

Report progetto

Anno Accademico 2018 - 2019

Studenti:

Alberto Bezzon 1211016

Tommaso Carraro 1210937

TERZA PARTE

Lo scopo della terza parte della tesina è l'inserimento del processo modellato tramite Petri-net nella parte 1 in un sistema informativo, in particolare è stato utilizzato il software YAWL. Lo scopo di questo report è la descrizione del modello realizzato e delle scelte progettuali. La descrizione del modello seguirà la struttura utilizzata nel primo report, ovvero verrà descritto ogni requisito singolarmente. La descrizione di un requisito comprende un'immagine della parte di modello che implementa il requisito, una descrizione testuale del requisito, una descrizione delle variabili di task e di rete che il requisito ha richiesto e, se il requisito ha richiesto la modellazione di task manuali, viene fornita una descrizione del processo dalla parte client di YAWL per ogni task manuale.

REQUISITO 1

DESCRIZIONE

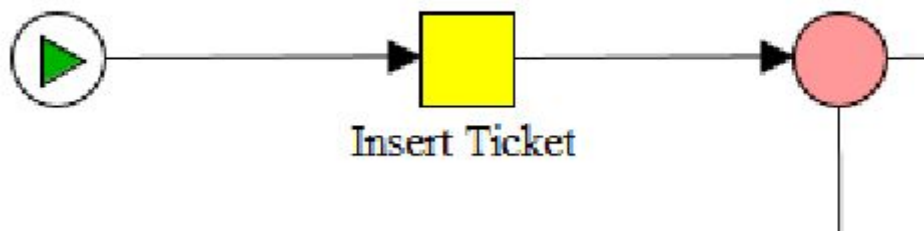


Fig. 1: Parte di rete che modella il primo requisito

Quando viene commessa una violazione deve essere eseguito il task *Insert Ticket*. Tale task è manuale e richiede l'inserimento da parte di un operatore di ruolo *Insertor* delle seguenti informazioni:

- Numero dell'articolo violato;
- Nome del violatore;
- Indirizzo del violatore;
- Somma che il violatore deve pagare per la violazione commessa.

DESCRIZIONE DECOMPOSIZIONE

La decomposizione del task *Insert Ticket* comprende le seguenti variabili:

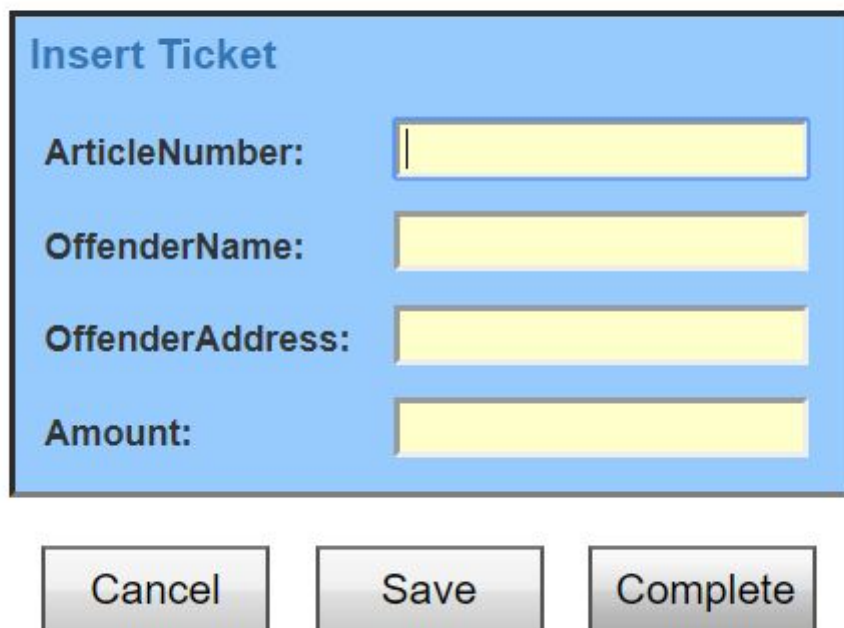
- *ArticleNumber* (integer): questa variabile rappresenta il numero dell'articolo violato. È una variabile di output in quanto il task richiede che sia l'operatore ad inserire il suo valore;
- *OffenderName* (string): questa variabile rappresenta il nome del violatore. È una variabile di output in quanto il task richiede che sia l'operatore ad inserire il suo valore;
- *OffenderAddress* (string): questa variabile rappresenta l'indirizzo di residenza del violatore. È una variabile di output in quanto il task richiede che sia l'operatore ad inserire il suo valore;
- *Amount* (double): questa variabile rappresenta la somma che il violatore deve pagare per la violazione commessa. È una variabile di output in quanto il task richiede che sia l'operatore ad inserire il suo valore.

