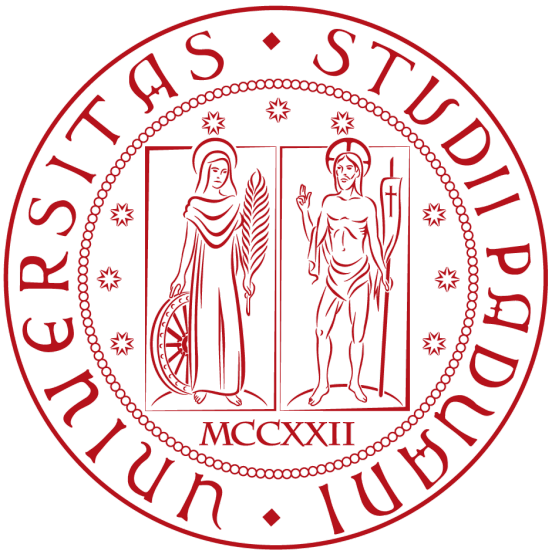
Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA
Corso di Advanced Topics in Computer Science



Report progetto

Anno Accademico 2018 - 2019

Studenti:

Alberto Bezzon 1211016
Tommaso Carraro 1210937

PRIMA PARTE

REQUISITO 1

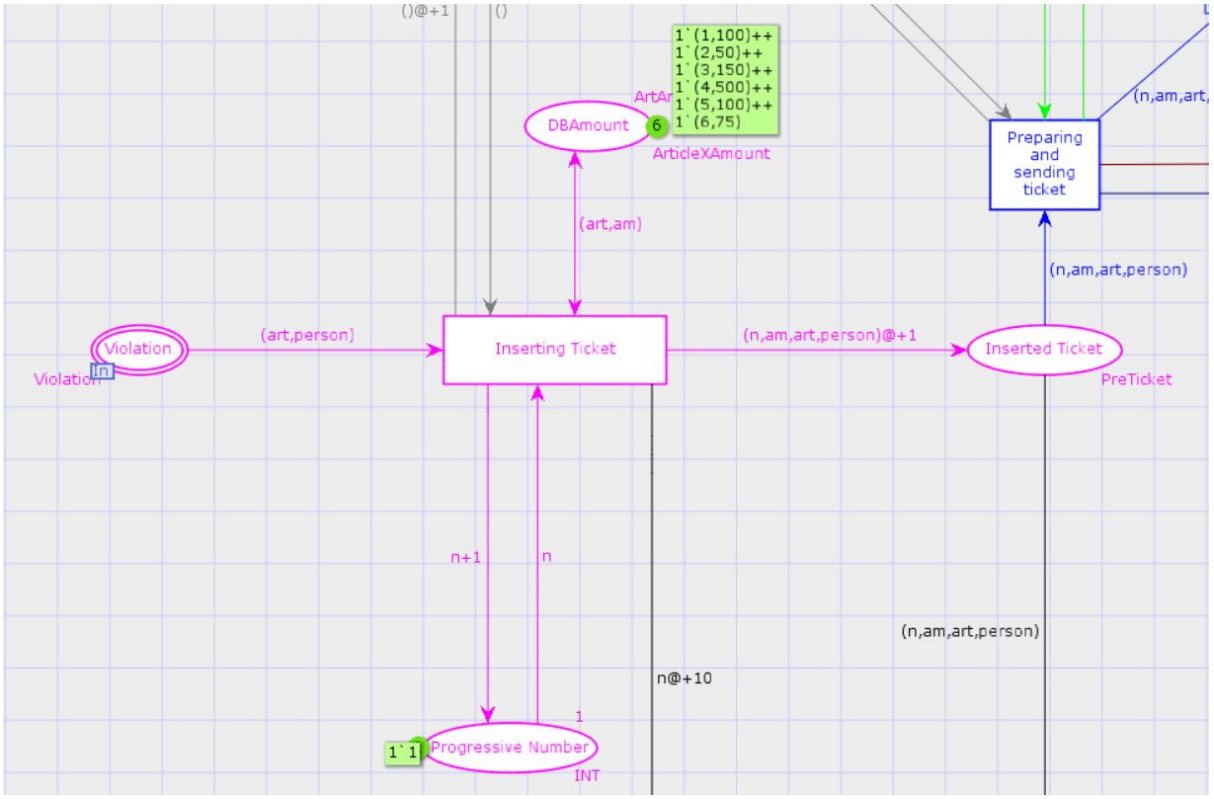


Fig. 1: Parte di rete che modella il primo requisito

DESCRIZIONE

La produzione di un token sul place *Violation* corrisponde alla segnalazione di una violazione. Quando il sistema registra una violazione deve essere creato un ticket corrispondente alla violazione commessa.
La transizione *Inserting Ticket* modella l'attività di inserimento di un ticket nel sistema.
Il place *Progressive Number* permette di associare ad ogni ticket creato un codice progressivo e univoco.
Il place *DBAmount* permette di associare l'articolo violato con l'importo che deve essere pagato.
La transizione produce un token di *color set PreTicket* sul place *Inserted Ticket*. Una volta creato, un ticket è di tipo pre-ticket, nel senso che non è ancora stato inviato e quindi non è associato ad una lista di pagamenti. L'attività di inserimento di un ticket richiede un giorno, questo è stato modellato tramite un incremento di 1 sul tempo associato al token prodotto.

