Università
Ca'Foscari
Venezia

Corso di Laurea Magistrale

in Economia e Gestione delle Aziende

ex D.M. 270/2004

Tesi di Laurea

Le Reti di Petri e la loro applicazione nei flussi di lavoro

Relatore
Ch. Prof. Daniela Favaretto

Laureando
Michele Faganello
Matricola 870137

Anno Accademico
2021 / 2022

INDICE

INTRODUZIONE

In questo elaborato si vuole analizzare ed approfondire le Reti di Petri, uno strumento di modellazione concepito da Carl Adam Petri nella sua tesi di dottorato nel 1962. Si tratta di un tool dotato di una base matematica molta solida e formalizzata il quale dispone anche di una veste grafica facilmente comprensibile, per facilitarne la visualizzazione, che rende queste Reti uno strumento di progettazione molte potente.

Nel primo capitolo verrà illustrato questo strumento andando a definire i suoi componenti fondamentali, ovvero Posti, Transizioni, Archi e Gettoni, per poi precisare il suo funzionamento attraverso le regole di abilitazione e scatto in modo da poter comprendere la prospettiva dinamica di queste Reti. Verrà quindi fornito subito un esempio per mostrare come queste Reti permettono di modellare i sistemi e per consentire di apprezzare la loro capacità di calarsi nella realtà. Inoltre, essendo uno strumento matematico ne verrà fornita anche una rappresentazione matriciale. Successivamente verranno esaminate alcune modellistiche ricorrenti all'interno di queste reti utili per capirne il funzionamento, per andare poi e definirne le proprietà principali ed esaminando infine alcune classi particolari di Reti.

Nel secondo capitolo si vuole invece approfondire quanto questo strumento permetta di modellare sistemi reali e complessi, anche attraverso una veste grafica estesa rispetto alle reti "classiche" trattate nel capitolo uno. In particolare, verranno prese in considerazione tre estensioni di particolare rilevanza: quella del colore (o dei dati), quella del tempo e quella gerarchica che permettono di passare dalla Reti di Petri alle Reti di Petri di Alto Livello. Queste tre prevedono l'aggiunta di informazioni e condizioni logiche che vanno ad influenzare il funzionamento della rete. Come si può capire quindi con l'aggiunta dei dati, della variabile tempo e di una rappresentazione gerarchica (quindi a livelli) è evidente che la capacità di rappresentazione della realtà e quindi di sistemi reali aumenta notevolmente.

Nel terzo e ultimo capitolo si vuole invece approfondire l'applicazione delle Reti in materia di gestione dei flussi di lavoro. Inizialmente verrà descritto il motivo da cui deriva l'esistenza dei flussi di lavoro per parlare poi dei sistemi di gestione dei flussi di lavoro (WFMS) i quali vengono definiti dalla Workflow Management Coalition come "*un sistema che definisce, gestisce ed esegue completamente i flussi di lavoro attraverso l'esecuzione di software il cui ordine di esecuzione è guidato da una rappresentazione al computer della logica del flusso di lavoro"*. Successivamente verranno descritti alcuni concetti essenziali in materia di workflow, in particolare la relazione esistente tra caso, task, work-item ed activity.

Infine, tutti i concetti precedentemente esposti verranno "tradotti" nel linguaggio delle Reti di Petri ottenendo quelle che vengono definite Reti di Flusso di Lavoro che permettono la modellazione e rappresentazione dei flussi di lavoro. Dopodiché al fine di verificare la correttezza dei flussi che possono essere modellati attraverso queste Reti, intesa come assenza di errori logici che compromettano l'esecuzione del flusso, verrà definita e analizzata la proprietà di solidità.

A conclusione di questo lavoro verrà fornito un esempio, semplice ma dettagliato, di Rete di Flusso di Lavoro utilizzata per modellare un processo per la gestione degli ordini. Nel corso dell'esempio, anche grazie ad una rappresentazione grafica, verrà descritto e analizzato il flusso e il suo funzionamento.

CAPITOLO I

INTRODUZIONE ALLE RETI DI PETRI

1.1 Le Reti di Petri: definizione

Le Reti di Petri, note anche come reti posto/transizione o reti P/T, sono uno strumento di modellazione, grafica e matematica, che consente di descrivere e rappresentare la struttura di un sistema distribuito, gli eventi che in esso si verificano nell'ordine in cui si presentano e che permette inoltre di impostare equazioni di stato, equazioni algebriche e altri modelli matematici che consentono di gestire il comportamento dei sistemi.

Essendo infatti una tecnica di modellazione di sistemi discreti e dinamici descrive stati e passaggi di stato. Tali reti possono anche essere usate come mezzo di comunicazione visiva (come si vedrà nel successivo paragrafo).

Furono sviluppate da Carl Adam Petri nella sua tesi di dottorato nel 1962 presso la facoltà di Matematica e Fisica dell'Università Tecnica di Amsterdam. Di fatto con la sua teoria ha introdotto un approccio nuovo e innovativo nella concettualizzazione dei sistemi fornendo un framework che può essere applicato su numerosi tipi di sistemi discreti e utile anche per un esteso ambiente di ricerca (produzione, trasporti, informatica ecc.)[1].

Diverse sono le motivazioni che spingono al loro utilizzo. Innanzitutto, vengono proposte come strumento formale per azzerare l'ambiguità nella rappresentazione e per eseguire analisi e verifiche sul comportamento dei sistemi. In secondo luogo, forniscono una veste grafica molto intuitiva che ne semplifica l'uso e la comprensione. Infine, consentono di

[1] Half a century after Carl Adam Petri's Ph.D. thesis: A perspective on the field, Manuel Silva https://www.sciencedirect.com/science/article/pii/S1367578813000473

rappresentare concetti come la sincronizzazione di processi, il susseguirsi degli eventi, operazioni concorrenti o conflittuali e la condivisione di risorse. Tutto questo è infatti molto utile quando si parla di *workflow management.* Infatti, qualunque sistema o flusso di lavoro è costituito da un insieme di attività che può essere modellato indicando quali sono gli Stati del sistema, e questo sia prima che dopo l'esecuzione delle varie attività.

1.2 Una descrizione dello strumento

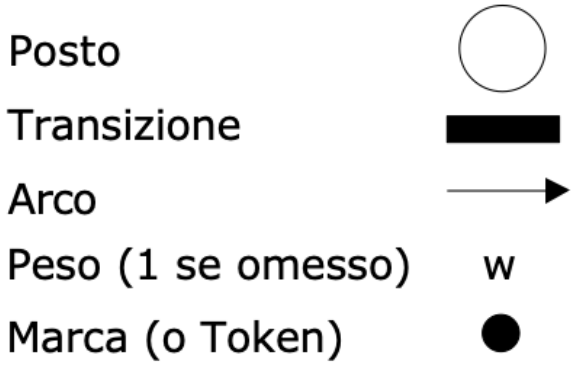
Per comprendere appieno tale strumento è fondamentale darne una descrizione[2]. Un grafo orientato, pesato e bipartito insieme ad una Marcatura iniziale è la base di partenza. La bipartizione deriva dal fatto che all'interno di questo troviamo due tipi di nodi: "Posti" e "Transizioni", che rappresentano rispettivamente gli stati e le attività della nostra rete. Questi sono collegati da Archi, rappresentanti la direzione della transizione di stato, che congiungono un Posto e una Transizione o una Transizione ed un Posto. Si capisce quindi che Nodi appartenenti alla stessa tipologia non possono essere collegati tra loro. Un Posto con un arco uscente verso una Transizione prende il nome di Pre-condizione per quella Transizione, viceversa un Posto con un arco entrante proveniente da una Transizione è detto Post-condizione di quest'ultima. Ad ogni arco viene attribuito un peso W, ma nel caso in cui il peso sia unitario l'informazione viene omessa. Per calarci maggiormente nella realtà diamo alcune interpretazioni a ciò che rappresentano Posti e Transizioni. Le pre-condizioni possono rappresentare segnali di input, buffer, risorse necessarie, condizioni. Le post-condizioni esprimono segnali di output, risorse rilasciate. Le transizioni corrispondono ad eventi, task, processori, fasi logiche o di calcolo.

[2] Petri Nets: Properties, Analysis and Application. Tadao Murata https://ieeexplore.ieee.org/document/24143

Lo stato attuale della rete/sistema viene descritto dalla “Marcatura iniziale”, in generale una marcatura è una funzione che associa ad ogni Posto un numero intero non negativo di “Gettoni”. La marcatura viene espressa attraverso un vettore M con tante componenti quanti sono i Posti nella rete e il cui valore corrisponde al numero di Gettoni presenti in ogni Posto. Sempre per dare un risvolto reale a quanto detto, quando un Posto risulta essere marcato, ovvero contiene un gettone, significa che la condizione associata a quel Posto è soddisfatta e questo può essere inteso ad esempio come disponibilità di una certa risorsa (Quando di seguito verrà introdotta la “Regola di scatto” questo sarà più chiaro).

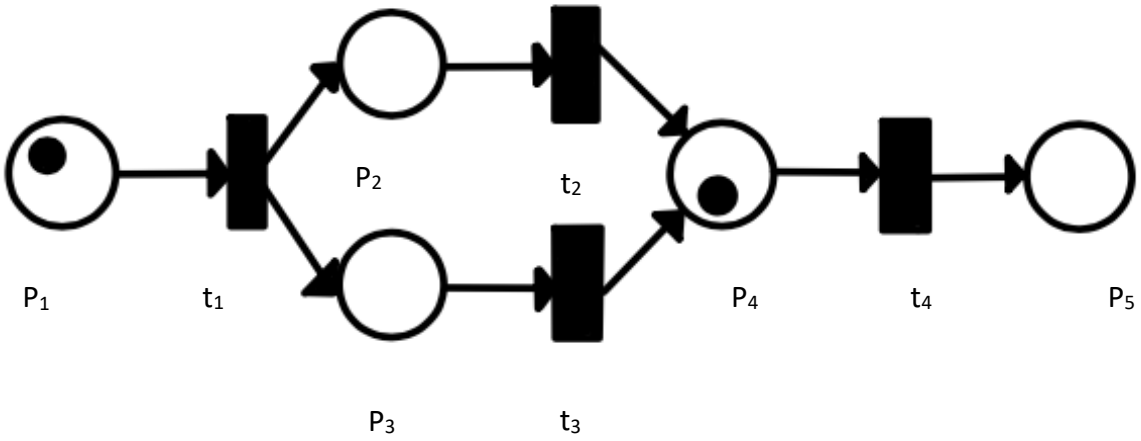
È quindi importante sottolineare come la struttura di una Rete di Petri sia fissa mentre la distribuzione dei gettoni nei posti possa cambiare.