Queste variabili presentano un binding con delle variabili di rete che condividono la stessa denominazione.

DESCRIZIONE DEL PROCESSO DALLA PARTE CLIENT

Ogni caso lanciato per questo modello corrisponde alla ricezione di una violazione da parte del sistema. Quando una violazione viene commessa il sistema offre agli operatori di ruolo *Insertor* l'esecuzione del task *Insert Ticket*. Se un operatore di ruolo *Insertor* risulta disponibile e decide di eseguire il task, allora verrà visualizzata la seguente interfaccia grafica. L'interfaccia richiede l'inserimento delle informazioni precedentemente descritte. Nel caso in cui l'operatore non inserisca delle informazioni sono stati impostati dei valori di default per le variabili di rete.

Edit Work Item: 27.1



Insert Ticket

ArticleNumber:

OffenderName:

OffenderAddress:

Amount:

Fig. 2: Interfaccia grafica del client YAWL (*Insert Ticket* task)

REQUISITO 2

DESCRIZIONE

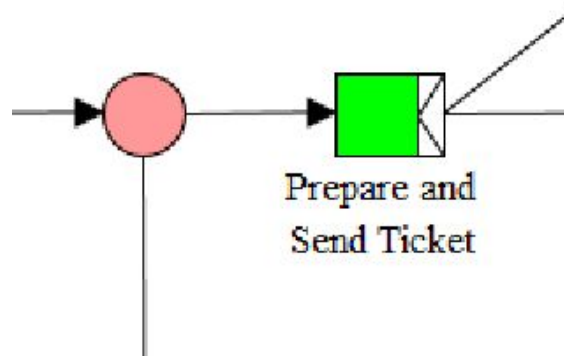


Fig. 3: Parte di rete che modella il secondo requisito

Una volta che il ticket è stato inserito nel sistema esso deve essere preparato e spedito per posta al violatore. Questo processo è modellato dal task manuale *Prepare and Send Ticket* che può essere eseguito solamente da un operatore di ruolo *Sender*. Il task prevede la visualizzazione delle seguenti informazioni in fase di invio del ticket:

- Numero dell'articolo violato;
- Nome del violatore;
- Indirizzo del violatore;
- Somma che il violatore deve pagare per la violazione commessa.

DESCRIZIONE DECOMPOSIZIONE

La decomposizione del task *Prepare and Send Ticket* comprende le seguenti variabili:

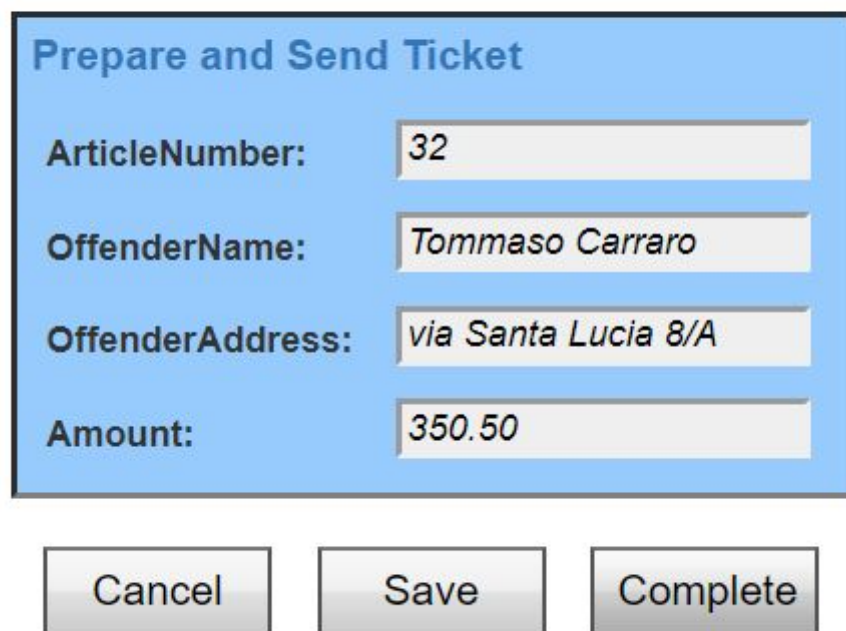
- *ArticleNumber* (integer): questa variabile rappresenta il numero dell'articolo violato. È una variabile di input in quanto il task richiede che l'operatore possa leggere il suo valore prima di inviare il ticket;
- *OffenderName* (string): questa variabile rappresenta il nome del violatore. È una variabile di input in quanto il task richiede che l'operatore possa leggere il suo valore prima di inviare il ticket;
- *OffenderAddress* (string): questa variabile rappresenta l'indirizzo di residenza del violatore. È una variabile di input in quanto il task richiede che l'operatore possa leggere il suo valore prima di inviare il ticket;
- *Amount* (double): questa variabile rappresenta la somma che il violatore deve pagare per la violazione commessa. È una variabile di input in quanto il task richiede che l'operatore possa leggere il suo valore prima di inviare il ticket.

