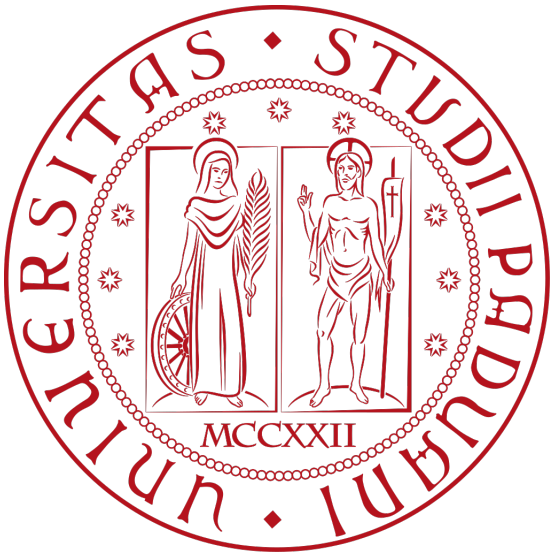
Advanced Topics in Computer Science

Report for Part#3 of Assignment (2018/2019)



Università degli studi di Padova
Laurea Magistrale in Informatica

YAWL Model - Building & Deployment

Studenti	**Fasolato Francesco** (1177742)
	Zecchin Giacomo (1179034)
Docente	**De Leoni Massimiliano**

May 28, 2019

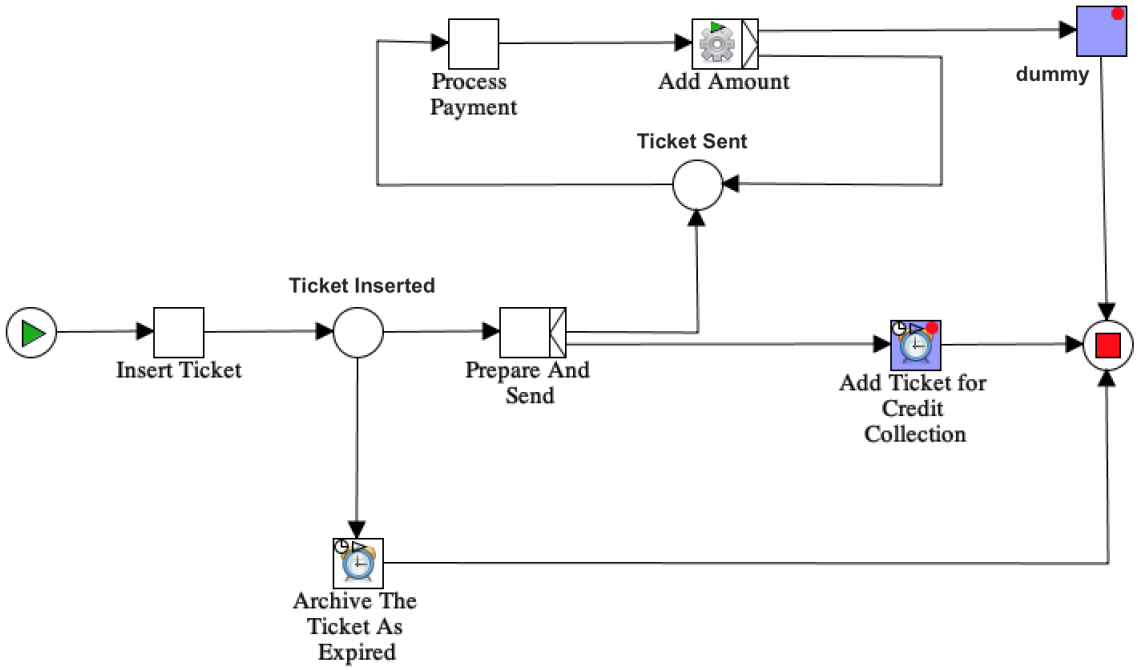


Figura 1: Road-Traffic Ticket Management in YAWL

1 Variabili e ruoli del modello

In questa sezione verranno elencate tutte le variabili utilizzate all'interno della rete ed i ruoli (con relativo personale) disponibili.