REQUISITO 2

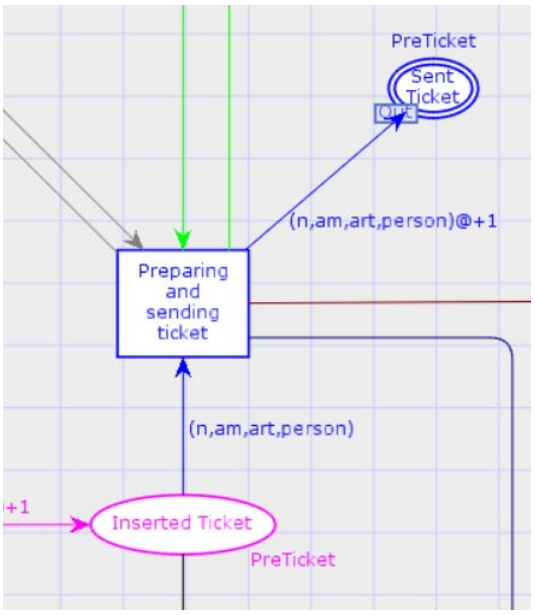


Fig. 2: Parte di rete che modella il secondo requisito

DESCRIZIONE

I ticket creati ed inseriti nel sistema sono rappresentati da token all'interno del place *Inserted Ticket*. Una volta che un ticket è stato inserito nel sistema diventa pronto per essere inviato. La transizione *Preparing and sending ticket* modella l'invio di un ticket alla persona interessata. La transizione consuma un token da *Inserted Ticket* e produce un token su *Sent Ticket*. La preparazione e l'invio del ticket richiede un giorno, per cui il token prodotto su *Sent Ticket* riceve un *delay* di un'unità di tempo.

REQUISITO 3

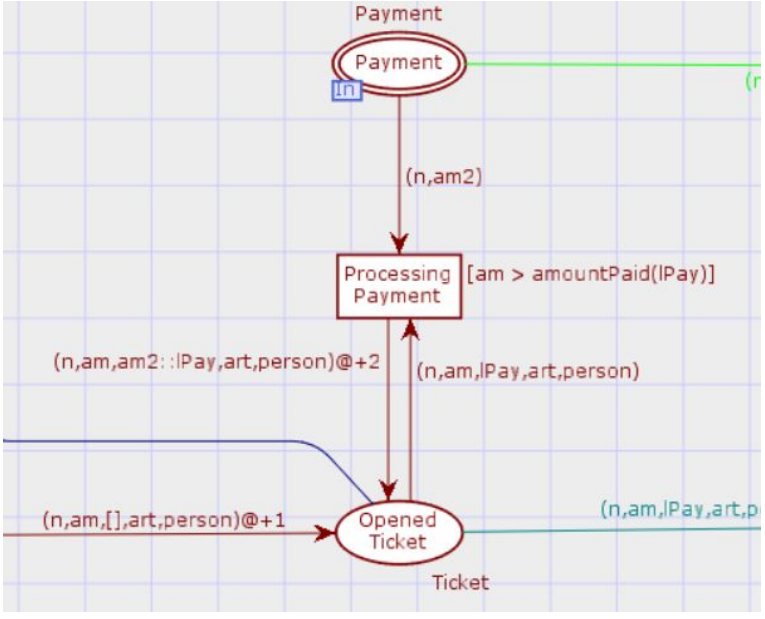


Fig. 3: Parte di rete che modella il terzo requisito

DESCRIZIONE

Nella nostra soluzione, una volta che un ticket viene inviato, esso viene anche aperto. Un ticket è aperto se sta attendendo il dovuto pagamento. Un ticket aperto si differenzia da un pre-ticket in quanto gli viene associata una lista di pagamenti, inizialmente vuota. Un ticket rimane aperto fino a quando il pagamento non è stato del tutto versato o fino a quando non sono scaduti i 180 giorni (requisito 5) che vengono dati a disposizione per il pagamento. Il processo di apertura di un ticket è modellato dalla produzione di un token sul place *Opened Ticket* da parte della transizione *Preparing and sending ticket*. Poiché preparare e inviare un ticket richiede un giorno, anche i token prodotti su *Opened Ticket* ricevono un *delay* di un'unità di tempo. Il processamento di un pagamento è modellato dalla transizione *Processing Payment* che consuma un token da *Opened Ticket* e da *Payment* e riproduce il token su *Opened Ticket* accodando alla lista dei pagamenti del ticket la somma prelevata da *Payment*. La transizione può essere eseguita solamente se il termine di pagamento non è scaduto, se è arrivato almeno un pagamento per il ticket da processare e se non si ha già terminato di pagare il ticket. Poiché il processamento di un pagamento richiede due giorni, la transizione produce il token su *Opened Ticket* con un *delay* di due giorni di tempo.