Il tutto può anche assumere una veste grafica: i Posti sono rappresentati con un tondo, le Transizioni con un rettangolo, gli Archi da una freccia con sopra espresso il peso W e la Marcatura espressa dai Gettoni è rappresentata attraverso dei pallini neri all’interno dei posti (figura 1).



(figura 1, rappresentazione degli elementi essenziali di una Rete di Petri)

La figura 2 fornisce un esempio di Rete di Petri.



(figura 2, rappresentazione di una Rete di Petri)

Fornendo quindi una definizione formale una Reti di Petri è descritta dalla seguente quintupla (P, T, F, W, M_0), dove:

$P = \{p_1, p_2, \ldots, p_m\}$ indica l'insieme finito dei Posti;

$T = \{t_1, t_2, \ldots, t_n\}$ indica l'insieme finito (e disgiunto da P) delle Transizioni;

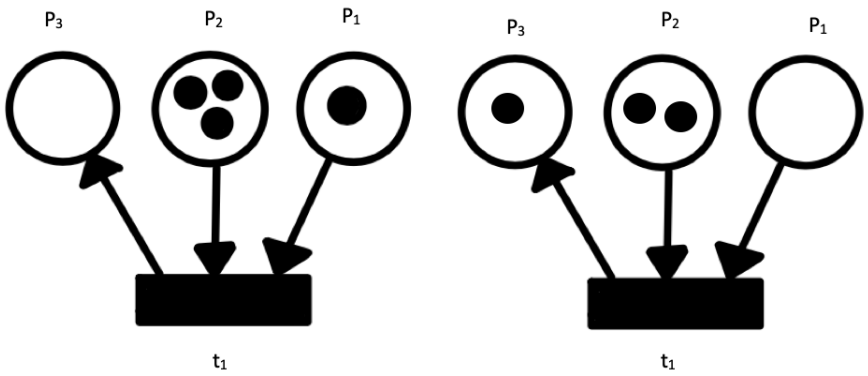
$F \subseteq (P \times T) \cup (T \times P)$ indica l'insieme degli archi in entrata e in uscita

con $P \cap T = \varnothing$ e $P \cup T \neq \varnothing$;

$W: F \rightarrow \{1, 2, 3, \ldots\}$ è la funzione che attribuisce un peso agli archi;

$M_0: P \rightarrow \{0, 1, 2, 3, \ldots\}$ indica la funzione Marcatura iniziale.

Detto ciò possiamo descrivere anche il comportamento dinamico di una Rete di Petri in termini di modifica degli stati del nostro sistema. Per fare ciò dobbiamo introdurre una regola molto importante: la regola di abilitazione e di scatto di una Transizione. Una transizione t è detta abilitata se ogni sua pre-condizione è marcata, ovvero se contiene uno o più gettoni. Una volta abilitata può scattare (a seconda che l'evento si verifichi o meno). Lo scatto di una transizione t sottrae uno o più gettoni (a seconda del peso dell'arco) da ogni pre-condizione e ne inserisce uno o più in ogni post-condizione. (figura 3)



(figura 3, abilitazione e scatto di una transizione)

La figura 3 rappresenta una semplicissima Rete di Petri con tre posti (p_1, p_2, p_3) e una transizione t_1. I Posti p_1 e p_2 rappresentano le pre-condizioni della transizione t_1, mentre p_3 è l'unica post-condizione. La marcatura iniziale è $M_0 = (1,3,0)$. Come possiamo notare dalla figura a sinistra la transizione t_1 risulta essere abilitata dato che le sue pre-condizioni contengono almeno un gettone, il che è sufficiente dato che il peso dell'arco che li collega alla transizione è w=1. La transizione t_1 può quindi scattare e come conseguenza lo scatto rimuove i gettoni dalle pre-condizioni per aggiungerli nella post-condizione, la Rete raggiunge così la marcatura $M_1 = (0, 2, 1)$ come possiamo notare nella figura a destra.

Emerge quindi come all'interno di queste Reti i Posti rappresentino delle componenti passive in quanto non possono cambiare lo stato della Rete, mentre le Transizioni sono gli elementi attivi dato che se attivate ne modificano lo stato.

Come possiamo capire dalla regola appena citata si desume che ogni Posto può contenere un numero illimitato di gettoni. Una rete di questa tipologia viene definita Rete a capacità infinita. Però per modellare dei sistemi più fedeli alla realtà è auspicabile introdurre un limite superiore al numero di gettoni che possiamo trovare in ogni posto, detto capacità K(p). Ottenendo quella che si definisce una Rete a capacità finita.

Sulla base di ciò è utile quindi rivedere la regola di cui sopra. In questo tipo di rete una transazione t è abilitata quando ogni sua pre-condizione abbia almeno un gettone e ogni sua post-condizione abbia un numero di gettoni che, con lo scattare della transizione, non

superi la capacità k(p). Tale regola con il vincolo prende il nome di regola di transizione rigida, la precedente regola è invece detta regola di transizione debole.

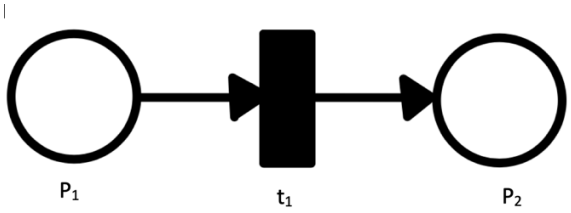
1.2.1 Un semplice esempio pratico: il distributore automatico

Di seguito si propone un semplice esempio utile a spiegare in maniera pratica le componenti delle Reti di Petri e come queste possono essere utilizzate per modellare un sistema[3], in questo caso un distributore automatico.

Il distributore è dotato di una fessura per inserire le monete e di uno scomparto in cui vengono fatti cadere i prodotti (ad esempio un succo di frutta).

Questo può essere modellato attraverso una Rete (figura 4) con due posti e una transizione i quali rappresentano:

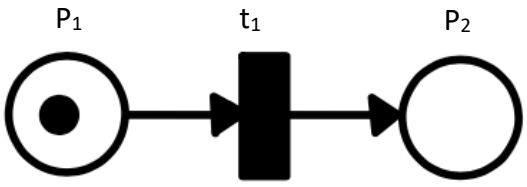
- p_1 monete inserite nella fessura;
- p_2 prodotto erogato nello scomparto;
- t_1 erogazione del prodotto.



(figura 4, modellazione del distributore attraverso Rete di Petri)

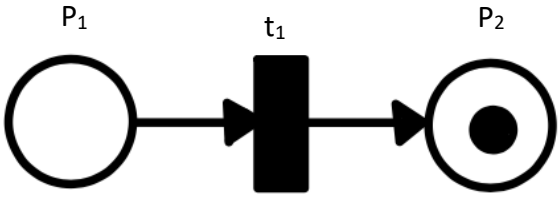
Lo stato iniziale di questo sistema è rappresentato dalla marcatura $M_0 = (1,0)$, questo sta a significare che una moneta è stata inserita nella fessura (p_1) mentre nello scomparto non è presente nessun prodotto da prelevare (p_2), come si vede nella figura 4a. Il posto rappresentante la fessura dal momento che contiene una moneta è marcato, invece lo scomparto essendo vuoto non contiene nessuna marca/gettone.

[3] Understanding Petri Nets. Modelling Techniques, Analysis Methods, Case Studies, Wolfgang Reisig https://link.springer.com/content/pdf/10.1007/978-3-642-33278-4.pdf



(figura 4a, modellazione del distributore attraverso Rete di Petri)

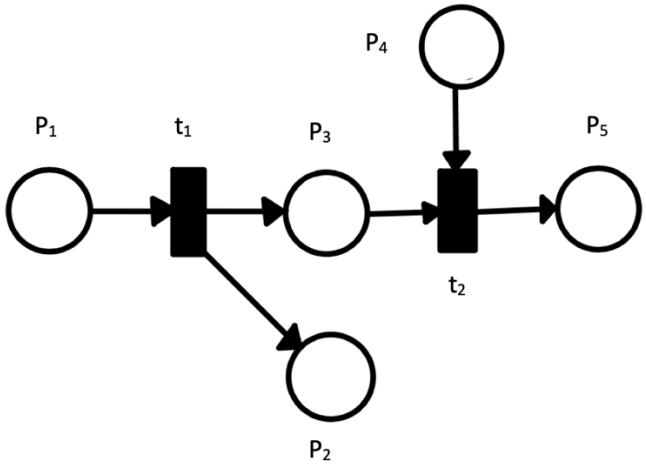
Data questa distribuzione dei gettoni il distributore può prendere la moneta ed erogare un succo di frutta. La transizione t_1 è infatti abilitata, il suo arco in ingresso parte da p_1 che contiene un gettone. La Transizione t_1 può quindi scattare, erogando il succo, e modificare lo stato del sistema giungendo ad una nuova marcatura $M_1 = (0, 1)$, (figura 4b).



(figura 4b, modellazione del distributore attraverso Rete di Petri)

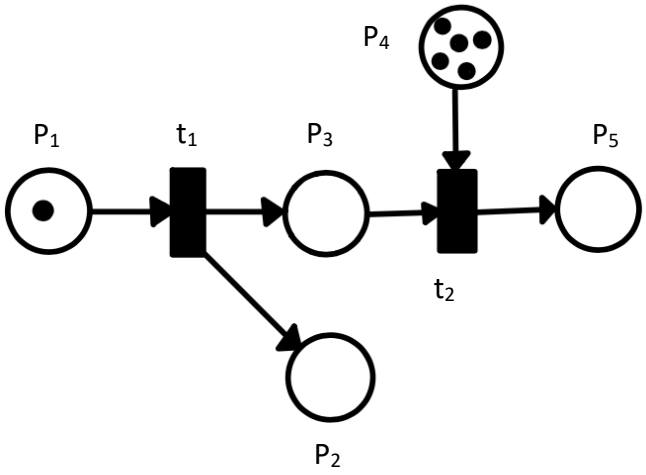
Dopo il verificarsi della transizione non c'è più nessuna moneta inserita nella fessura mentre nello scomparto è presente un prodotto, la transizione t_1 è ora disabilitata.

Come step successivo possiamo "esplodere" il contenuto del distributore andando a mettere in evidenza la sua parte interna che contiene diverse componenti aggiuntive come il magazzino per i succhi, la cassa per le monete e un segnale che rappresenta una parte fondamentale nella gestione dell'erogazione. Questo viene rappresentato attraverso una rete con cinque posti e due transizioni (figura 5) così descritti: p_1 monete inserite, p_2 cassa per le monete inserite, p_3 segnale per procedere all'erogazione, p_4 magazzino per i succhi, p_5 prodotto erogato nello scomparto, t_1 moneta accettata e t_2 erogazione del prodotto.



