Rupos Demo: Process analysis into ProM

Roberto Bruni Andrea Corradini Gianluigi Ferrari
Roberto Guanciale Giorgio Spagnolo

Dipartimento di Informatica, Pisa

RUPOS Demo
22 Marzo 2012

Analisi e verifica dei pattern fondamentali

Obiettivo

- Integrare in RUPOS strumenti di analisi a runtime di processi
- Analisi basata sul confronto di logs con un modello del processo

Strategia

- Adottare e raffinare metodi formali disponibili (Reti di Petri)
- Integrare ed estendere infrastrutture software esistenti (ProM)
- Work-flow metodologico:
  1. I processi di business sono modellati con diagrammi BPMN
  2. I diagrammi BPMN vengono trasformati in Reti di Petri
  3. I logs di Istanze di Processi vengono analizzati con tecniche disponibili per Reti di Petri, oppurtunamente raffinate
  4. I risultati dell'analisi vengono proiettati sul modello BPMN iniziale

Analisi e verifica dei pattern fondamentali

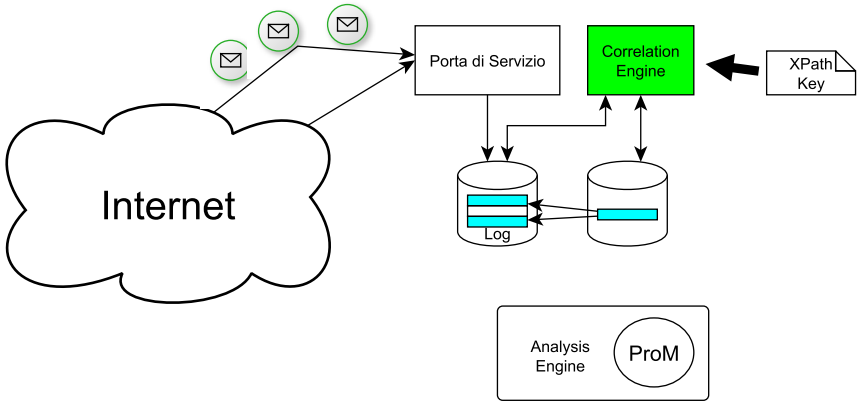
Obiettivo

- Integrare in RUPOS strumenti di analisi a runtime di processi
- Analisi basata sul confronto di logs con un modello del processo

