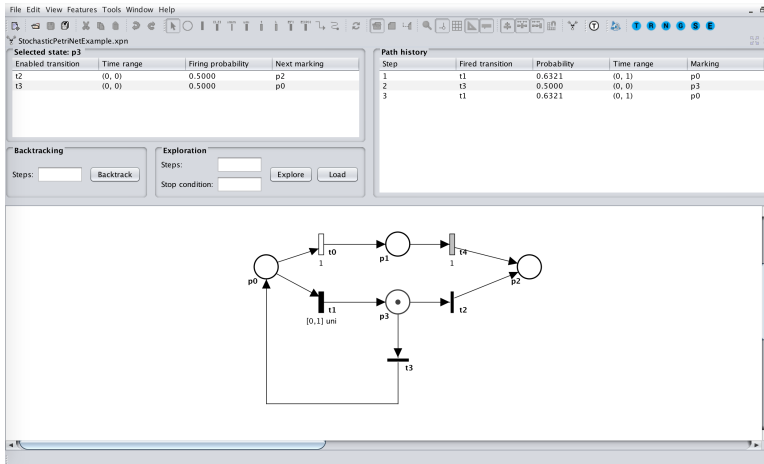


enabledSuccession

Map <String, TokenGameSuccession>



- ▶ 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

The screenshot shows the Stochastic Petri Net (SPN) tool interface. The main window displays a Petri net diagram with places p_0 , p_1 , p_2 , and p_3 , and transitions t_0 , t_1 , t_2 , and t_3 . The diagram includes a self-loop on p_0 labeled t_0 , a transition t_1 from p_0 to p_1 with a uniform distribution $[0,1]$ uni, a transition t_2 from p_1 to p_2 , and a transition t_3 from p_2 back to p_0 .

The interface includes several panels and controls:

- Selected state: p_3** (labeled 1): A table showing enabled transitions, time ranges, firing probabilities, and next markings.
- Backtracking** (labeled 3): A panel with a "Steps:" input field and a "Backtrack" button.
- Exploration** (labeled 2 and 5): A panel with a "Steps:" input field, a "Stop condition:" input field, and "Explore" and "Load" buttons.
- Path history** (labeled 4): A table showing the sequence of steps, fired transitions, probabilities, time ranges, and markings.

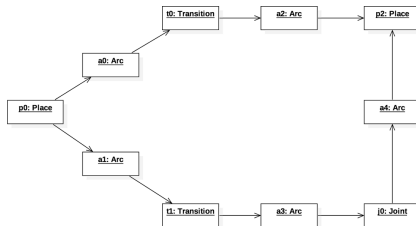
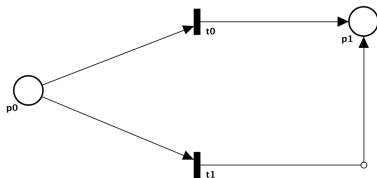
Enabled transition	Time range	Firing probability	Next marking
t_2	(0, 0)	0.5000	p_2
t_3	(0, 0)	0.5000	p_0

Step	Fired transition	Probability	Time range	Marking
1	t_1	0.6321	(0, 1)	p_0
2	t_3	0.5000	(0, 0)	p_3
3	t_1	0.6321	(0, 1)	p_0

Editor View VS Token Game View

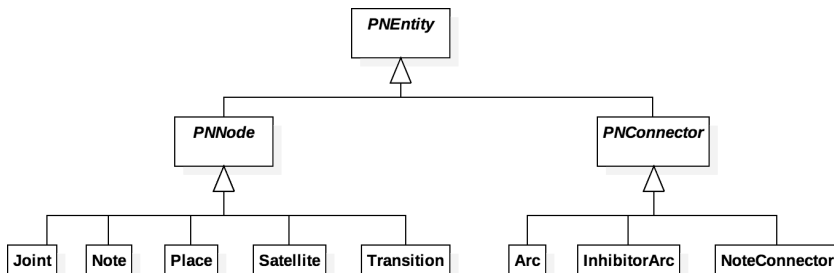
- ▶ Permettere alle reti delle due viste di evolvere in maniera indipendente l'una dall'altra
- ▶ **Deep copy** delle entità che formano la rete di Petri
 - ▶ *Shallow copy*: copia della struttura dati
 - ▶ *Deep copy*: copia dei singoli elementi della struttura

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



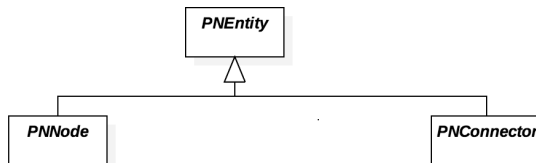
Problema

- ▶ 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<PConnector> inConnectors = new HashSet<>()

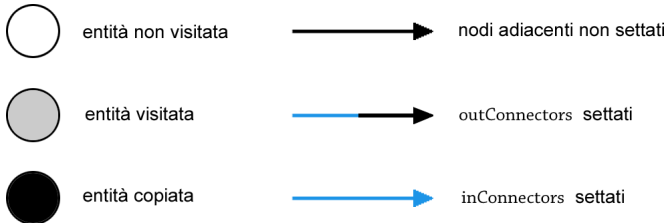
- Set<PConnector> outConnectors = new HashSet<>()
```

```
- PNode fromNode

- PNode 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