(figura 5, modellazione del distributore “esploso” attraverso Rete di Petri)

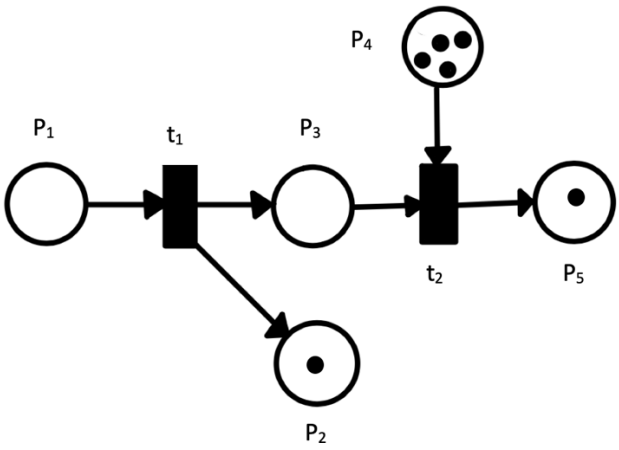
Inizialmente la Rete si trova nella marcatura M_0 = (1, 0, 0, 5, 0), troviamo quindi una moneta inserita nella fessura e cinque succhi disponibili nel magazzino (figura 5a).



(figura 5a, modellazione del distributore “esploso” attraverso Rete di Petri)

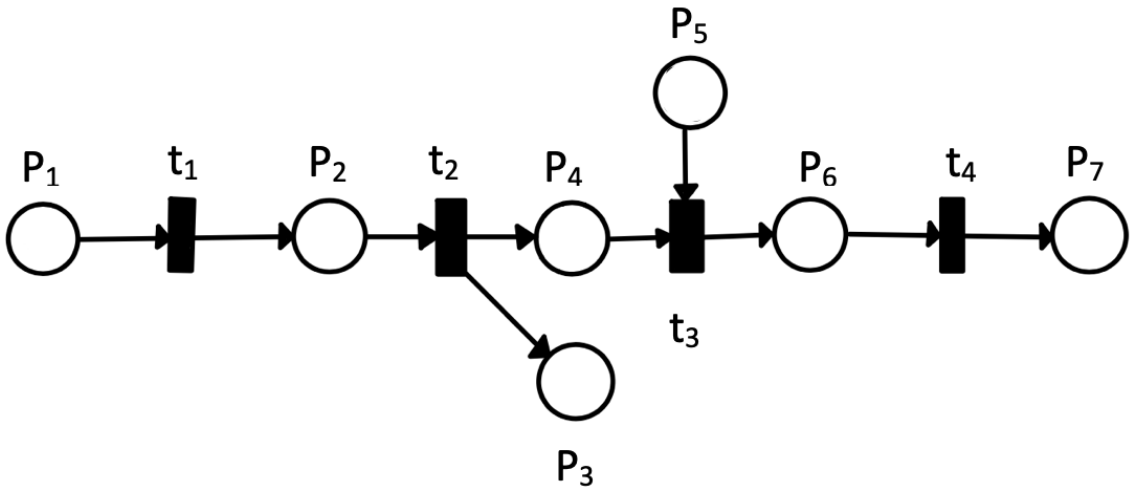
Sulla base di questa marcatura la transizione t_1 è abilitata e quindi può accettare la moneta inserita. Con l’accettazione della moneta t_1 può scattare e questo implica che la moneta accettata finirà nella cassa e che verrà generato contestualmente un segnale di input per l’erogazione del prodotto (con questo scatto compariranno quindi un gettone in p_2 e uno in p_3). Così facendo anche la transizione responsabile dell’erogazione, t_2, risulterà abilitata essendo le sue pre-condizioni, p_3 e p_4, marcate. Infatti, il segnale di input è stato attivato e ci sono succhi disponibili nel magazzino. Con lo scattare di t_2 il segnale di input viene spento e un succo viene prelevato dal magazzino ed erogato nello scomparto (p_5).

Il sistema raggiungerà così la marcatura M_1 = (0, 1, 0, 4, 1) che indica la presenza di una moneta nella cassa, di quattro succhi nel magazzino e di un prodotto erogato nello scomparto (figura 5b).



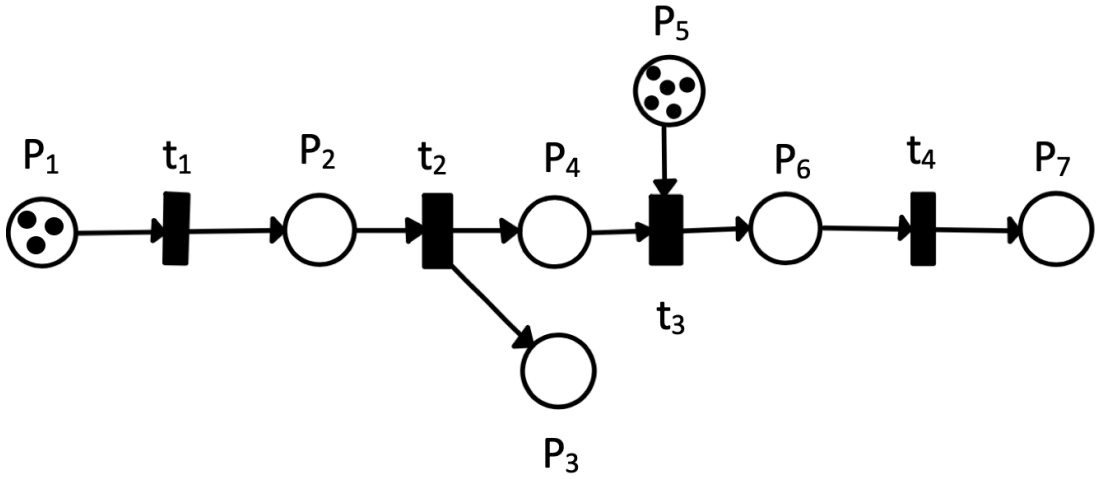
(figura 5b, modellazione del distributore “esploso” attraverso Rete di Petri)

Il distributore così modellato è un sistema chiuso, ciò che manca sono le interazioni con l’ambiente esterno, serve qualcuno che inserisca le monete e che prelevi il prodotto una volta che questo è stato erogato. Possiamo modellare ciò aggiungendo alla precedente rete un posto e una transizione all’inizio e alla fine (come mostrato dalla figura 6) ottenendo quindi una rete con sette posti e quattro transizioni con la seguente descrizione: p_1 cliente pronto ad inserire la moneta,p_2 moneta inserita, p_3 cassa per le monete inserite, p_4 segnale per procedere all’erogazione, p_5 magazzino per i succhi, p_6 prodotto erogato nello scomparto, p_7 cliente pronto a consumare, t_1 inserire la moneta, t_2 moneta accettata, t_3 erogazione del prodotto e t_4 prendere succo dallo scomparto.



(figura 6, modellazione del distributore con interazioni esterne attraverso Rete di Petri)

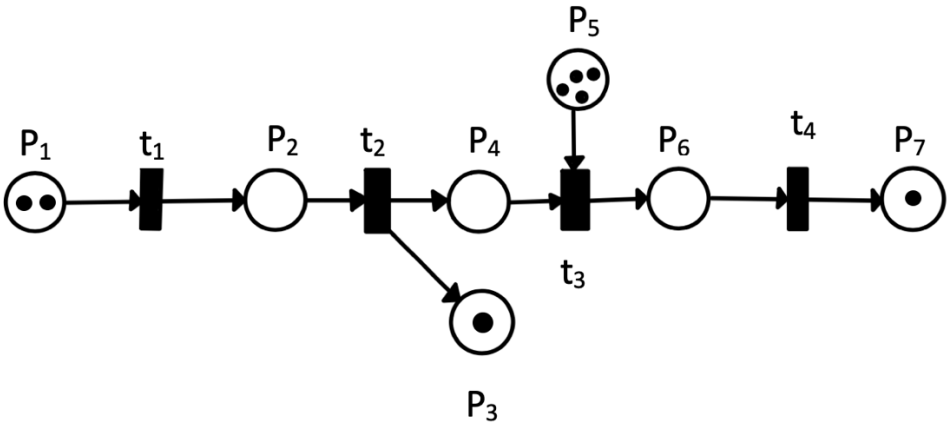
Possiamo vedere come nello stato iniziale (figura 6a), espresso dalla marcatura M_0 = (3, 0, 0, 0, 5, 0, 0), ci siano tre clienti pronti ad inserire la moneta e cinque succhi nel magazzino del distributore.



(figura 6a, modellazione del distributore con interazioni esterne attraverso Rete di Petri)

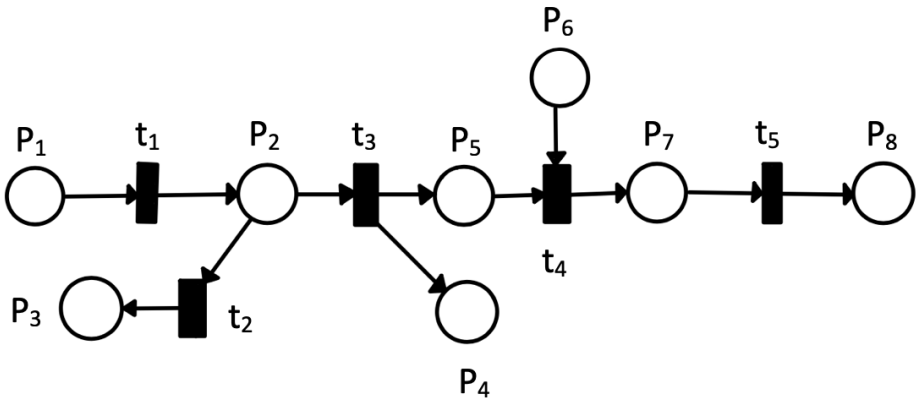
La transizione t_1 è abilitata dato che ci sono ben tre clienti pronti ad inserire la moneta nel distributore. Con il suo scatto la transizione rimuove un gettone da p_1 e lo inserisce in p_2, quindi ora c'è una moneta inserita. A questo punto il funzionamento della rete è il medesimo descritto sopra. Veniamo dunque alla parte finale, dopo i vari passaggi troveremo un prodotto erogato nello scomparto (p_6 sarà marcato) e di conseguenza verrà abilitata la transizione t_4 che potrà scattare e quindi il prodotto verrà preso dallo scomparto e finalmente

sarà nelle mani del cliente pronto a consumarlo. Verrà quindi raggiunto lo stato finale M_1 = (2, 0, 1, 0, 4, 0, 1), come mostrato dalla figura 6b