REQUISITO 4

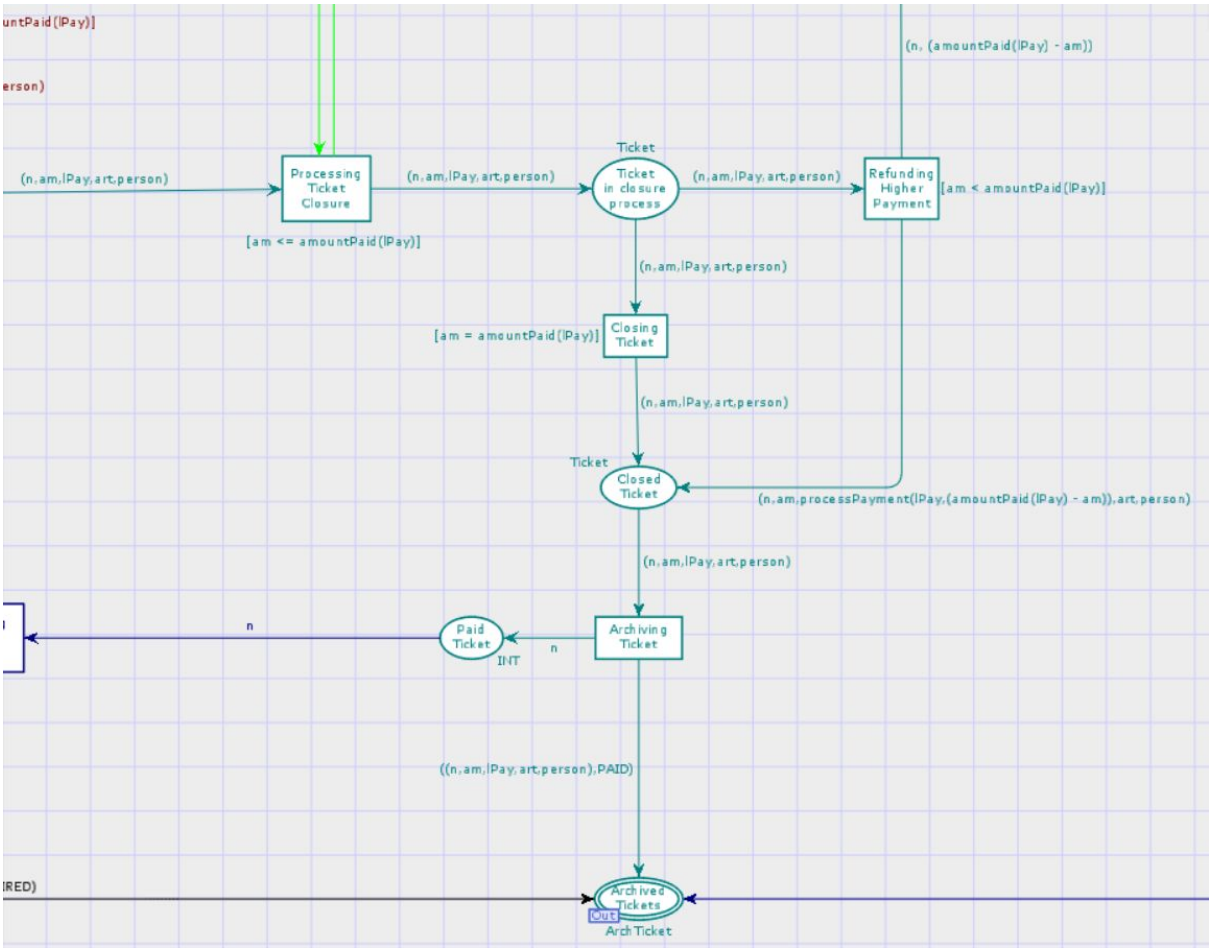
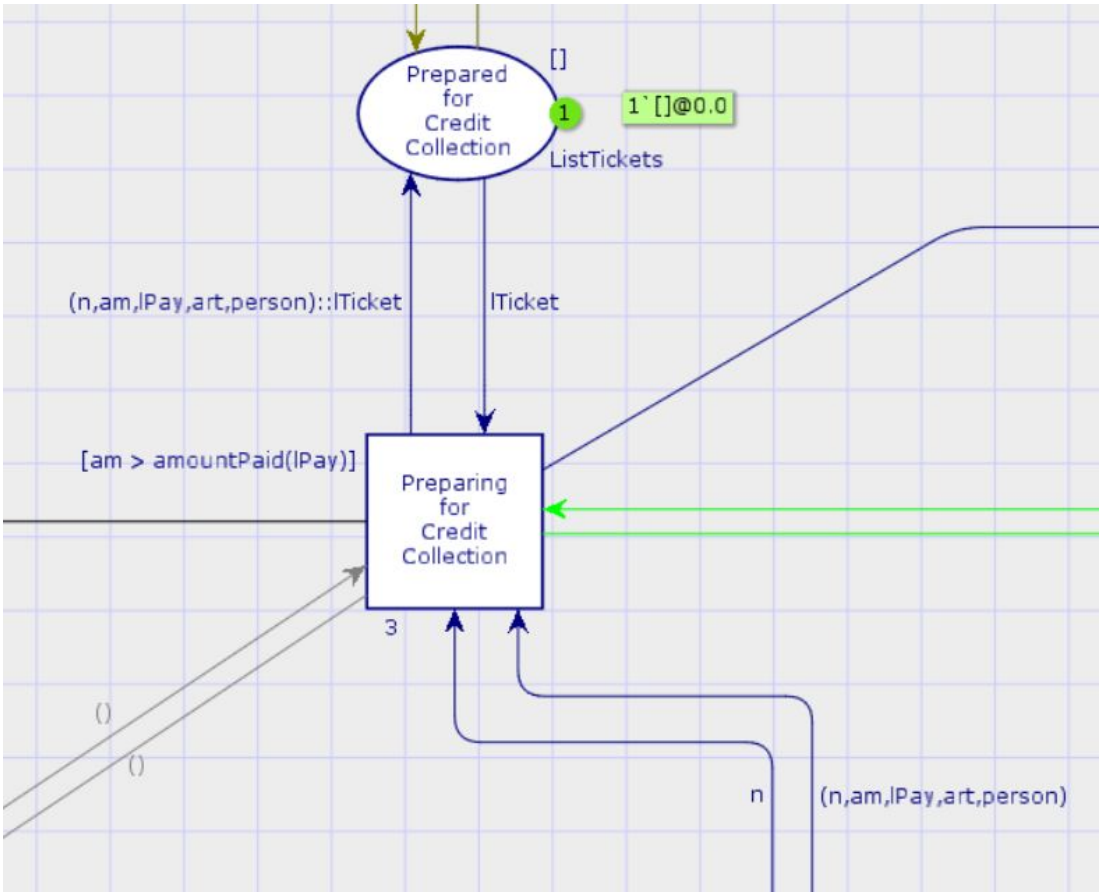