1.1 Variabili

Tralasciando le variabili di rete *Name*, *Address* e *ArtNumber*, che vengono utilizzate per identificare univocamente una multa, andremo a descrivere le variabili fondamentali per il corretto funzionamento del processo (tutte le variabili di tipo numerico sono sempre inizializzate a 0). La variabile *ViolationAmount*, inserita nella rete dal task `Insert Ticket`, rappresenta la somma totale che il trasgressore deve pagare. *AmountPaid* invece, è la variabile che rappresenta la somma versata dal trasgressore. La ricezione di questo pagamento comporta l'esecuzione del task `Process Payment`. La variabile *SumPaid* riporta, infine, la somma dei pagamenti parziali ricevuti.

N.B.: la variabile di rete *SumPaid*, all'interno del task `Process Payment`, assume il nome locale *PartialPayment*. Abbiamo cambiato il nome della variabile poiché questa viene mostrata all'utente all'interno del proprio pannello di controllo e quindi risultava essere più "user-friendly".
Inoltre, il tast `dummy` (non presente in tabella 1) non è automatico e non contiene variabili associate: è stato creato appositamente per introdurre la Cancelation Region, che verrà descritta nella prossima sezione.
La tabella seguente presenta tutte le variabili presenti nella rete, raggruppate per task:

Task	Variabili	Tipo	Scope
Insert Ticket	ArtNumber	int	Output
	Name	string	Output
	Address	string	Output
	ViolationAmount	double	Output
Archive The Ticket As Expired	(Automated)		
Prepare And Send	ArtNumber	int	Input
	Name	string	Input
	Address	string	Input
	ViolationAmount	double	Input
Process Payment	Name	string	Input
	Address	string	Input
	ViolationAmount	double	Input
	PartialPayment	double	Input
	AmountPaid	double	Output
Add Amount	AmountPaid	double	Input
	ViolationAmount	double	Input
	SumPaid	double	Input
	query	string	Input
	result	double	Output (To Net Variable SumPaid)
Add Ticket For Credit Collection	(Automated)		

Tabella 1: Decomposition Variables

1.2 Ruoli

Di seguito viene riportato l'elenco degli utenti creati ed i rispettivi ruoli:

Utente	Username	Password	Ruolo
Francesco Fasolato	Franz	1234	Insert Tickets
Alessandro Bari	ABari	1234	Insert Tickets
Giacomo Zecchin	JZ	1234	Prepare And Send
Guido Santi	GS	1234	Prepare And Send
Massimiliano De Leoni	MDL	1234	Processor
Filippo Berto	bertof	1234	Processor

Tabella 2: Utenti & Ruoli

2 Descrizione del modello

Il task `Insert Ticket` permette all'operatore afferente al ruolo di "Insert Tickets" di inserire le informazioni riguardo la multa.

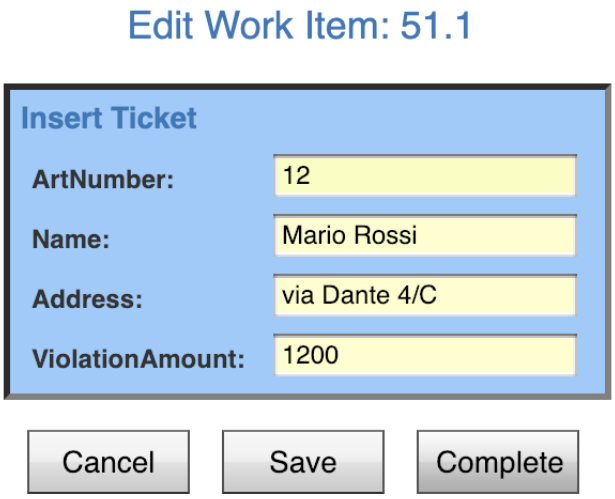


Figura 2: Front-end relativo al task `Insert Ticket`

Una volta inserite queste informazioni, un utente della rete con il ruolo di "Prepare And Send" ha a disposizione 10 giorni per spedire la multa, altrimenti questa verrà considerata scaduta ed il flusso terminerà.