Queste variabili presentano un binding con delle variabili di rete che condividono la stessa denominazione.

DESCRIZIONE DEL PROCESSO DALLA PARTE CLIENT

Quando un ticket viene inserito nel sistema da parte di un operatore di ruolo *Insertor*, l'attività *Prepare and Send Ticket* viene offerta ad un operatore di ruolo *Sender*. Se un operatore di ruolo *Sender* risulta disponibile, allora può eseguire l'attività. L'esecuzione dell'attività consiste nella visualizzazione della seguente interfaccia grafica, la quale permette la visualizzazione delle informazioni descritte precedentemente e la conferma di invio del ticket al violatore.

Edit Work Item: 27.2



Prepare and Send Ticket	
ArticleNumber:	32
OffenderName:	Tommaso Carraro
OffenderAddress:	via Santa Lucia 8/A
Amount:	350.50

Cancel Save Complete

Fig. 4: Interfaccia grafica del client YAWL (*Prepare and Send Ticket* task)

DESCRIZIONE DEL PROCESSO DALLA PARTE CLIENT

Quando un ticket viene preparato e spedito al violatore da parte di un operatore di ruolo *Sender*, l'attività *Process Payment* viene offerta ad un operatore di ruolo *Processor*. Se un operatore di ruolo *Processor* risulta disponibile, allora può eseguire l'attività. L'esecuzione dell'attività consiste nella visualizzazione della seguente interfaccia grafica, la quale permette la visualizzazione delle informazioni descritte precedentemente e l'inserimento della somma parziale pagata dal violatore. Questa attività continuerà ad essere offerta ad operatori di ruolo *Processor* fino a quando il violatore non avrà pagato interamente la somma dovuta oppure finché non saranno scaduti i 180 giorni (requisito 5) che vengono dati a disposizione dal sistema per il pagamento totale della somma.

Edit Work Item: 27.3

Process Payment

OffenderName:

OffenderAddress:

Payment:

Cancel

Save

Complete

Fig. 6: Interfaccia grafica del client YAWL (*Process Payment* task)

DESCRIZIONE DECOMPOSIZIONE AUTOMATIC PAYMENT PROCESSING

La decomposizione del task automatico *Automatic Payment Processing* presenta una codelet XQueryEvaluator. La codelet si occupa di sommare alla variabile di rete AmountPaid, la quale contiene la somma fino ad ora pagata dal violatore (inizialmente nulla), la variabile Payment, il quale valore è stato inserito da un operatore durante l'esecuzione del task *Process Payment*. Le variabili di task coinvolte nella codelet sono:

- query: questa variabile contiene il risultato della somma descritta, che viene poi copiato nella variabile result;
- result: questa variabile contiene il risultato della codelet, che viene poi inserito nella variabile di rete AmountPaid, in modo da aggiornare il suo valore.

Il task *Automatic Payment Processing* presenta inoltre uno XOR-split. Infatti, se con l'ultima esecuzione del task *Process Payment* è stata raggiunta la somma totale da pagare ($\text{AmountPaid} \geq \text{Amount}$), allora l'esecuzione del modello continuerà con l'esecuzione del task automatico *Close and Archive Ticket*. Se invece la somma pagata durante l'ultima esecuzione del task *Process Payment* non è stata sufficiente a coprire l'intera somma relativa alla violazione commessa ($\text{AmountPaid} < \text{Amount}$), allora verrà offerta nuovamente l'attività *Process Payment* ad un operatore di ruolo *Processor*, in modo da permettere al violatore di continuare a pagare.