Fig. 4: Parte di rete che modella il quarto requisito

DESCRIZIONE

Se un ticket in *Opened Ticket* è stato pagato del tutto, ovvero la somma dei pagamenti, calcolata con la funzione *amountPaid*, è maggiore o uguale alla somma da pagare, può essere eseguita solamente la transizione *Processing Ticket Closure*. Essa consuma un token da *Opened Ticket* e produce un token su *Ticket in closure process*. Arrivati a questo punto può essere che l'importo sia stato pagato perfettamente oppure che il pagante abbia superato la somma da pagare. In questo caso la rete è stata modellata per effettuare un rimborso della somma aggiuntiva (non era richiesto dai requisiti ma sembrava interessante implementarlo). Quindi se un ticket è stato pagato perfettamente viene eseguita la transizione *Closing Ticket* che consuma il token da *Ticket in closure process* e lo produce su *Closed Ticket*, mentre se è stato pagato più del dovuto viene eseguita la transizione *Refunding Higher Payment*. Essa consuma un token da *Ticket in closure process* e produce un token su *Rifiuted Payment* e un token su *Closed Ticket*. Su *Rifiuted Payment* viene prodotta la somma rimborsata, mentre su *Closed Ticket* viene prodotto un ticket dove l'ultimo elemento della lista dei pagamenti viene ridotto della somma rimborsata (l'ultimo pagamento è quello che deve essere in parte rimborsato). Questo processo viene eseguito dalla funzione *processPayment* che si occupa appunto di scorrere la lista dei pagamenti fino all'ultimo elemento e di modificare lo stesso come richiesto. Una volta che un ticket viene chiuso, esso può essere archiviato. Questo processo è eseguito dalla transizione *Archiving Ticket* che si occupa di archiviare il ticket con ragione PAID, che indica che il ticket è stato pagato interamente.