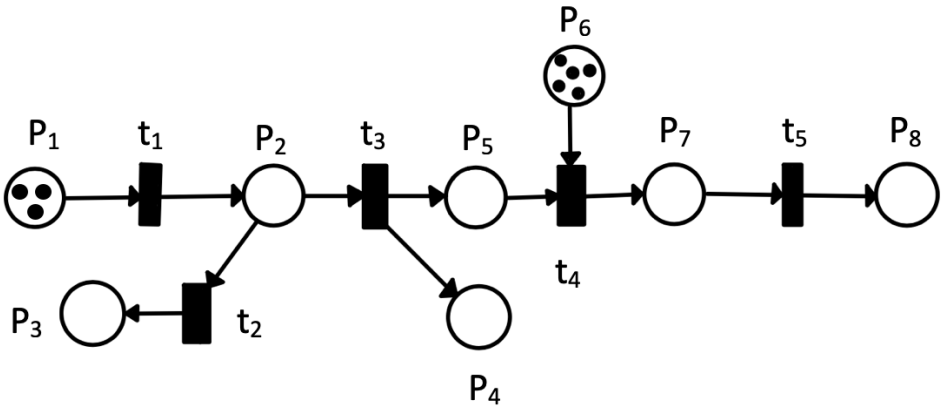
(figura 6b, modellazione del distributore con interazioni esterne attraverso Rete di Petri)

Infine, il distributore potrebbe dover restituire le monete all'ambiente esterno per alcuni motivi (per esempio monete difettose o false oppure lo stesso cliente potrebbe voler indietro le monete) ma si tratta di dettagli che a noi non interessa modellare, è sufficiente che il distributore o le accetti o le ridia indietro. Per fare ciò è sufficiente modificare la Rete come nella figura 7 qui sotto, creando una rete con otto posti e quattro transizioni, nella quale: p_1 cliente pronto ad inserire la moneta, p_2 moneta inserita, p_3 moneta restituita, p_4 cassa per le monete inserite, p_5 segnale per procede all'erogazione, p_6 magazzino per i succhi, p_7 prodotto erogato nello scomparto, p_8 cliente pronto a consumare, t_1 inserire la moneta,t_2 restituire moneta, t_3 moneta accettata, t_4 erogazione del prodotto, t_5 prendere succo dallo scomparto.



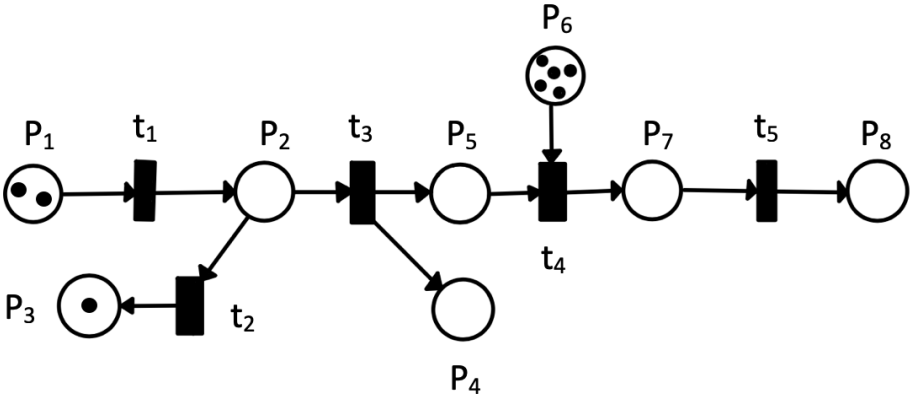
(figura 7, rappresentazione finale del distributore attraverso Rete di Petri)

La situazione inziale della rete (figura 7a) è descritta dalla marcatura M_0 = (3,0, 0, 0, 0, 5, 0, 0).



(figura 7a, rappresentazione finale del distributore attraverso Rete di Petri)

Nella situazione di partenza la transizione t_1 risulta abilitata e quindi con il suo scatto la moneta viene inserita (compare quindi un gettone in p_2). Possiamo quindi notare che come conseguenza di ciò vengono abilitate due transizioni differenti: t_3 moneta accettata oppure t_2 restituire moneta. Vediamo quindi che lo scatto di una delle due disabilita l'altra. Questo nelle Reti di Petri viene chiamato conflitto (approfondiremo queste modellistiche nel prossimo paragrafo). Nel caso in cui la moneta venga accettata l'evoluzione della rete è la stessa di cui sopra, nel caso in cui invece sia rifiutata la transizione che scatta è t_2, restituire moneta. Con lo scattare di questa transizione la moneta inserita (p_2) viene restituita e quindi compare un gettone in p_3. Questa situazione è descritta dalla marcatura M_1 = (2, 0, 1, 0, 0, 5, 0, 0), come si nota dalla figura 7b si cui sotto.



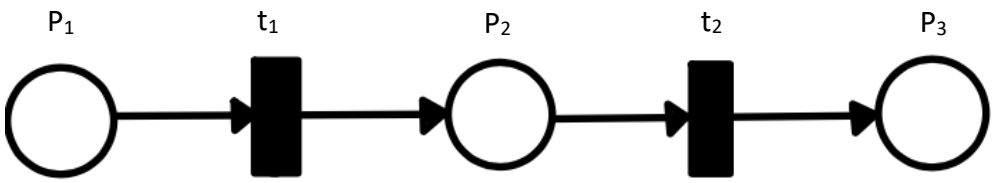
(figura 7b, rappresentazione finale del distributore attraverso Rete di Petri)

Grazie a questo esempio siamo quindi riusciti a modellare le componenti essenziali della struttura e del comportamento di un distributore automatico come una Rete di Petri.

1.3 Strutture modellistiche nelle Reti di Petri

Essendo le Reti di Petri uno strumento di modellazione efficiente in questo paragrafo vengono analizzate alcune situazioni di base che si incontrano molte volte durante la modellazione di un sistema reale. È quindi necessario introdurre alcune modellistiche fondamentali utili a capirne il funzionamento e che permettono di rivelare il potere di modellazione e la facilità di rappresentazione di tali reti [4]. Analizziamo quindi i concetti di: sequenza, conflitto (strutturale ed effettivo), concorrenza (strutturale ed effettiva), sincronizzazione, inizio concorrenza e confusione.

Due transizioni t_1 e t_2 sono in sequenza se la prima precede la seconda e se solo lo scattare di t_1 abilita t_2 (figura 8). Esiste quindi un vincolo di precedenza tra le due transazioni (attività).

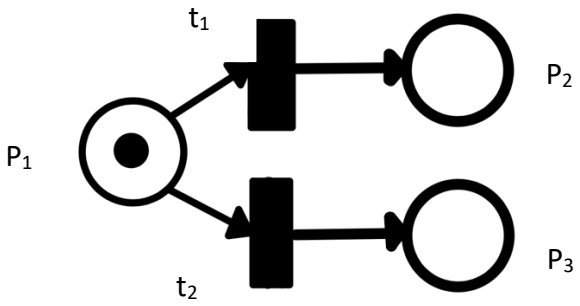


(figura 8, struttura modellistica: sequenza)

Sono invece in conflitto strutturale due transizioni t_1 e t_2 nel caso in cui queste hanno almeno una pre-condizione in comune. Sono inoltre in conflitto effettivo qualora data una certa Marcatura m risultino entrambe abilitate e il numero di gettoni che le loro precondizioni contengono non è sufficiente per farle scattare entrambe (figura 9). Di conseguenza lo scatto di una transizione disabilita l'altra.

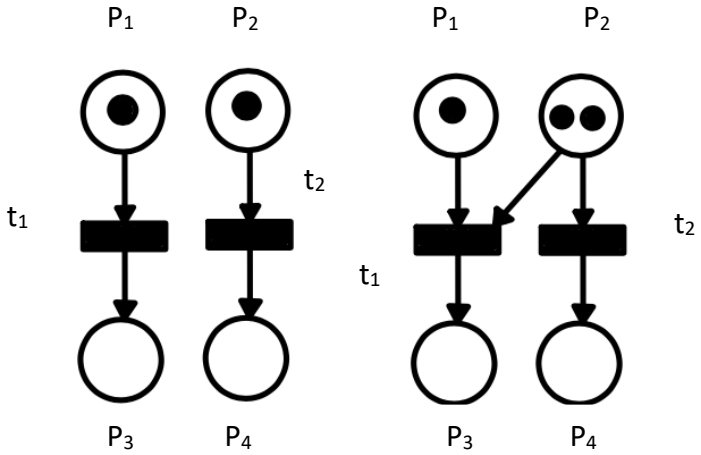
[4] A Study of Petri Nets. Modelling, Analysis and Simulation https://abhishekhalder.bitbucket.io/PetriNetReport.pdf

Possiamo quindi facilmente dedurre che la prima tipologia di conflitto deriva dalla struttura della rete, mentre il secondo dipende anche dalla marcatura.



(figura 9, struttura modellistica: conflitto)

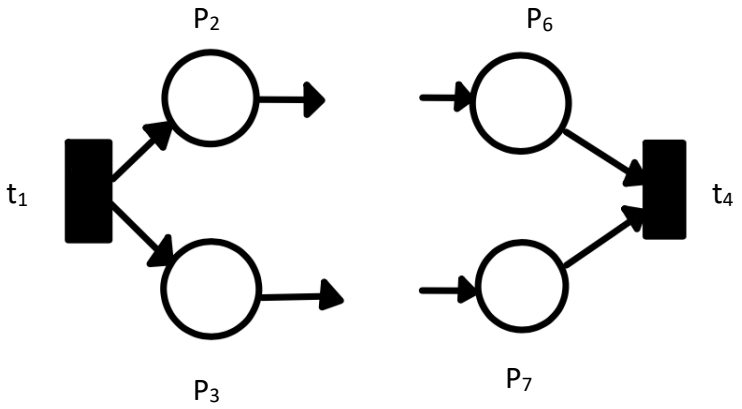
Se due transizioni t_1 e t_2 non condividono nessuna pre-condizione, e quindi lo scatto di una delle due non disabilita l'altra, si dicono essere in concorrenza strutturale. Invece quando le due transizioni condividono delle pre-condizioni ma sussiste una marcatura m tale per cui se una delle due scatta l'altra non viene disabilitata si dice che queste sono in concorrenza effettiva. Possiamo infatti notare nella figura 10 qui sotto riportata che le transizioni T1 e T2 condividono come pre-condizione P2, però dal momento che il peso degli archi che collegano il posto alle transizioni è unitario e dato che P2 contiene due gettoni, entrambe le transizioni risultano abilitate.