Strategia

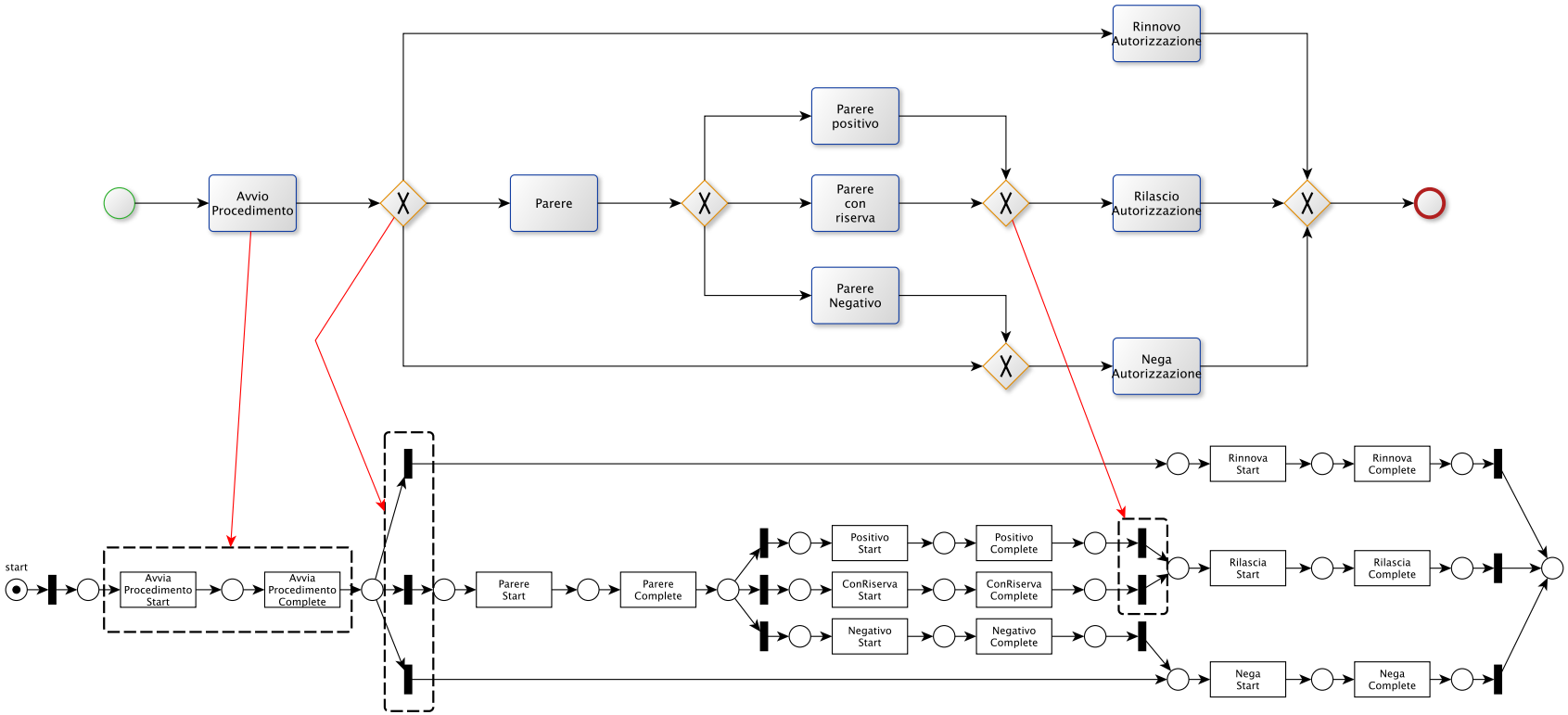
- Adottare e raffinare metodi formali disponibili (Reti di Petri)
- Integrare ed estendere infrastrutture software esistenti (ProM)
- Work-flow metodologico:
  1. I processi di business sono modellati con diagrammi BPMN
  2. I diagrammi BPMN vengono trasformati in Reti di Petri
  3. I logs di Istanze di Processi vengono analizzati con tecniche disponibili per Reti di Petri, oppurtunamente raffinate
  4. I risultati dell'analisi vengono proiettati sul modello BPMN iniziale

Architettura complessiva: integrazione di ProM in RUPOS



- I messaggi SOAP tra attori del processo vengono intercettati dalla Porta di Servizio (PdS)
- La PdS memorizza informazioni essenziali dei messaggi nel log
- La PdS estrae parti dei messaggi memorizzandoli nel database dei log
- Il Motore di Correlazione raggruppa i messaggi in istanze di processi usando "correlation sets" (XPath)
- I log di istanze di processi vengono trasformati in tracce di eventi corrispondenti a transizioni della Rete di Petri

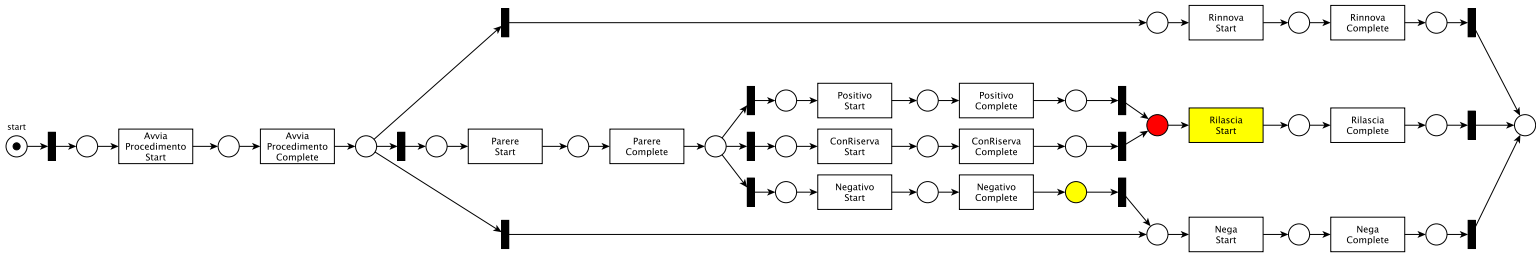
Esempio di processo BPMN e di traduzione in Rete di Petri