REQUISITO 5



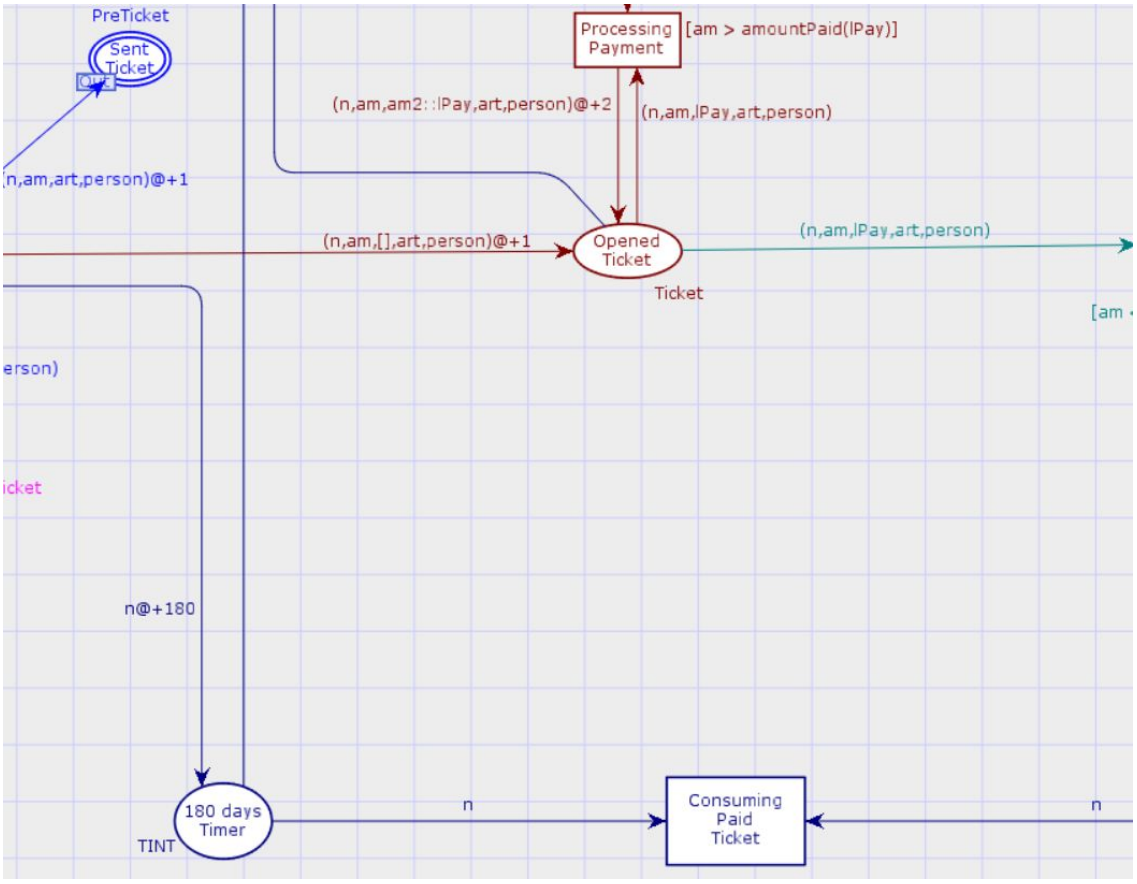


Fig. 5: Parte di rete che modella il quinto requisito

DESCRIZIONE

Come già detto precedentemente, una volta che un ticket viene inviato e aperto tramite la transizione *Preparing and sending ticket*, deve essere fatto partire un timer di 180 giorni. Esso rappresenta il tempo che viene dato a disposizione al pagante per saldare il debito. Il timer è stato modellato tramite la produzione di un token (codice ticket) con un delay di 180 giorni sul place *180 days Timer* da parte della transizione *Preparing and sending ticket*. Una volta scaduto il timer, se il ticket corrispondente si trova ancora in *Opened Ticket* e il rispettivo pagamento non è ancora stato portato a compimento, il ticket deve essere preparato per la *Credit Collection*. La transizione *Preparing for Credit Collection* si occupa appunto di eseguire questo processo. Essa consuma un ticket da *Opened Ticket* e lo aggiunge alla lista dei ticket da inviare in *Credit Collection* (ovviamente la transizione consuma anche il token relativo alla scadenza del timer dal place *180 days Timer*). Questa lista è mantenuta nel place *Prepared for Credit Collection*. Per il place è stato impostato un marking iniziale corrispondente alla lista vuota. La transizione recupera la lista dal place e la riproduce sullo stesso aggiungendo in coda il nuovo ticket da aggiungere alla *Credit Collection*. Per la transizione è stata impostata un'alta priorità (3), questo fa in modo che appena i timer iniziano a scadere sul place *180 days Timer* e la transizione diventa *enable*, essa sarà sempre la prima ad essere eseguita rispetto a tutte le altre transizioni *enable*. In questo modo non si presenta il rischio che un pagamento che arriva durante la scadenza del ticket venga accettato. Poiché nel place *Opened Ticket* possono essere presenti anche ticket pagati del tutto, ma che non sono ancora stati processati per la chisura, è stata inserita una guardia sulla transizione *Preparing for Credit Collection*. Questa guardia fa in modo che vengano consumati solamente i ticket scaduti e parzialmente pagati. La transizione si occupa infine di archiviare il ticket con ragione CCOLL. Questo processo è stato modellato tramite la produzione di un token sul place *Archivied Tickets*.

REQUISITO 6

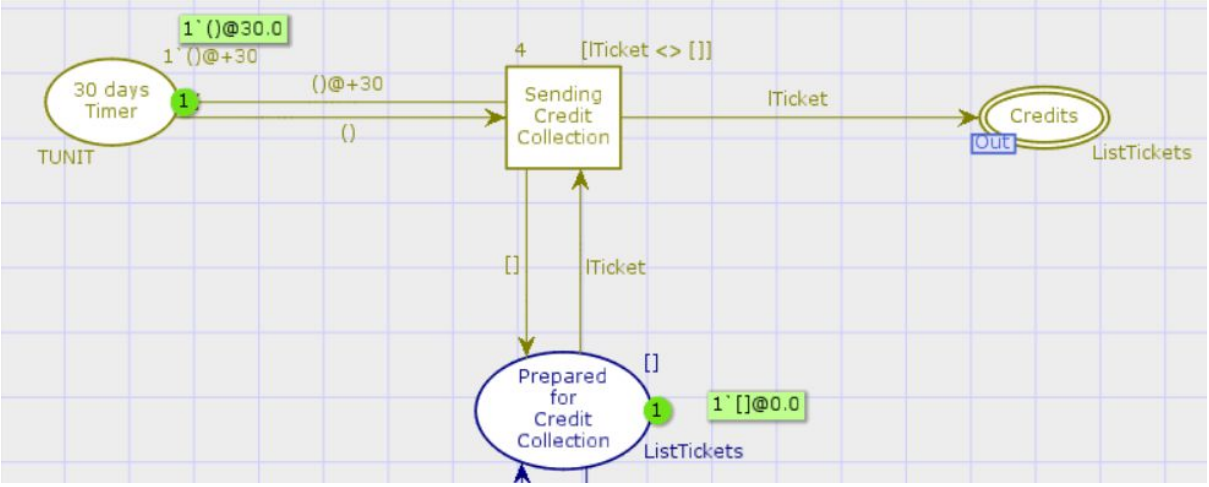


Fig. 6: Parte di rete che modella il sesto requisito

DESCRIZIONE

Per modellare il fatto che ogni 30 giorni deve essere spedita la lista di ticket in *Credit Collection* è stata creata la transizione *Sending Credit Collection*. La transizione può essere eseguita solamente se ci sono ticket in *Credit Collection* (la lista nel place *Prepared for Credit Collection* non è vuota) e deve essere eseguita ogni 30 giorni. Se dopo 30 giorni la lista sul place *Prepared for Credit Collection* è ancora vuota, il timer continua ad avanzare oltre i 30 giorni. Per fare in modo che la transizione sia la prima ad essere eseguita nel caso in cui la lista non fosse vuota, gli è stata assegnata una priorità più alta (4) rispetto alla transizione *Preparing for Credit Collection*. In questo modo la transizione sarà la prima ad essere eseguita nel caso in cui anche *Preparing for Credit Collection* fosse *enable* a sua volta. Questo permette di inviare la lista alla scadenza dei 30 giorni invece di accettare un nuovo ticket per la *Credit Collection*; il nuovo ticket verrà accettato in seguito all'invio della lista, come richiesto. La transizione consuma la lista sul place *Prepared for Credit Collection* e la produce sul place *Credits*, dopodiché setta il place *Prepared for Credit Collection* con la lista vuota e si occupa di far ripartire il timer di 30 giorni, producendo un token con un *delay* di 30 giorni sul place *30 Days Timer*. Poiché il tutto funzioni, per il place *30 Days Timer* è stato settato un marking iniziale corrispondende ad un token con un *delay* di 30 giorni.