(figura 10, Struttura modellstica: concorrenza)

Esistono inoltre due costrutti legati al concetto di cui sopra: la sincronizzazione, che si verifica quando più posti abilitano la stessa transizione (figura 11b), e l'inizio concorrenza,

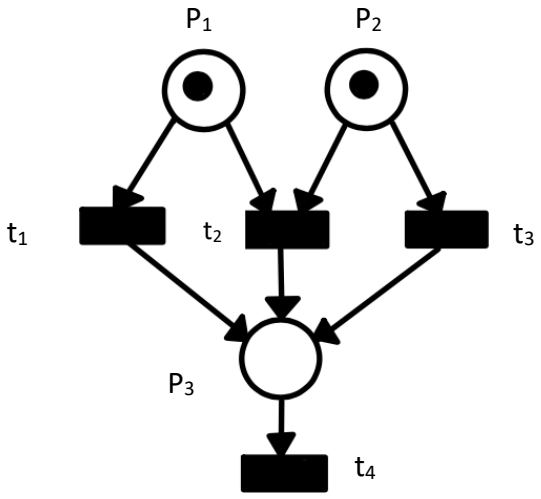
ovvero quando due post-condizioni ricevono gettoni allo scattare di una stessa transizione (figura 11a). Detto questo, in generale le sequenze concorrenti si attivano a seguito dello scatto di una transizione di Inizio concorrenza. Di contro la concorrenza si conclude attraverso lo scatto di una transizione di sincronizzazione.



(figura 11a, inizio concorrenza)

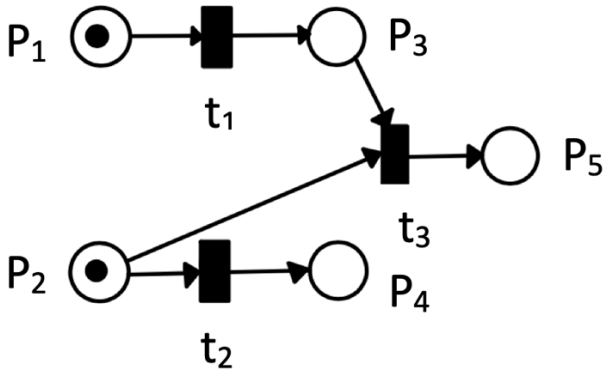
(figura 11b, sincronizzazione)

Infine, parliamo della confusione, che si verifica quando il conflitto e la concorrenza esistono simultaneamente. In tale caso non si può sapere se è necessario oppure no risolvere un conflitto nel passare allo stato successivo della rete (marcatura successiva). Infatti, nella figura 12 possiamo notare come t_1 e t_3 sono tra loro in concorrenza, ma sono anche entrambe in conflitto con t_2 e questo appunto genera confusione.



(figura 12, struttura modellistica: confusione)

Possono verificarsi inoltre due tipi di confusione. Quella simmetrica in cui due transazioni son tra loro concorrenti e allo stesso tempo sono in conflitto con un'altra transizione (come nella figura sopra). La confusione asimmetrica (figura 13) invece si verifica quando due transizioni t_1 e t_2 sono in concorrenza e se con l'attivarsi di una delle due, per esempio t_1, l'altra entra in conflitto con un'altra transizione, t_3. Quindi si verifica quando un posto della rete è pre-condizione per un set di transizioni ma esiste un altro posto della rete che è pre-condizione di un sottoinsieme di quelle transizioni.



(figura 13, struttura modellistica: confusione asimmetrica)

1.4 Rappresentazione matriciale delle Reti di Petri

Le Reti di Petri possono essere anche rappresentate graficamente tramite delle matrici.

Per fare questo è necessario introdurre alcuni concetti[5]:

- matrice di ingresso;
- matrice d'uscita;
- matrice d'incidenza;
- vettore della marcatura;
- sequenza degli scatti.

[5] Automazione industriale: Controllo logico con Reti di Petri
http://www.schoolofnerd.it/sites/default/files/ControlloLogicoRetiPetri_Parte2.pdf

Sempre attraverso questa rappresentazione è possibile inoltre descrivere la dinamica di evoluzione di una rete in termini di equazione di stato.

La matrice di ingresso I è una matrice con tante righe quanti sono i Posti nella rete e tante colonne quante sono le Transizioni all'interno della rete. Ogni suo elemento (k, j) vale quanto il peso dell'arco W che collega il Posto k alla Transizione j (vale zero nel caso in cui l'arco non esista). Tale matrice viene definita nel seguente modo:

$I_{|P|,|T|}$ con $I(k, j)=W(p_k, t_j)$, $\forall$ $(p_k, t_j)\in F$ e $I(k, j)= 0$, $\forall$ $(p_k, t_j) \notin F$

La matrice d'uscita O, come la precedente ha tante righe quanti sono i Posti nella rete e tante colonne quante sono le Transizioni all'interno della rete. In questo caso però ogni elemento che la compone ha un valore pari al peso dell'arco che unisce la Transizione k e il Posto j. Possiamo quindi definire così la matrice O:

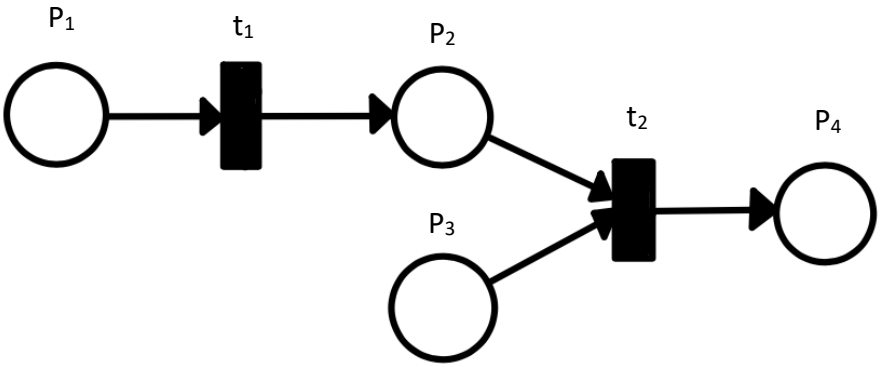
$O_{|P|,|T|}$ con $O(k, j)=W(t_j, p_k)$ $\forall$ $(t_j, p_k) \in F$ e $O(k, j)= 0$, $\forall$ $(t_j, p_k) \notin F$

In pratica queste due matrici ci permettono di capire la topologia della rete riportando il peso degli archi che collegano le pre-condizioni e le post-condizioni alle transizioni. Si tratta di dati necessari al fine di poter conoscere quali transazioni sono abilitate dalla marcatura e per studiare le conseguenze dello scatto delle transizioni stesse.

Grazie a queste due matrici è possibile costruire la matrice d'incidenza C, anch'essa dotata di tante righe quanti sono i Posti nella rete e tante colonne quante sono le Transizioni all'interno della rete. Viene quindi definita come: C = O – I.

Di seguito viene fornito un semplice esempio in cui a partire dalla struttura della rete vengono ricavate le matrici I, O e C.

La struttura della rete con quattro posti e due transizioni viene riportata qui sotto nella figura 14.

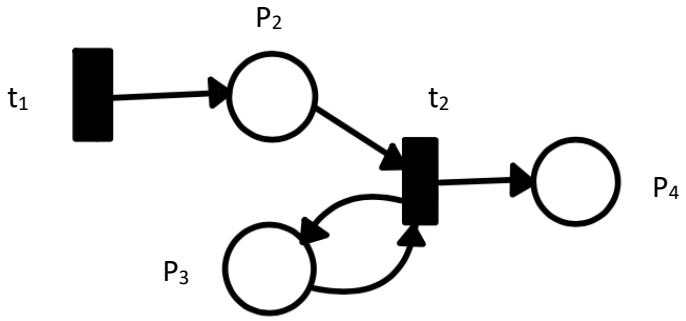


(figura 14, struttura della Rete)

Dalla Rete di cui sopra ricaviamo le seguenti matrici:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \qquad O = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \qquad C = O - I = \begin{pmatrix} -1 & 0 \\ 1 & -1 \\ 0 & -1 \\ 0 & 1 \end{pmatrix}$$

Possiamo fare subito un'osservazione al riguardo. È importante sottolineare come la matrice C non contenga, in alcuni casi, esattamente tutte le informazioni contenute nelle matrici di Ingresso e di Uscita. Infatti nel caso in cui un posto e una transizione fossero collegati da due archi di uguale peso ma direzione opposta (il cosiddetto Autoanello o Autoloop, figura 15) nella matrice di Incidenza quell'elemento assumerebbe valore zero. Ovvero come se il posto e la transizione in questione non fossero collegati da nessun arco. Infatti come possiamo notare dalla figura sottostante il Posto p_3 è allo stesso tempo precondizione e post condizione della transizione t_2. Nelle matrici di Ingresso e di Uscita questi collegamenti assumerebbero il valore 1 e dato che vale la regola C = O – I, nella matrice di incidenza il posto assumerebbe valore zero.



(figura 15, Autoanello)

Di contro per una rete pura, cioè priva di auto-anelli, le matrici I e O prevedono elementi non nulli in posizioni reciprocamente esclusive. In questo caso quindi l'uso e le informazioni contenute nella matrice C sono analoghi all'uso delle matrici di Ingesso e Uscita da cui poi deriva quella di incidenza. È quindi possibile ricavare direttamente la matrice di incidenza, con tante righe quante sono i Posti e tante colonne quante sono le Transizione, in cui ogni suo elemento (k, j) vale quanto il peso dell'arco che collega il Posto alla transizione ma con il segno meno davanti, o quanto il peso dell'arco che collega la Transizione al Posto.

Il vettore della marcatura m è un vettore colonna di dimensione |P| (cioè il numero di posti nella rete) le cui componenti sono numeri interi corrispondenti al numero di gettoni collocati in ogni posto (ovvero la marcatura della rete). Possiamo definire così questo vettore: $m = [m_1, m_2, \ldots, m_{|p|}]$ con $m_i = M(p_i)$, $i = 1,2, \ldots, |P|$.

La sequenza degli scatti, σ, abilitata in una data marcatura M_0, corrisponde ad una sequenza di transizioni t_i, con $1 \leq i \leq n$, tali per cui t_i è abilitata in M_0 e lo scatto conduce ad una marcatura in cui t_{i+1} è abilitata. Il vettore associato alla sequenza di scatti è un vettore riga di dimensione |T| (ovvero il numero di transizioni nella Rete), i cui componenti σ_i corrispondono al numero di quante volte una transizione t_i è scattata nella sequenza.
Ovviamente non tutte le sequenze di transizioni sono una sequenza di scatti. Affinché questo possa essere vero è necessario trovare una sequenza di marcature che abiliti le transizioni della sequenza. La sequenza di marcature ovviamente non è automatico che

esista. Quando quest'ultima esiste se dice che il vettore degli scatti è ammissibile. Viceversa, si dirà che non è ammissibile.