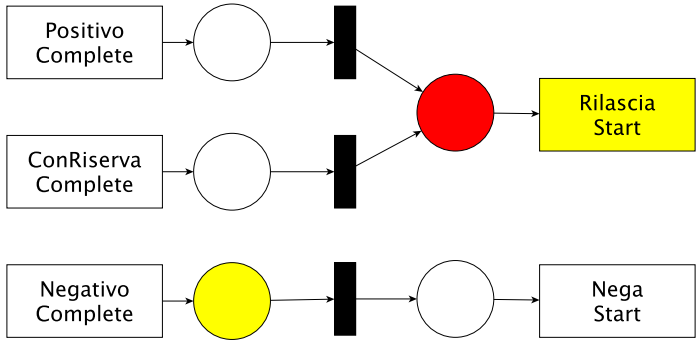


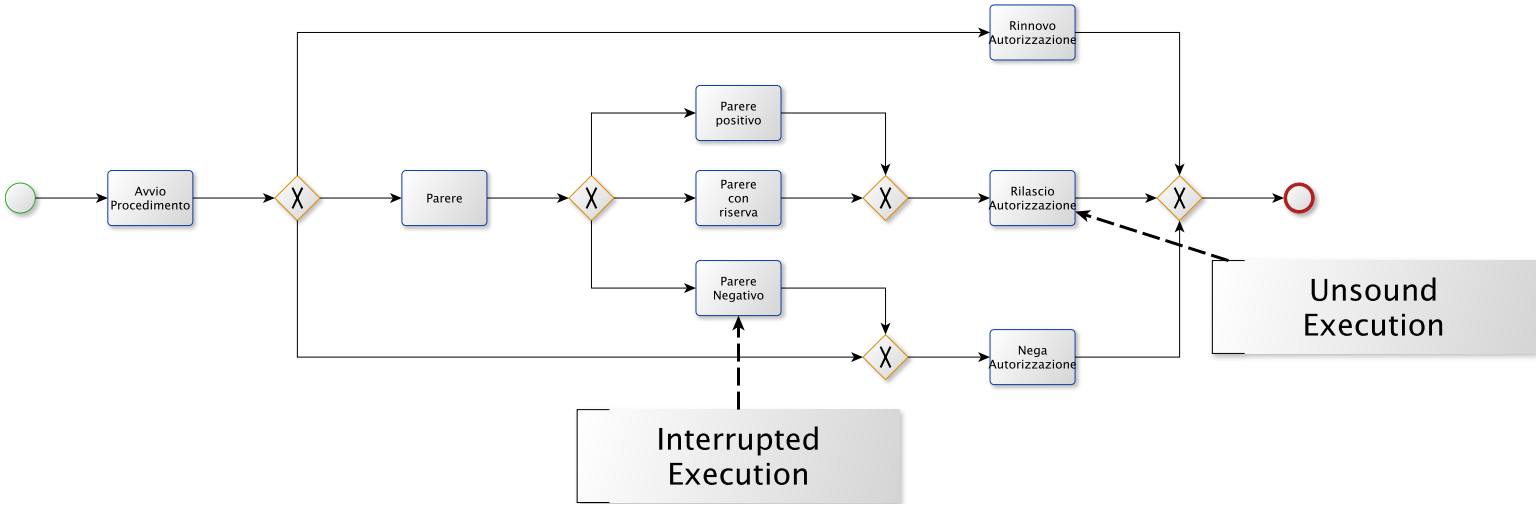
Da messaggi SOAP a eventi/transizioni della Rete di Petri

Messaggi SOAP	**Eventi BPMN**		
richiestaAutorizzazione request	AvvioProcedimento start	AvvioProcedimento complete	
interrogaStatoAutorizzazione response[Rinnovo]	RinnovaAutorizzazione start	RinnovaAutorizzazione complete	
interrogaStatoAutorizzazione response[Rilascio]	RilascioAutorizzazione start	RilascioAutorizzazione complete	
interrogaStatoAutorizzazione response[Nega]	RilascioAutorizzazione start	NegaAutorizzazione complete	
richiestaParere request	Parere start		
emissioneParere request[Negativo]	Parere complete	ParereNegativo start	ParereNegativo complete
emissioneParere request[Positivo]	Parere complete	ParerePositivo start	ParerePositivo complete
emissioneParere request[conRiserva]	Parere complete	ParereConRiserva start	ParereConRiserva complete