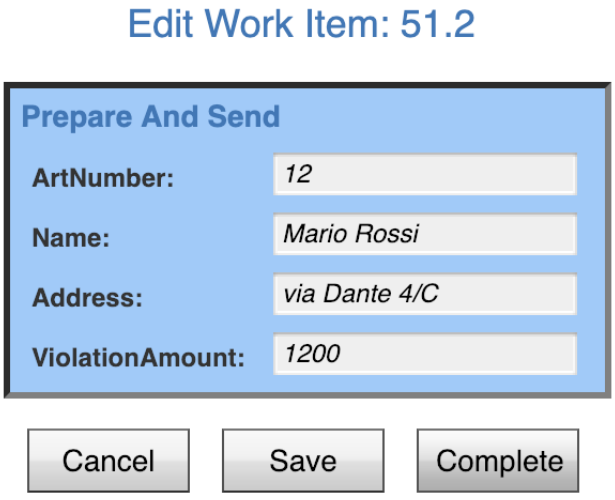


Figura 3: Front-end relativo al task `Prepare And Send`

Il caso in cui un operatore prepari ed invii la multa tramite il task `Prepare And Send`, comporterà la generazione di due token diversi attraverso uno split di tipo AND:

- Un token entra in una Deferred Choice (`Ticket Sent`). Questo implica che il task `Process Payment` verrà eseguito da un operatore di tipo "Processor" solo se si verificherà una determinata condizione. La condizione per la sua attivazione è

la ricezione da parte del sistema di un pagamento (parziale o completo). L'operatore "Processor" inserirà la variabile *AmountPaid* nella rete, questo provocherà l'esecuzione del task automatico `Add Amount`.

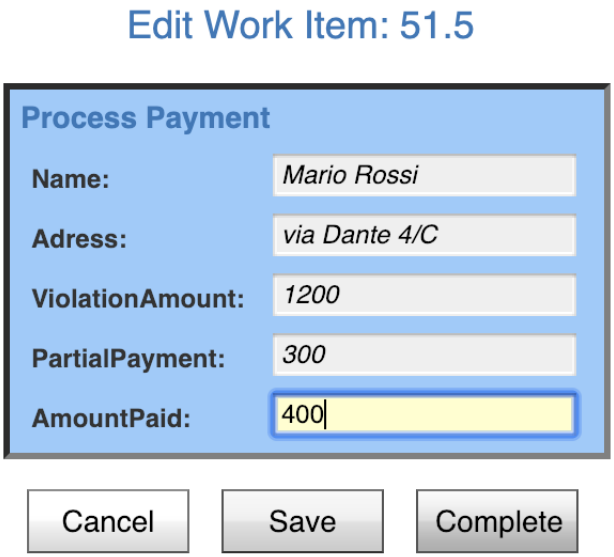


Figura 4: Front-end relativo al task `Process Payment`

`Add Amount` (contrassegnato con l'icona di un ingranaggio) contiene un codelet di tipo XQueryEvaluator che, nella parte relativa alla **query**, non farà altro che sommare il valore contenuto all'interno della variabile *AmountPaid* a quello della variabile *SumPaid*. Il **result** della query sarà salvato all'interno della stessa variabile *SumPaid* attraverso il "binding".
Infine, `Add Amount` presenta uno split di tipo XOR la cui prima condizione permette di terminare il flusso nel caso in cui la somma contenuta all'interno di *SumPaid* sia maggiore od uguale a quella totale che il trasgressore deve versare, (valore contenuto in *ViolationAmount*). Altrimenti, la condizione di default obbliga a tornare alla Deferred Choice `Ticket Sent`;

- L'altro token provoca l'esecuzione di `Add Ticket For Credit Collection`, dopo aver aspettato 180 giorni (6 mesi nella rete fornita). Questo limite temporale indica al trasgressore la scadenza per pagare la multa in toto. Dopo l'esecuzione del task il flusso termina e la multa è preparata per il "Recupero Crediti".

2.1 Timers

Nella rete sono presenti **due timer** (nei task con l'icona della sveglia):

1. Il primo è relativo al task automatico `Archive The Ticket As Expired` e provoca la sua esecuzione dopo 10 giorni da quella di `Insert Ticket`, nel caso in cui non venga eseguito il task `Prepare And Send` prima dello scadere del tempo (regolamentato dalla Deferred Choice `Ticket Inserted`);

2. Il secondo è relativo al task automatico `Add Ticket For Credit Collection` e la sua esecuzione avviene dopo 180 giorni da quella di `Prepare And Send`.

Entrambi i timer iniziano nel momento in cui viene offerta l'esecuzione del relativo task ("On offer") e scadono al termine della rispettiva durata ("After duration").