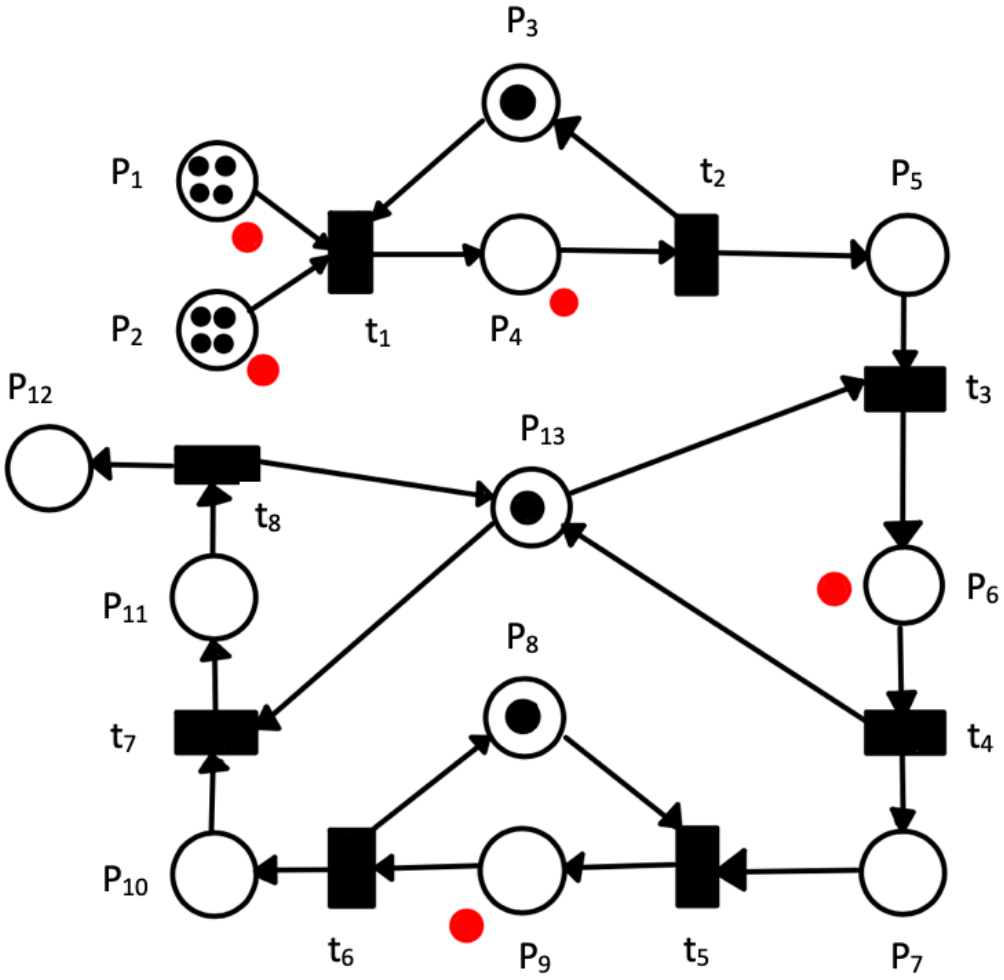
Grazie a quanto detto sopra siamo in grado di formalizzare in maniera molta compatta l'evoluzione di una Rete di Petri attraverso un'equazione di stato. Così chiamata perché permette di calcolare lo stato successivo di una rete (cioè la sua marcatura) a partire da una marcatura iniziale e dall'evento che si verifica (ovvero lo scattare delle transizioni).
Quindi partendo dalla marcatura M_0 (marcatura iniziale) di una certa Rete con matrice di incidenza C e applicando una sequenza di scatti σ ammissibile è possibile raggiungere la marcatura successiva M_1. Da cui la formula $M_1 = M_0 + C\sigma^t$
Di seguito viene fornita una dimostrazione attraverso un semplice esempio.

Esempio: attraverso l'uso dell'equazione fondamentale dimostriamo se esiste un vettore degli scatti σ ammissibile per la Rete P/T nella figura 16 (nella quale la marcatura iniziale M_0 è rappresentata dai gettoni all'interno dei posti e la marcatura d'arrivo M_1 al loro esterno) composta da tredici posti e otto transizioni così descritti:

-p_1 telai disponibili
-p_2 kit componenti disponibili
-p_3 macchina assemblatrice ferma
-p_4 macchina assemblatrice in funzione
-p_5 bici assemblate pronte per la verniciatura
-p_6 bici in trasferimento
-p_7 bici pronte per la verniciatura
-p_8 macchina per verniciatura libera
-p_9 macchina per verniciatura occupata
-p_{10} bici completate pronte per andare in magazzino
-p_{11} bici in trasferimento
-p_{12} bici in magazzino prodotti finiti
-p_{13} macchina per il trasferimento libera
-t_1 inizio assemblaggio
-t_2 fine assemblaggio
-t_3 inizio trasferimento
-t_4 fine trasferimento
-t_5 inizio verniciatura
-t_6 fine verniciatura
-t_7 inizio trasferimento in magazzino
-t_8 fine trasferimento in magazzino

M_0 = (4, 4, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1)

M_1 = (1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0)



(figura 16, struttura della Rete)

$$\begin{pmatrix}1\\1\\0\\1\\0\\1\\0\\0\\1\\0\\0\\0\\0\end{pmatrix} = \begin{pmatrix}4\\4\\1\\0\\0\\0\\0\\1\\0\\0\\0\\0\\1\end{pmatrix} + \begin{pmatrix}-1&0&0&0&0&0&0&0\\-1&0&0&0&0&0&0&0\\-1&1&0&0&0&0&0&0\\1&-1&0&0&0&0&0&0\\0&1&-1&0&0&0&0&0\\0&0&1&-1&0&0&0&0\\0&0&0&1&-1&0&0&0\\0&0&0&0&-1&1&0&0\\0&0&0&0&1&-1&0&0\\0&0&0&0&0&1&-1&0\\0&0&0&0&0&0&1&-1\\0&0&0&0&0&0&0&1\\0&0&-1&1&0&0&-1&1\end{pmatrix} \text{x} \begin{pmatrix}a\\b\\c\\d\\e\\f\\g\\h\end{pmatrix}$$

da cui otteniamo il seguente sistema di13 equazioni in 8 incognite:

$$\begin{cases} 1 = 4 - a \\ 1 = 4 - a \\ 0 = 1 - a + b \\ 1 = 0 + a - b \\ 0 = 0 + b - c \\ 1 = 0 + c - d \\ 0 = 0 + d - e \\ 0 = 1 - e + f \\ 1 = 0 + e - f \\ 0 = 0 + f - g \\ 0 = 0 + g - h \\ 0 = 0 + h \\ 0 = 1 - c + d - g + h \end{cases} \quad \text{risolvendo} \quad \begin{cases} a = \mathbf{3} \\ a = 3 \\ b = a - 1 = \mathbf{2} \\ b = a - 1 = 2 \\ c = b = \mathbf{2} \\ d = c - 1 = \mathbf{1} \\ e = d = \mathbf{1} \\ f = e - 1 = \mathbf{0} \\ f = e - 1 = 0 \\ g = f = \mathbf{0} \\ h = g = \mathbf{0} \\ h = 0 \\ 0 = 1 - 2 + 1 - 0 + 0 = 0 \end{cases}$$

Possiamo quindi concludere che applicando la sequenza di scatti σ (3, 2, 2, 1, 1, 0, 0, 0) è possibile raggiungere la marcatura M_1 partendo dalla marcatura M_0. Il vettore degli scatti è quindi ammissibile.

1.5 Analisi delle proprietà delle Reti di Petri, approccio comportamentale

Lo scopo dell'uso delle reti di Petri consiste nel riuscire a dedurre importanti conclusioni sul sistema oggetto di analisi, quindi dopo aver modellato il sistema lo step successivo prevede di analizzare il modello per capirne le proprietà e per riuscire a rispondere a domande del tipo come dovrebbe comportarsi il sistema in determinate situazioni operative, quali proprietà inerenti la struttura della sua rete, cosa aspettarsi e cosa no dal suo funzionamento. Detto questo possiamo analizzare alcune proprietà delle Reti di Petri, le quali dipendono dallo Stato iniziale del sistema, ovvero la Marcatura iniziale, e vengono dette proprietà comportamentali. Ci riferiamo alle proprietà di: raggiungibilità, limitatezza, vivacità [6], reversibilità e stato principale, copertura e persistenza[7].

1.5.1 Proprietà di raggiungibilità

La raggiungibilità è essenziale per analizzare gli aspetti dinamici dei sistemi. In una Rete di Petri lo scatto di una transizione cambia inevitabilmente la distribuzione dei gettoni sulla base della regola di scatto. È evidente che una sequenza di scatti genererà diverse distribuzioni dei gettoni all'interno della rete e quindi diverse marcature si susseguiranno. Ci si chiede quindi se esiste un'attivazione sequenziale che permette di passare dalla marcatura di partenza ad una nuova marcatura. Sulla base di questo una marcatura M si definisce raggiungibile a partire dalla marcatura M_0 se esiste una sequenza di transizioni che facendole scattare permette di passare da M_0 a M. Una sequenza di scatti è indicata con $\sigma = (t_1, t_2, \ldots, t_n)$, detto anche vettore degli scatti. I valori degli elementi di questo vettore corrispondono al numero di scatti di una determinata transizione. Quindi se M è

[6] A Study of Petri Nets. Modelling, Analysis and Simulation https://abhishekhalder.bitbucket.io/PetriNetReport.pdf
[7] Petri Nets: Properties, Analysis and Application. Tadao Murata https://ieeexplore.ieee.org/document/24143

raggiungibile da M_0 attraverso σ scriviamo M_0 [σ > M. Viene inoltre chiamato insieme di raggiungibilità l'insieme di tutte le possibili marcature raggiungibili partendo da M_0. Può anche capitare che M possa essere raggiunto da M_0 attraverso lo scattare di una sola transizione, in questo caso si dice che M è immediatamente raggiungibile da M_0.
(vedere l'esempio di cui sopra)

1.5.2 Proprietà di limitatezza

Viene definita limitata una rete di Petri se il numero di gettoni che ogni posto può contenere non supera un certo numero finito k per qualunque marcatura che può essere raggiunta a partire da M_0, ($M(p) \leq k$). Inoltre, una rete in cui la capacità massima di ogni Posto è limitata a 1 viene detta Rete Sicura. La sicurezza è quindi un caso speciale di limitatezza. Si tratta di una proprietà molto importante se pensiamo al significato fisico che può avere perché spesso i Posti in una Rete di Petri servono per rappresentare i buffer, ad esempio di un processo produttivo. Essendo la Rete sicura è possibile assicurare che non ci siano over-flow nei buffer e questo indipendentemente dalla sequenza di scatti che si verifica.