REQUISITO 5

DESCRIZIONE

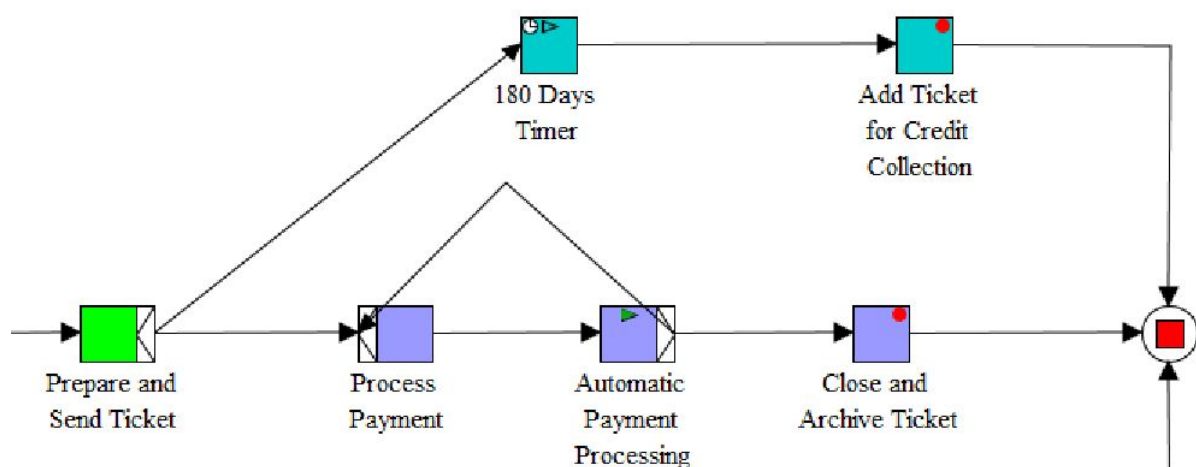


Fig. 7: Parte di rete che modella il quinto requisito

In seguito alla preparazione e all'invio di un ticket vengono dati a disposizione 180 giorni al violatore per pagare l'intera somma associata alla violazione commessa. Se al termine dei 180 giorni il violatore non ha ancora pagato interamente la somma dovuta ($\text{AmountPaid} < \text{Amount}$), allora il ticket deve essere preparato per la Credit Collection. Per modellare questo processo si è inserito un AND-split nel task *Prepare and Send Ticket*. Tale split permette di offrire contemporaneamente due task:

1. il processamento del pagamento relativo al ticket (*Process Payment* task);
2. l'avvio di un timer di 180 giorni (*180 Days Timer* task).

In questo modo, se alla scadenza del timer il violatore non avrà ancora pagato interamente la somma dovuta, verrà eseguito il task automatico *Add Ticket for Credit Collection* che terminerà la gestione del ticket.

DESCRIZIONE CANCELLATION REGION DELLA RETE

Nella rete modellata sono presenti due cancellation region:

1. Task *Close and Archive Ticket*: se si arriva all'esecuzione di questo task significa che il violatore ha pagato interamente la somma dovuta ($\text{AmountPaid} \geq \text{Amount}$) e quindi il timer relativo al pagamento del ticket deve essere fermato. La cancellation region associata a questo task permette di eliminare il token presente nel task automatico *180 Days Timer*;
2. Task *Add Ticket for Credit Collection*: se si arriva all'esecuzione di questo task significa che dopo 180 giorni il violatore non ha ancora terminato di pagare la somma dovuta. In questo

caso bisogna terminare la gestione del ticket aggiungendolo alla Credit Collection e bloccare la possibilità di effettuare ulteriori pagamenti. La cancellation region associata a questo task permette di eliminare il token presente nel task manuale *Process Payment* o nel task automatico *Automatic Payment Processing*.