Esempio di analisi di conformance

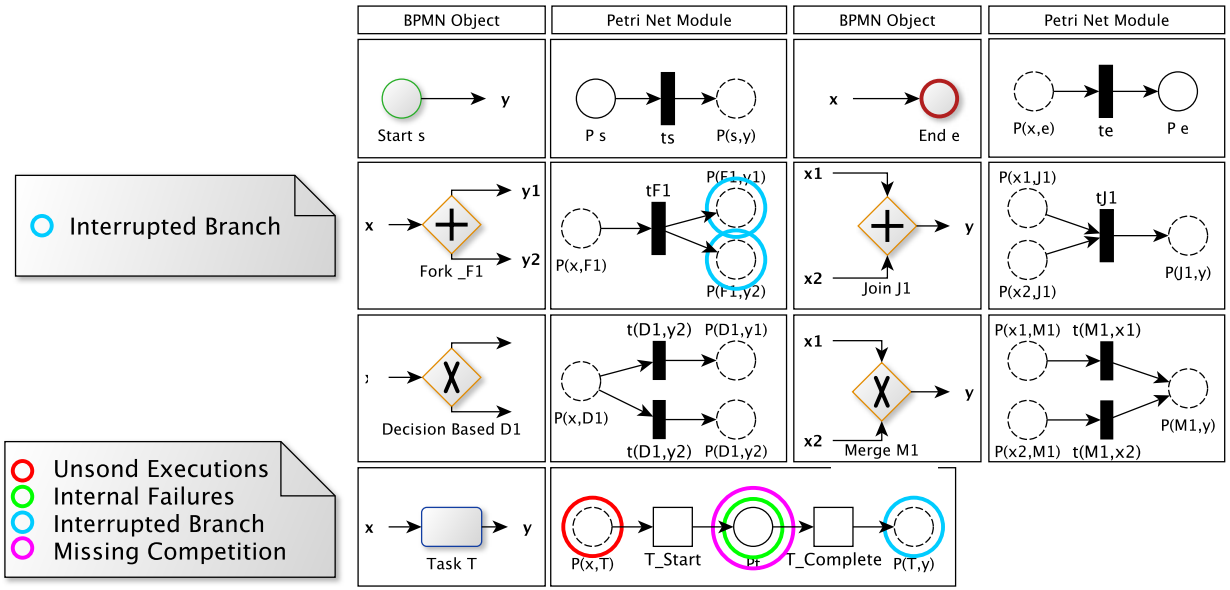




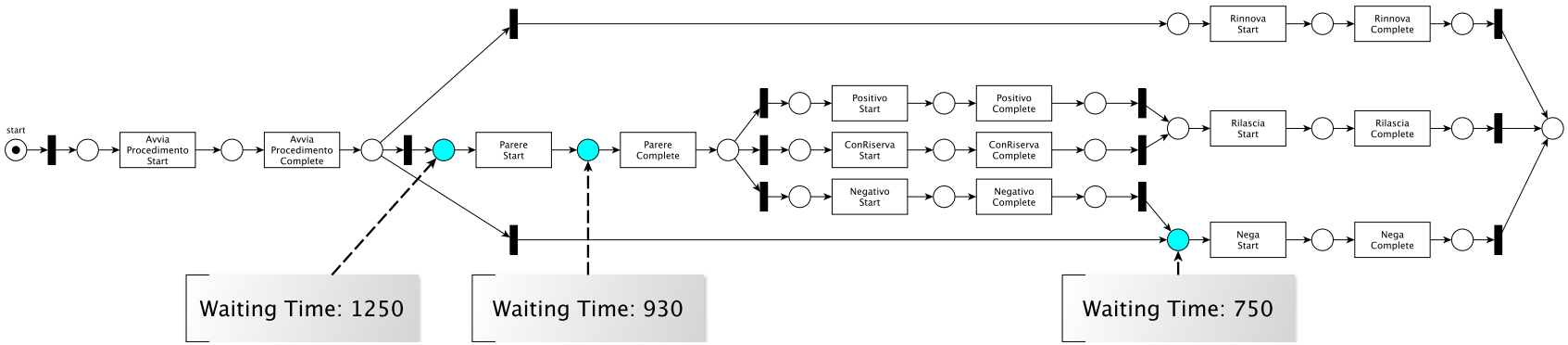


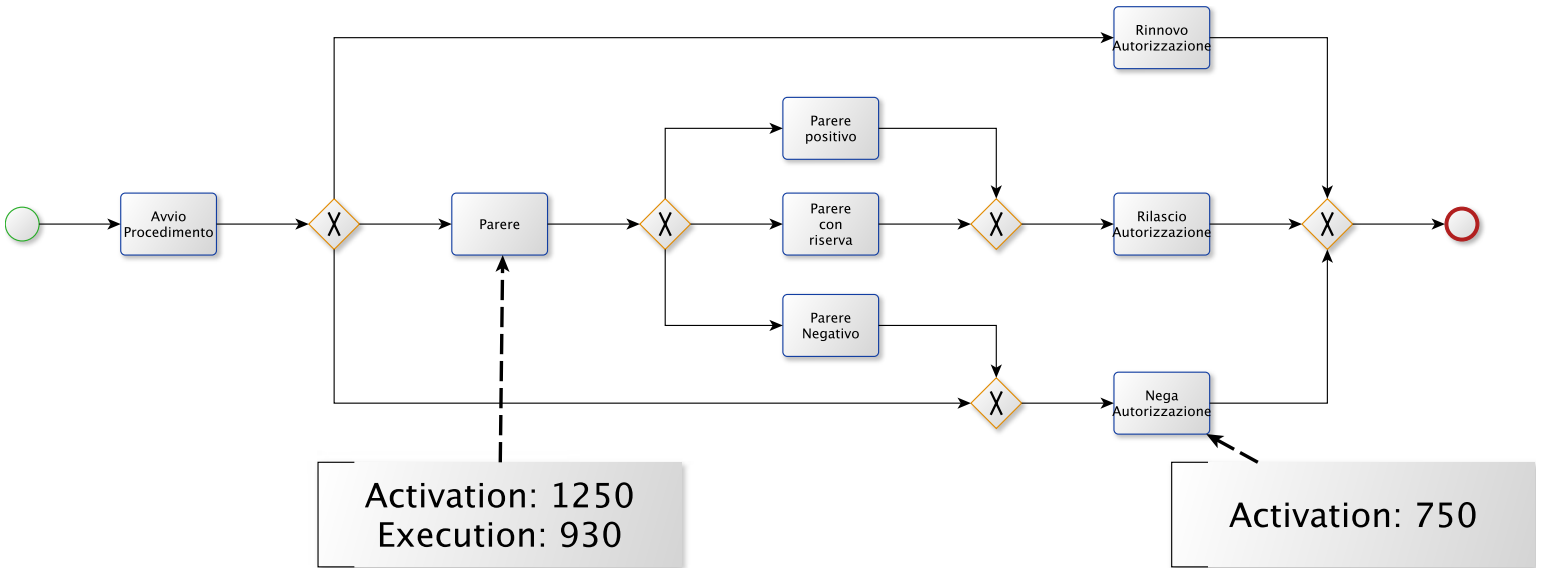
Proiezione dei risultati di analisi su BPMN (Conformance)

- Token mancanti: Il log-replay produce token mancanti solo per eseguire transizioni visibili ⇒ pre-set di almeno una transizione visibile
- Token rimanenti Le transizioni invisibili sono eseguite solo se richiesto da una transizione visibile ⇒ piazze nel post-set di una transizione visibile o di una transizione invisibile che produce più di un token