1.5.3 Proprietà di vivacità

La vivacità di una rete dipende dall'assenza di punti morti. È viva una rete che, in modo indipendente dalla marcatura che è stata raggiunta a partire dalla marcatura iniziale, permette di far scattare qualunque transizione attraverso un'ulteriore sequenza di scatto.
In sostanza una rete viva non ammette punti morti.

La vivacità è una proprietà molto importante per alcuni sistemi. Però per sistemi molto grandi non è pratica ed è piuttosto costoso verificare ciò. È tuttavia possibile definire diversi livelli di vivacità. Una transazione t in una rete di Petri è definita:

- L0-viva (o morta) quando la transizione non scatta in nessuna marcatura raggiungibile;
- L1-viva, potenzialmente scattante, se può scattare almeno una volta in una sequenza di scatti;
- L2-viva, se dato un qualunque numero intero positivo k, può scattare almeno k volte in una sequenza di scatti;
- L3-viva se e solo se esiste una sequenza di scatto dove la transizione scatta all'infinito;
- L4 viva se è L1-viva per ogni possibile marcatura.

L'ultimo livello di vivacità corrisponde all'assenza di punti morti. È inoltre facile capire che per ogni transizione, ogni livello di vivacità implica quello precedente ad eccezione di L0-viva.

È inoltre possibile andare a definire, grazie a quanto detto sopra, dei gradi di vivacità per la Rete nella sua totalità. Infatti, una Rete di Petri viene definita L-k viva (dove k indica il grado di vivacità) se ogni sua transizione è L-k viva, con k=0, 1, 2, 3, 4. Sulla base di ciò possiamo definire, ad esempio, una Rete di Petri L4-viva solo se ogni sua transizione risulti essere L4-viva.

1.5.4 Proprietà di reversibilità e stato principale

Una Rete si definisce reversibile se, per ogni marcatura M appartenente all'insieme delle possibili marcature raggiungibili nella rete, la marcatura iniziale M_0 è raggiungibile partendo dalla marcatura M. In altre parole, una rete è reversibile se è sempre possibile ritornare allo stato iniziale. In alcuni casi tuttavia non è necessario tornare allo stato iniziale fintanto che è possibile tornare ad un qualche stato principale. Si può quindi rendere meno

stringente la proprietà di reversibilità definendo uno stato principale. Quindi una marcatura M^l viene indicata come stato principale se, per ogni marcatura M dell'insieme delle possibili marcature raggiungibili, M^l è raggiungibile da M.

1.5.5 Proprietà di copertura

La marcatura M di una Rete di Petri è detta copribile se c'è una marcatura M^l tale per cui il numero di gettoni in ogni posto è maggiore o uguale rispetto alla marcatura di partenza M, $[M^l(p) \geq M(p)]$.

Inoltre, questa proprietà è legata al livello di vitalità L1, infatti data una marcatura che permette di abilitare una transizione, quest'ultima è morta se e solo se la marcatura in questione non è copribile. Viceversa, è L1-viva se e solo se è copribile.

1.5.6 Proprietà di persistenza

Quando lo scatto di una transizione non disabilita un'altra transizione si dice che in questo caso la rete è persistente. Quindi possiamo dire che se la rete è persistente una transizione abilitata può essere disabilitata solo dal suo stesso scatto Quindi in questo tipo di rete una transazione abilitata resterà tale fintantoché non scatterà. È possiamo desumere anche che, dato che lo scatto di una transizione non disabilita l'altra, le due transizioni in questione non sono in conflitto tra di loro. Infatti, in una situazione di conflitto una transizione può essere disabilitata dallo scatto di un'altra transizione poiché hanno una precondizione in comune. Quindi le reti persistenti sono sempre reti con assenza di conflitti.

1.6 Particolari classi di Reti di Petri

Oltre alle Reti di Petri Ordinarie possiamo trovare alcuni particolari tipologie di Rete[8]. Una Rete di Petri in cui tutte le transizioni hanno solo una pre-condizione e solo una post-condizione prende il nome di Macchina a Stati Finiti. Attraverso una definizione formale possiamo definirla così: è una Rete in cui $\forall\ t_j \in T$, $|\bullet t_j| = 1$ e $|t_j \bullet| = 1$.

Tale tipologia di reti ha la caratteristica di essere conservativa, nel senso che il numero di gettoni all'interno della rete resta sempre invariato. Inoltre, è viva se e solo se è possibile andare da un nodo all'altro della rete semplicemente seguendo la relazione di flusso (in altre parole se è fortemente connessa) e se è presente almeno un gettone.

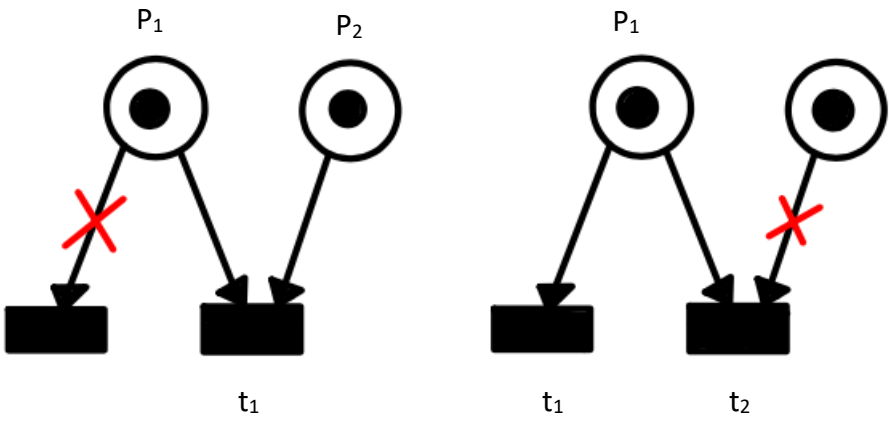
Possiamo anche notare come tale rete possa rappresentare conflitti ma non possa invece modellare la presenza di sincronizzazione e l'inizio concorrenza.

Un'ulteriore sottoclasse delle Reti di Petri è la cosiddetta classe dei Grafi Marcati. Questa tipologia è duale rispetto alla precedente categoria, infatti in questo caso ogni Posto ha una sola transizione sia in ingresso che in uscita. Inoltre, proprio perché duale rispetto alla precedente questa classe è in grado di rappresentare concorrenza e sincronizzazione ma non di modellare i conflitti.

Atro tipo è la Rete a Scelta Libera che viene modellizzata perché andando a mescolare situazioni di conflitto e di concorrenza si crea una situazione di confusione, nella quale non è chiara la possibile evoluzione della rete. Tali reti quindi escludono la possibilità che la confusione si verifichi. Viene infatti definita così: è una rete che per ogni arco che collega un posto e una transizione o quel posto è l'unico in ingresso per quella transizione, in questo modo non c'è concorrenza, o quella transizione è l'unica in uscita da quel posto, quindi non si verificano situazioni di conflitto (figura 17). L'importanza di ciò sta nel come

[8]Automazione industriale: controllo logico con reti di Petri
http://www.schoolofnerd.it/sites/default/files/ControlloLogicoRetiPetri_Parte2.pdf

sono ammessi conflitti controllati. Infatti, i conflitti si verificano quando un posto è pre-condizione per più transizioni, ma in questo caso sarà anche l'unico ingresso per quelle transizioni, che saranno quindi in conflitto simultaneo. Si vuole quindi evitare che situazioni di conflitto possano non essere contemporaneamente abilitate, perché altrimenti la risoluzione dei conflitti non sarebbe libera dipendendo anche da altri posti. Questo permette quindi di scegliere in modo libero quale transizione attivare dato che la presenza di gettoni in altri posti non influenza la scelta della transizione da attivare.

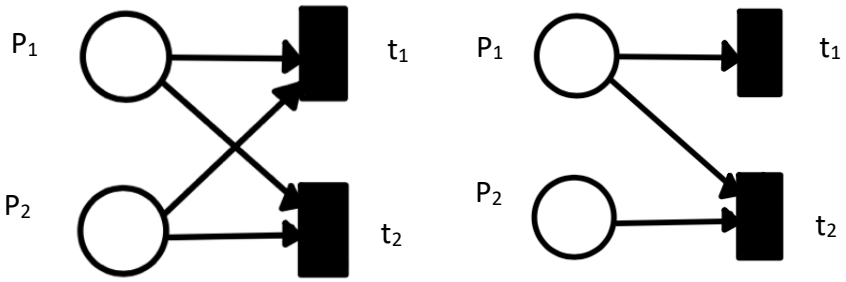


(figura 17, Rete a scelta libera)

Da questo è possibile dedurre che le macchine a stati liberi e i grafi marcati sono reti a scelta libera, ma non è altrettanto vero il contrario.

Altre due tipologie sono le Reti a Scelta libera Estesa e le Reti a Scelta Libera Asimmetrica. Nella prima (figura 18a) si fa riferimento a reti in cui se due posti p hanno una transizione di uscita in comune allora questi avranno le stesse transizioni di uscita.

La seconda invece (figura 18b) definisce un tipo di reti in cui se due posti p hanno alcune transizioni di uscita in comune allora l'insieme delle transizioni in uscita di uno dei due contiene quello dell'altro.



(figura 18a, Rete a scelta libera Estesa) (figura 18b, Rete a scelta libera Asimmetrica)

CAPITOLO II

LE RETI DI PETRI DI ALTO LIVELLO

2.1 Le reti per la modellazione dei sistemi complessi

Come abbiamo potuto capire le Reti di Petri sono uno strumento molto semplice ed intuitivo con cui è possibile analizzare e modellare dei sistemi, tuttavia è importante sottolineare che questo formalismo non è adatto per modellare dei sistemi più complessi. Quindi oltre alle reti classiche in letteratura sono state proposte varie estensioni, alcune delle quali non forniscono supporto attraverso strumenti e nessuna applicazione pratica, rimanendo quindi una semplice proposta su carta, altre invece sono usate con frequenza e supportate in modo piuttosto ampio. Numerose sono quindi le varie estensioni che vengono accettate ma principalmente tre sono quelle maggiormente riconosciute ed usate[9]:

- estensione del colore;
- estensione del tempo;
- estensione gerarchica.

Nella prima tipologia di estensione per ottenere una maggior realisticità del modello vengono aggiunti dei colori (cioè delle informazioni aggiuntive) ai gettoni in modo che questi possano essere tra loro distinguibili e avere delle determinate caratteristiche. Senza ciò le Reti sarebbero un semplice linguaggio privo di parametri o variabili.