REQUISITO 6

DESCRIZIONE

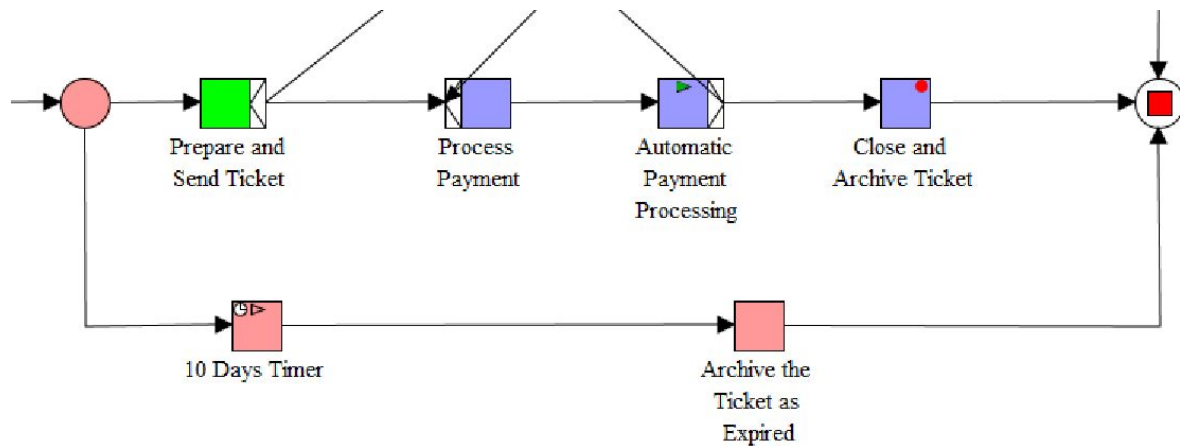


Fig. 8: Parte di rete che modella il sesto requisito

Se dopo l'inserimento di un ticket nel sistema esso non viene preparato ed inviato al violatore entro 10 giorni, il ticket deve essere archiviato come scaduto e il violatore non è tenuto a pagare alcuna somma. Questo processo è stato modellato tramite l'inserimento di una condition subito dopo il task *Insert Ticket*. Tale condition permette di offrire in maniera esclusiva i seguenti due task:

1. Preparazione e invio per posta del ticket (*Prepare and Send Ticket* task);
2. Avvio di un timer di 10 giorni (*10 Days Timer* task).

Il fatto che questi due task vengano offerti in maniera esclusiva significa che se un operatore di ruolo *Sender* esegue il task *Prepare and Send Ticket*, allora il token presente sulla condition viene consumato dall'esecuzione di tale task e quindi il timer viene bloccato. Se invece alla scadenza dei 10 giorni dati a disposizione dal sistema per la preparazione e l'invio del ticket nessun operatore ha ancora eseguito il task *Prepare and Send Ticket*, allora il token presente sulla condition viene consumato dal task *10 Days Timer* e il task *Prepare and Send Ticket* non verrà più offerto agli operatori di ruolo *Sender*. Se il timer scade, deve essere eseguito il task automatico *Archive the Ticket as Expired* che termina la gestione del ticket.

RETE COMPLETA

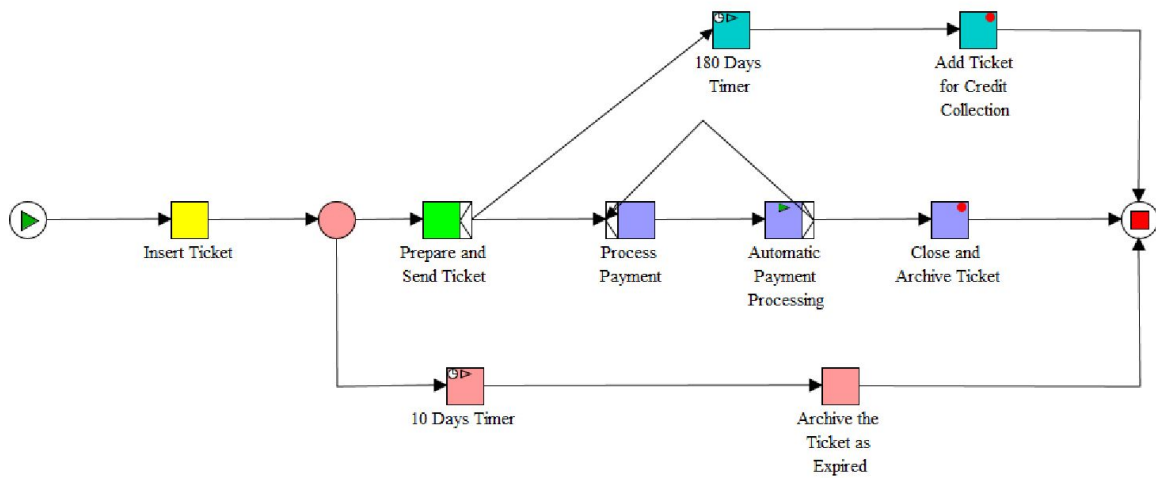


Fig.9 : Rete completa