REQUISITO 7

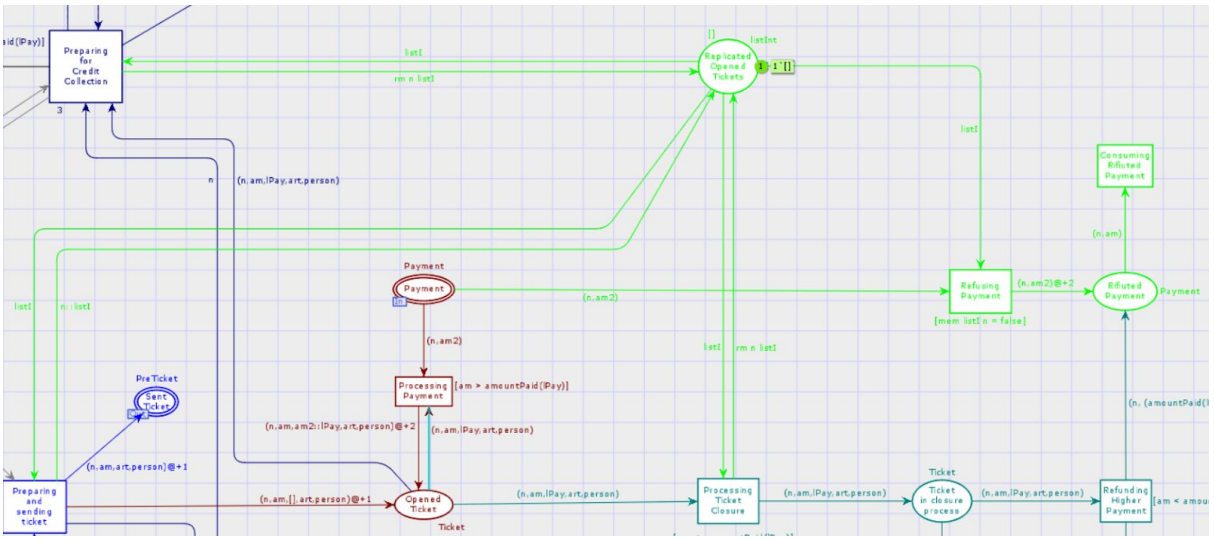


Fig. 7: Parte di rete che modella il settimo requisito

DESCRIZIONE

Per modellare il rifiuto dei pagamenti che arrivano in seguito all'archiviazione dei ticket, si sono dovuti replicare i ticket aperti nel place *Replicated Opened Tickets*. Questo place contiene una lista dei codici dei ticket aperti. Per questo motivo si è dovuto introdurre il *color set listInt* e la variabile *listI*. Ogni volta che viene aperto un ticket, viene inserito in questa lista, mentre ogni volta che un ticket viene pagato completamente, viene rimosso dalla lista. L'idea è che se arriva un pagamento per un ticket che non è all'interno della lista sul place *Replicated Opened Ticket*, questo deve essere rifiutato. La transizione che si occupa del rifiuto dei pagamenti è *Refusing Payment*, che può essere eseguita solamente se il numero di ticket associato al pagamento ricevuto non è contenuto nella lista dei numeri di ticket in *Replicated Opened Tickets*. Questo è stato modellato tramite la guardia presente sulla transizione stessa. Se un ticket non compare in questa lista significa che è stato archiviato e che il pagamento può essere quindi rifiutato. Perché il tutto funzioni correttamente è stato settato il marking iniziale del place *Replicated Opened Tickets* con la lista vuota.