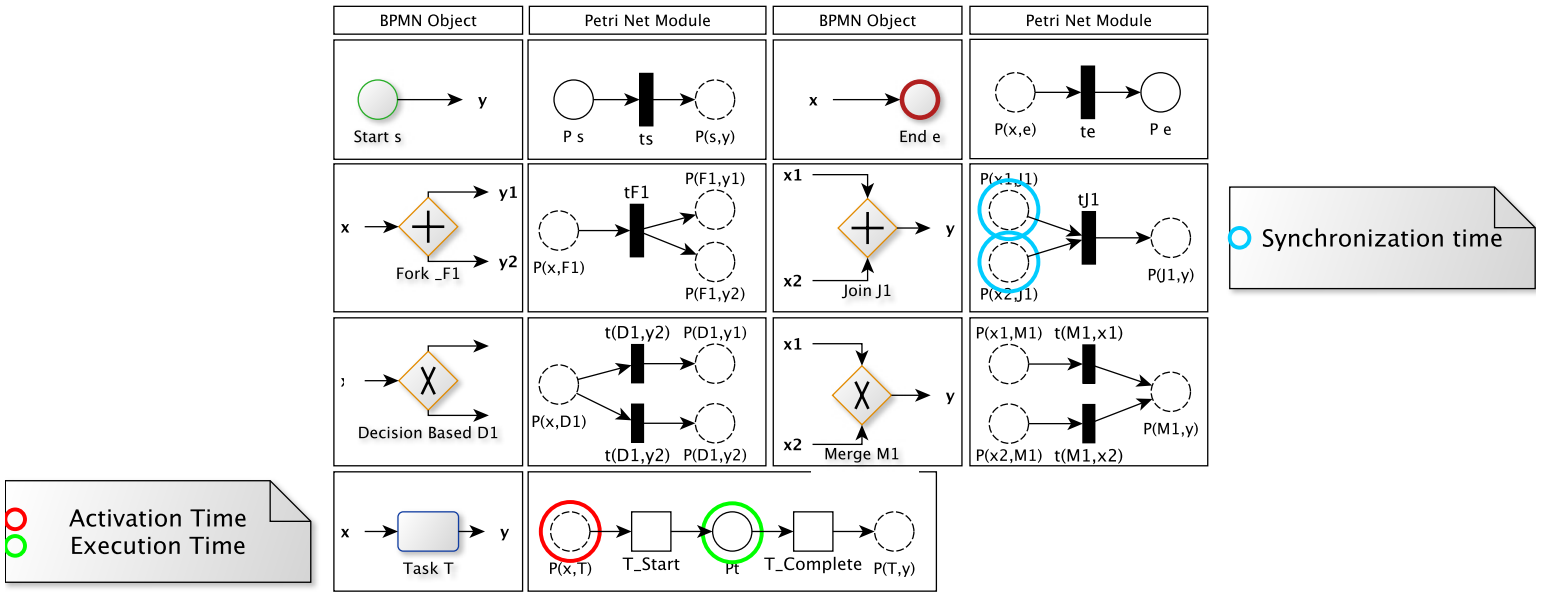
Esempio di analisi di performance





Proiezione dei risultati di analisi su BPMN (Performance)

- Tempo di attesa: transizioni invisibili eseguite immediatamente ⇒ pre-set di transizioni visibili
- Tempo di sincronizzazione piazze che hanno almeno una transizione nel loro post-set che dipende da un'altra piazza



[A2.3] Middleware prototipale: rilasci a Giugno 2011

- Raffinamento dell'algoritmo di log-replay per una migliore gestione delle transizioni invisibili
- Metodologia per proiettare misure di analisi sul modello BPMN
- Nuovo contesto ProM per eseguire plugin in ambiente senza GUI
- Plugin per trasformazione di sequence di eventi in sequenze eager
- Plugin per valutazione di performance di una Rete di Petri

Middleware prototipale: rilasci ad oggi

- Plugin per trasformazione di Modelli BPMN in Reti di Petri
- Plugin per proiezione di misure di analisi sul modello BPMN originale

Middleware prototipale: sviluppi in corso

- Estensione della traduzione BPMN – Rete di Petri con gestione di ciclo di vita di task con eventi intermedi
- Integrazione nella piattaforma di toolkits di Data Mining

[A2.3] Middleware prototipale: rilasci a Giugno 2011

- Raffinamento dell'algoritmo di log-replay per una migliore gestione delle transizioni invisibili
- Metodologia per proiettare misure di analisi sul modello BPMN
- Nuovo contesto ProM per eseguire plugin in ambiente senza GUI
- Plugin per trasformazione di sequence di eventi in sequenze eager
- Plugin per valutazione di performance di una Rete di Petri

Middleware prototipale: rilasci ad oggi

- Plugin per trasformazione di Modelli BPMN in Reti di Petri
- Plugin per proiezione di misure di analisi sul modello BPMN originale