Osservazioni: Durante l'esecuzione dei test abbiamo notato che il flusso non gestisce bene il caso particolare in cui la ricezione (parziale o totale) della somma da versare da parte del trasgressore sia ricevuta al 180° giorno. È possibile che un operatore non riesca ad inserire in tempo i dati relativi all'importo e che la multa sia preparata per il "Recupero Crediti" nonostante sia stata saldata entro il tempo limite.
Perciò, secondo noi, sarebbe ragionevole incrementare la durata del timer di un giorno lavorativo, che ci sembra un limite massimo corretto per permettere agli operatori "Processor" di eseguire il task `Process Payment`.

2.2 Cancellation Regions

Nella rete sono presenti due Cancellation Regions (nei task di colore blu). Il task `Prepare And Send` presenta uno split di tipo AND, ma nella rete non sono presenti degli AND join. È stato quindi necessario introdurre due Cancellation Regions per poter consumare il token rimasto rispettivamente nel flusso proveniente dall'AND split che non termina.

1. Nel caso in cui venga eseguito il task `Add Ticket For Credit Collection` il flusso termina cancellando anche il token presente nella sua Cancellation Region, formata da:
   - `Ticket Sent`;
   - `Process Payment`;
   - `Add Amount`.
2. Nel caso, invece, in cui venga eseguito il task `dummy` (appositamente creato) il flusso termina cancellando anche il token presente nella sua Cancellation Region, formata da:
   - `Add Ticket For Credit Collection`.

3 Test eseguiti

I test seguenti sono stati svolti per verificare la correttezza della rete. I valori dei timers sono stati modificati unicamente allo scopo di effettuare i test. In particolare:

- Il timer relativo al task `Archive The Ticket As Expired` è stato diminuito da 10 giorni a 30 secondi;
- Il timer relativo al task `Add Ticket For Credit Collection` è stato diminuito da 180 giorni a 50 secondi.

Il browser utilizzato durante i test per gestire il "Pannello di Controllo" degli utenti è **Opera v. 60.0.3255.109** su un sistema operativo macOS Mojave 10.14.5 (18F132). In questa sezione verranno riportati i test eseguiti passo per passo:

- Test 1

    1. L'utente Franz accetta il task `Insert Ticket` ed inserisce i dati richiesti;
    2. Nessun utente di tipo *Prepare And Send* accetta l'offerta relativa;
    3. Dopo 30 secondi viene esegiuto il task `Archive The Ticket As Expired` ed il processo termina correttamente.

- Test 2

    1. L'utente ABari accetta il task `Insert Ticket` ed inserisce i dati richiesti;
    2. L'utente GS accetta il task `Prepare And Send` e lo completa confermando i dati inseriti entro 30 secondi;
    3. Il trasgressore non paga nessuna somma, quindi nessun utente di tipo *Processor* esegue `Process Payment`;
    4. Dopo 50 secondi viene eseguito il task `Add Ticket For Credit Collection` ed il processo termina correttamente.

- Test 3

    1. L'utente Franz accetta il task `Insert Ticket` ed inserisce i dati richiesti;
    2. L'utente JZ accetta il task `Prepare And Send` e lo completa confermando i dati inseriti entro 30 secondi;
    3. Il trasgressore paga una somma inferiore al totale richiesto, quindi il flusso torna alla Deferred Choice `Ticket Sent`;
    4. Il trasgressore paga un'altra somma, che aggiunta alla precedente **non** supera o eguaglia il totale richiesto, quindi il flusso torna alla Deferred Choice `Ticket Sent`;
    5. Scadono i 50 secondi, viene comunque eseguito il task `Add Ticket For Credit Collection` ed il processo termina correttamente.

- Test 4

    1. L'utente ABari accetta il task `Insert Ticket` ed inserisce i dati richiesti;
    2. L'utente GS accetta il task `Prepare And Send` e lo completa confermando i dati inseriti entro 30 secondi;
    3. Il trasgressore paga una somma inferiore al totale richiesto, quindi il flusso torna alla Deferred Choice `Ticket Sent`;
    4. Il trasgressore paga un'altra somma, che aggiunta alla precedente **non** supera o eguaglia il totale richiesto, quindi il flusso torna alla Deferred Choice `Ticket Sent`;
    5. Il trasgressore paga un'altra somma, che aggiunta alle precedenti, questa volta, supera o eguaglia il totale richiesto, entro i 50 secondi, quindi il processo termina correttamente.