REQUISITO 8

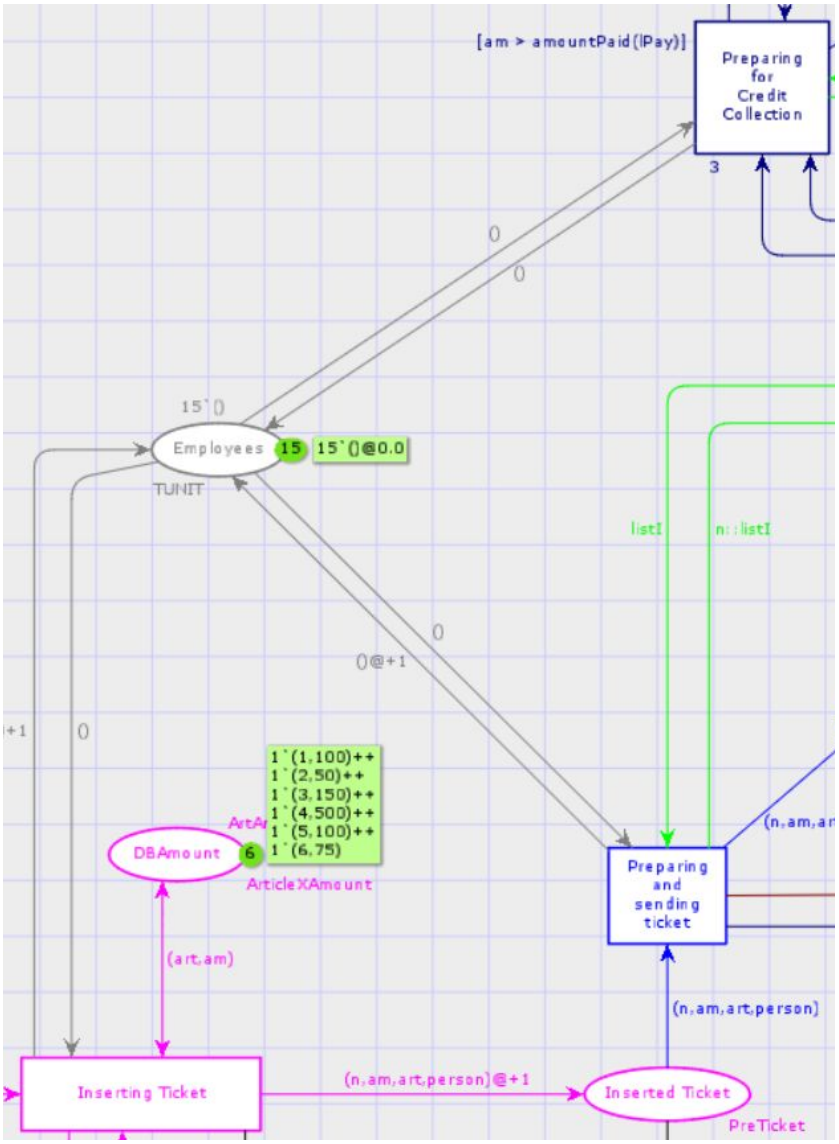


Fig. 8: Parte di rete che modella l'ottavo requisito

DESCRIZIONE

L'intero processo di gestione delle multe è gestito da 15 impiegati. Questa caratteristica è stata modellata tramite il place *Employees* avente come marking iniziale 15 token nulli, rappresentanti gli impiegati disponibili. Nel nostro caso vi sono solamente 3 attività che richiedono un impiegato, mentre tutte le altre vengono completate d'ufficio. Questa richiesta è stata modellata facendo consumare alle transizioni che implementano queste 3 attività un token da *Employees*. In questo modo le transizioni saranno attive solamente se ci sono impiegati disponibili, altrimenti per essere eseguite dovranno attendere la disponibilità degli impiegati. Poiché *Insert Ticket* e *Preparing and sending ticket* richiedono un giorno per essere eseguite, l'impiegato verrà riprodotto sul place *Employees* con un ritardo di un'unità di tempo. Per quanto riguarda *Preparing for Credit Collection* l'impiegato viene riprodotto senza ritardo, in quanto l'attività non richiede tempo per essere completata.