Middleware prototipale: sviluppi in corso

- Estensione della traduzione BPMN – Rete di Petri con gestione di ciclo di vita di task con eventi intermedi
- Integrazione nella piattaforma di toolkits di Data Mining

[A2.3] Middleware prototipale: rilasci a Giugno 2011

- Raffinamento dell'algoritmo di log-replay per una migliore gestione delle transizioni invisibili
- Metodologia per proiettare misure di analisi sul modello BPMN
- Nuovo contesto ProM per eseguire plugin in ambiente senza GUI
- Plugin per trasformazione di sequence di eventi in sequenze eager
- Plugin per valutazione di performance di una Rete di Petri

Middleware prototipale: rilasci ad oggi

- Plugin per trasformazione di Modelli BPMN in Reti di Petri
- Plugin per proiezione di misure di analisi sul modello BPMN originale

Middleware prototipale: sviluppi in corso

- Estensione della traduzione BPMN – Rete di Petri con gestione di ciclo di vita di task con eventi intermedi
- Integrazione nella piattaforma di toolkits di Data Mining

Tecniche di analisi di Reti di Petri adottate

- I log delle istanze di processi sono sequenze ordinate di eventi (e.g. in base a timestamp)
- Gli eventi dei log sono mappati su transizioni della rete
- Algoritmo di log-replay: ri-esegue un log di una istanza di processo in modo "non bloccante"
  1. Si mette un token nella piazza di partenza
  2. Se estrae il primo evento del log
  3. Si esegue la transizione corrispondente
     - se la transizione non è abilitata vengono creati artificialmente i token mancanti
- Metriche calcolate durante il log-replay
  - Numero di token mancanti o rimanenti per ogni piazza o transizione
  - Numero di attraversamenti per ogni arco
  - Tempo di soggiorno/attesa/sincronizzazione per ogni piazza

Tecniche di analisi di Reti di Petri adottate

- I log delle istanze di processi sono sequenze ordinate di eventi (e.g. in base a timestamp)
- Gli eventi dei log sono mappati su transizioni della rete
- Algoritmo di log-replay: ri-esegue un log di una istanza di processo in modo "non bloccante"
  1. Si mette un token nella piazza di partenza
  2. Se estrae il primo evento del log
  3. Si esegue la transizione corrispondente
     - se la transizione non è abilitata vengono creati artificialmente i token mancanti
- Metriche calcolate durante il log-replay
  - Numero di token mancanti o rimanenti per ogni piazza o transizione
  - Numero di attraversamenti per ogni arco
  - Tempo di soggiorno/attesa/sincronizzazione per ogni piazza

Tecniche di analisi di Reti di Petri adottate

- I log delle istanze di processi sono sequenze ordinate di eventi (e.g. in base a timestamp)
- Gli eventi dei log sono mappati su transizioni della rete
- Algoritmo di log-replay: ri-esegue un log di una istanza di processo in modo "non bloccante"
  1. Si mette un token nella piazza di partenza
  2. Se estrae il primo evento del log
  3. Si esegue la transizione corrispondente
     - se la transizione non è abilitata vengono creati artificialmente i token mancanti
- Metriche calcolate durante il log-replay
  - Numero di token mancanti o rimanenti per ogni piazza o transizione
  - Numero di attraversamenti per ogni arco
  - Tempo di soggiorno/attesa/sincronizzazione per ogni piazza

Un contributo originale: raffinamento dell'analisi di performance

- Sfrutta le tecniche standard di log-replay per riusare l'infrastruttura software esistente
- Trasforma la lista di transizioni risultante $R = [tr_1, ..., tr_n]$ in una sequenza "eager", cioè tale che per ogni transizione invisibile tr_i valga:
  - sia tr_p l'ultima transizione visibile che la precede ($p < i$)
  - allora $\bullet tr_i \cap tr_p \bullet \neq \emptyset$
- Un semplice algoritmo di trasformazione: per ogni transizione invisibile tr_i
  1. sposta verso sinistra la transizione finché non si trova una transizione visibile tale che $\bullet tr_i \cap tr_p \bullet \neq \emptyset$
- Non sono necessari cambi relativi alle metriche di conformance