REQUISITO 9

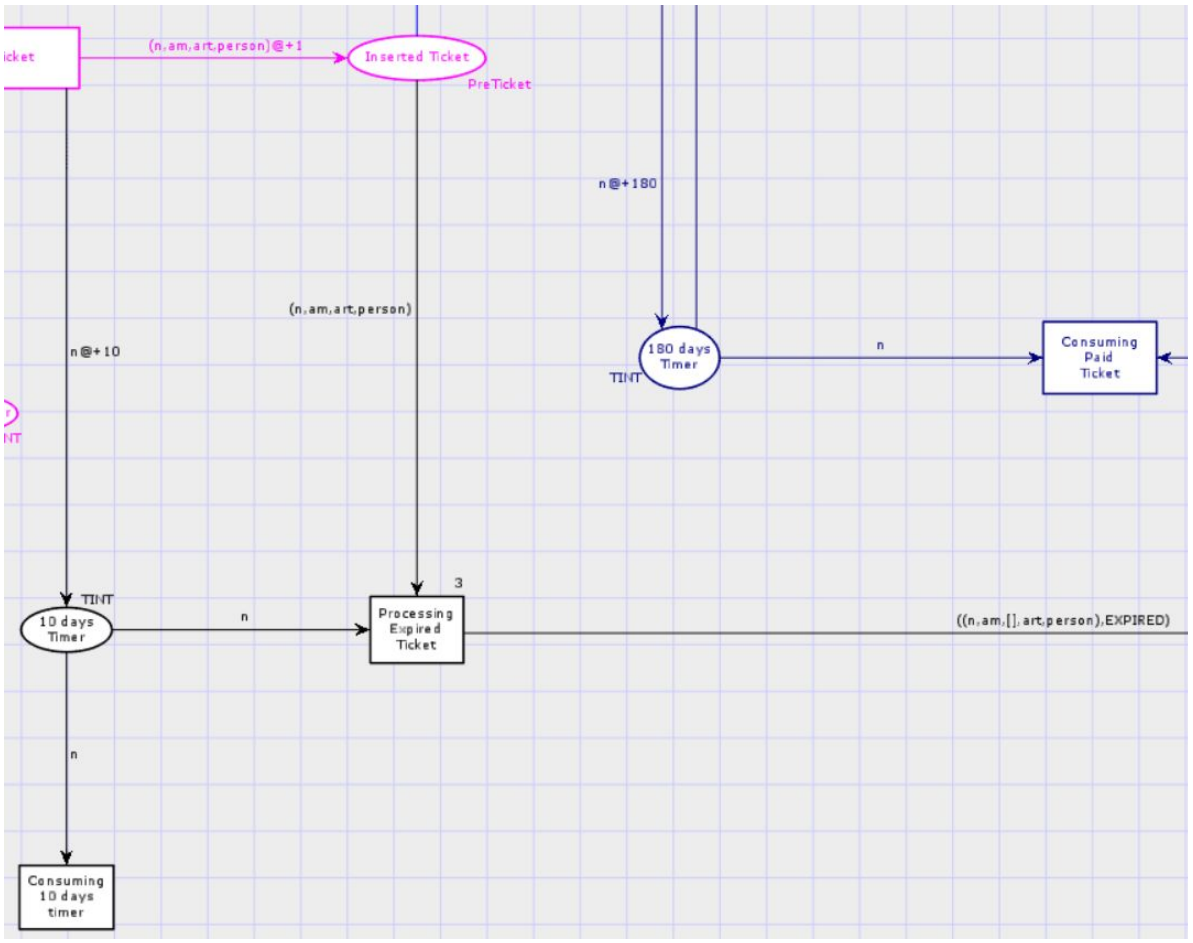


Fig. 9: Parte di rete che modella il nono requisito

DESCRIZIONE

Ogni volta che un ticket viene inserito, può richiedere un massimo di 10 giorni per essere inviato. Se non viene inviato entro questo intervallo temporale allora il ticket scade. Questa richiesta è stata modellata producendo un token (contenente il codice del ticket) con un *delay* di 10 giorni sul place *10 Days Timer* ogni volta che un ticket viene inserito nel sistema. La transizione *Processing Expired Ticket* si occupa di scartare i ticket che non vengono inviati entro il tempo prestabilito. Infatti non appena il timer scade (il token su *10 days Timer* diventa disponibile), se il token è ancora presente all'intero di *Inserted Ticket*, la transizione diventa *enable* e consuma il token producendo un token su *Archived Tickets* con ragione EXPIRED. Per la transizione è stata impostata un'alta priorità (3), in questo modo sarà sempre la prima ad essere eseguita tra tutte le transizioni *enable*. Questo predilige lo scarto di un ticket rispetto al suo invio (transizione *Preparing and sending ticket*) nel momento in cui scade il timer, nel caso in cui entrambe le transizioni fossero *enable*.

RETE COMPLETA

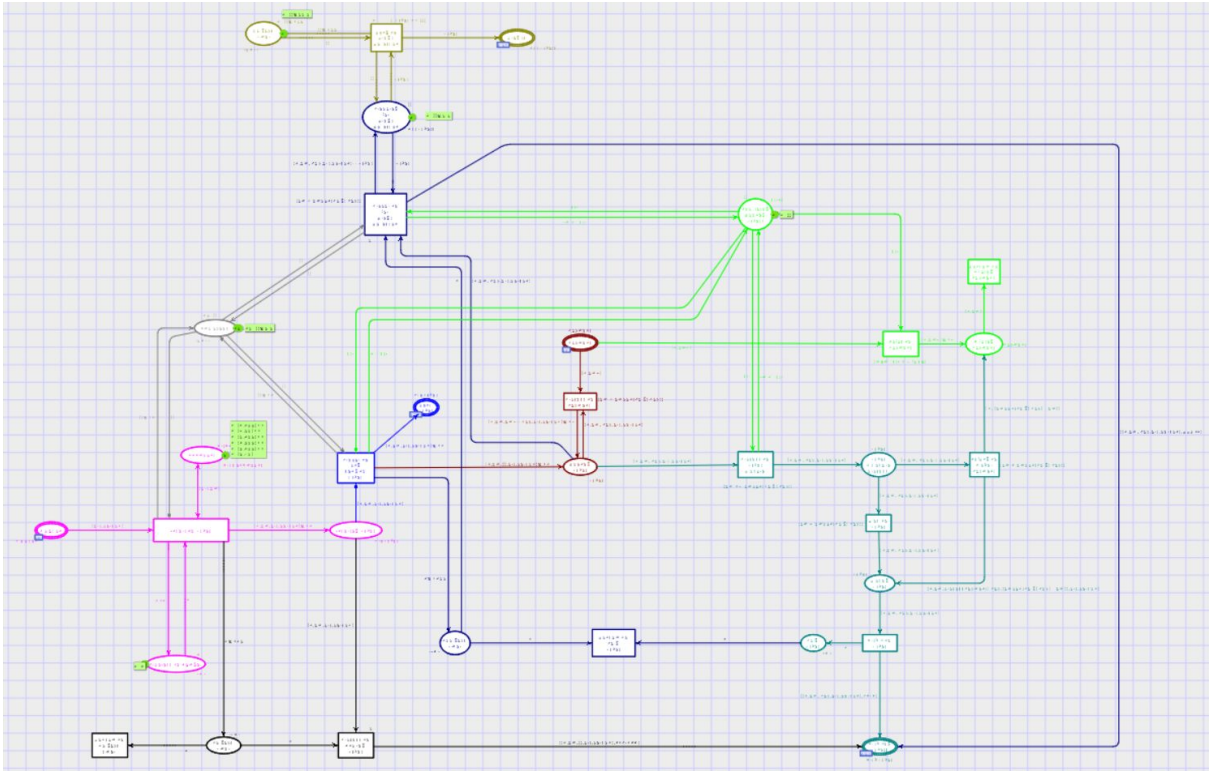


Fig. 10: Rete completa