Nella seconda come si capisce dal nome viene introdotta la variabile tempo. Infatti le reti di Petri trovano frequente applicazione nella modellazione di processi in cui il tempo è una componente fondamentale. Quindi il funzionamento di Posti e Transizioni è vincolato allo

[9] Transactions on Petri Nets and Other Models of Concurrency VII
https://link.springer.com/content/pdf/10.1007/978-3-642-38143-0.pdf

scorrere del tempo. Per esempio, parlando di workflow è importante che siano noti i tempi di utilizzo, i livelli di servizio e molto altro ancora.
L'ultimo tipo infine, proprio perché a volte i sistemi sono molto complessi, con molte relazioni ed entità, introduce la gerarchia consentendo di mettere in evidenza tutti gli "strati" che può appunto avere un sistema complesso.

Le Reti di Petri di Alto Livello incorporano i dati, il tempo e la gerarchia per modellare sistemi e processi complessi.

2.2 L'estensione del colore

Le Reti di Petri Colorate consistono in un linguaggio per la modellazione di sistemi discreti ed hanno la caratteristica di unire al proprio interno le peculiarità delle Reti tradizionali con le capacità dei linguaggi di programmazione di alto livello.
Rispetto alla Reti di cui abbiamo trattato nel primo capitolo, in cui gettoni erano semplicemente rappresentati da punti all'interno dei posti e quindi tutti uguali, in questa estensione riusciamo a distinguerli in quanto possono avere dei "colori" diversi, che vanno a definire degli attributi precisi per tutti i gettoni appartenenti una certa colorazione. Questo permette di creare modelli molto più precisi, soprattutto in tutti quei casi in cui diverse entità di un sistema hanno un comportamento molto simile.

Iniziamo quindi l'analisi di questa estensione con una definizione formale[10]: una Reti di Petri Colorata è una rete Z = (P, T, F, $\sum$, C, E, G, I) nella quale:

- P è l'insieme dei Posti;
- T è l'insieme delle Transizioni;

[10] Coloured Petri Nets, Modelling and Validation of Concurrent Systems
http://artamonoviv.ru/wp-content/uploads/2016/08/Coloured_Petri_Nets_modeling_and_validation_of_concurrent_systems__2009.pdf

- $F \subseteq (P \times T) \cup (T \times P)$ indica l'insieme degli archi in entrata e in uscita;

- $\sum$ definisce un insieme di colori all'interno del modello. Contiene quindi tutti i possibili colori, operazioni e funzioni usati all'interno della rete;

- C rappresenta la funzione colore la quale assegna ad ogni posto p un colore di $\sum$

- E definisce la funzione di espressione dell'arco che va a mappare ogni arco nell'espressione E;

- G è la funzione di guardia che assegna ad ogni transazione t una guardia, il cui compito è quello di fornire un controllo sull'attivazione della transizione. L'output di tale funzione dovrebbe restituire un valore di tipo vero/falso;

- I, funzione di inizializzazione, che associa ad ogni posto p un'espressione I che restituisce un multiset di gettoni che devono avere valore corrispondente al colore del posto, C(p).

Quindi come una classica Rete si compone di Posti, Transizioni ed Archi[11]. Ma in questo caso i Posti hanno un "tipo" (o "colore" o "ordinamento"), questo formalmente è definito dalla funzione colore, $C: P \rightarrow \sum$ con $C= \{c_1, c_2, \ldots, c_n\}$, che va a stabilire quale tipo di gettoni essi possono contenere (ad esempio INT, che sta ad indicare che quel posto accoglie solo gettoni con numeri interi). Ogni posto poi è munito di un contrassegno che consiste in un multiset di valori, che rappresenta un elemento fondamentale per questo tipo di estensione, in cui uno stesso elemento può apparire più volte. Uno degli elementi del contrassegno è il gettone.

Da un punto di vista più formale un multiset m in un insieme S, non vuoto, viene considerato come una funzione $m: S \rightarrow N$. Tale funzione assegna ad ogni elemento s quante volte esso appare all'interno del multiset m, [m(s)], numero che prende il nome di coefficiente di s in m. Inoltre l'elemento s può essere considerato parte del multiset m solo nel caso in cui il

suo numero di apparizioni è maggiore di zero, m(s) > 0. La dimensione di un multiset si ottiene sommando i vari coefficienti di apparizione dei sui elementi.

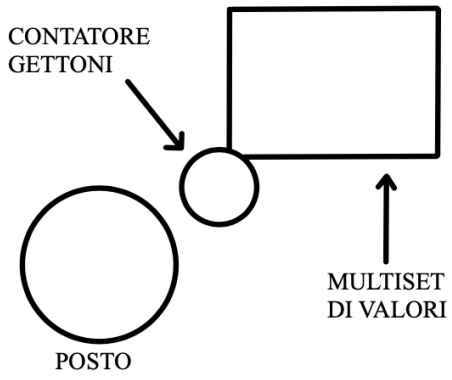
Inoltre, un multiset può essere costituito anche da un insieme infinito nel caso in cui l'insieme S sia infinito, in ogni caso però il numero di apparizioni di un elemento appartenente al multiset rimane finito.

Un multiset in cui il numero di apparizioni di ogni elemento è pari a zero prende il nome di multiset vuoto, espresso con 0/MS.

Chiariamo il tutto con un esempio: il multiset m_a= 3' (giallo) ++ 2' (verde) ++ 5' (rosso) ++ 4' (blu) può essere rappresentato dalla seguente funzione, la quale riporta i coefficienti di apparizione per ogni elemento

$$m_a(s) = \begin{cases} 3 \text{ se} = (\text{giallo}) \\ 2 \text{ se} = (\text{verde}) \\ 5 \text{ se} = (\text{rosso}) \\ 4 \text{ se} = (\text{blu}) \\ 0 \text{ altrimenti} \end{cases}$$

Questo ha ovviamente una rappresentazione grafica (figura 19): il contrassegno consiste in un cerchio e un rettangolo disegnati accanto ai posti.



(figura 19, Rappresentazione grafica del contrassegno)

Il primo accoglie un numero equivalente a quanti sono i gettoni presenti nel posto e il secondo contiene del testo che corrisponde al multiset di valori, nel quale usiamo un

apostrofo per separare il valore del gettone dal conteggio di quanti gettoni con quel valore sono compresi nel contrassegno (ad esempio “2 ’ giallo” sta a significare che ci sono due gettoni il cui valore è giallo). Nel caso in cui i gettoni appartenenti al contrassegno abbiamo valori diversi vengono separati da due segni di addizione, ++, (ad esempio 2 ’ giallo ++ 1 ’ verde).

Anche gli archi in questa estensione, rispetto alla versione classica, presentano delle caratteristiche aggiuntive. Ognuno di essi ha un’iscrizione, cioè un’espressione che può contenere una o più variabili di un certo tipo (la funzione E : N $\rightarrow$ EXPR assegna ad ogni arco n un espressione E(n)), queste ultime però devono essere dello stesso tipo del Posto a cui l’arco è connesso, in altre parole Tipo [E(p, t)] = C (p) se è un arco di ingresso, Tipo [E(t, p)] = C (p) se è un arco di uscita.

Le transizioni dal momento che possono avere più archi in entrata e in uscita hanno più variabili, indichiamo con Var(t) le variabili di una transizione. Ovviamente si tratta di quelle che appaiono sulle guardie che controllano la transizione e delle espressioni degli archi in ingresso e in uscita che le appartengono.

Inoltre, ad ogni variabile della transizione può essere assegnato un valore all’interno dell’insieme rappresentato dal suo tipo. Questo viene definito “legame”, formalmente: un legame b di una transazione t è costituito da una funzione che assegna ad ogni variabile v della transizione t un valore b(v) appartenente al tipo della variabile v, cioè b(v) $\in$ Tipo(v).

Inoltre, prende il nome di Elemento di Associazione la coppia (t, b), composta dalla transazione t e dal suo legame b. Tale elemento viene così indicato: “nome della transizione<variabile con valore associato>”. È inoltre importante sottolineare come questa associazione esiste al di là di un particolare contrassegno, infatti vengono presi in considerazione solo transizioni e variabili degli archi in ingresso o in uscita da esse.

Dopo aver fornito queste informazioni essenziali possiamo ora andare a definire la regola di abilitazione e di scatto per una Reti di Petri estesa con l’uso di dati. Data una certa

marcatura l'elemento di collegamento è definito abilitato se sono vere entrambe due condizioni: la guardia che controlla la transizione è soddisfatta (nel caso in cui questa sia presente) e tutte le pre-condizioni contengono per lo meno i gettoni il cui colore è esplicitato dall'espressione dell'arco che unisce i posti alla transizione nel rispettivo legame b, viene quindi specificato il multiset di gettoni che la transizione rimuove dai posti quando essa si verifica con il suo legame. Di conseguenza una transazione è definita abilitata se per una data marcatura i suoi elementi di associazione sono abilitati. Quindi nel caso in cui l'elemento di associazione, o analogamente la transizione, è abilitato questo può verificarsi e quindi scattare. Come conseguenza lo scatto in questo caso non fa si che semplicemente vengano eliminati i gettoni dalle pre-condizioni e che compaiano nella post-condizioni, ma proprio perché i gettoni appartengono ad un tipo lo scatto sottrae dal posto in input gettoni corrispondenti alle variabili sugli archi in ingresso e produce gettoni sui posti in output corrispondenti alle variabili esplicitate sugli archi in uscita dalla transizione.

2.2.1 Esempio di una Rete di Petri Colorata

Di seguito viene fornito un esempio di una Rete di Petri Colorata[12] (figura 20) per un processo di stampa in cui grazie a questo tipo di estensione è possibile distinguere tra i diversi tipi di stampante e assegnare ad ogni lavoro un identificatore univoco.

La rete è composta da quattro posti e tre transizioni:

-p_1 Stampanti pronte;
-p_2 Prossimo ID;
-p_3 Lavori;
-p_4 Occupata;

-t_1 Prossimo lavoro;
-t_2 Inizio lavoro;
-t_3 Fatto/fine lavoro;

Si vuole distinguere tra tre tipi di stampante: Canon, Epson e HP. In aggiunta ogni lavoro verrà distinto dagli altri grazie ad un ID. Sulla base di questo possiamo capire che una "richiesta di lavoro" consiste nell'abbinamento di un ID lavoro con un tipo di stampante e di conseguenza un "lavoro attivo" è una coppia che consiste in un tipo di stampante e una richiesta di lavoro.

Andiamo quindi a definire le dichiarazioni dei set di colori per questa rete:

- color set STAMPANTE = CANON, EPSON, HP;

- color set ID = INT, (INT= numero intero);

- color set LAVORO = ID x STAMPANTE;

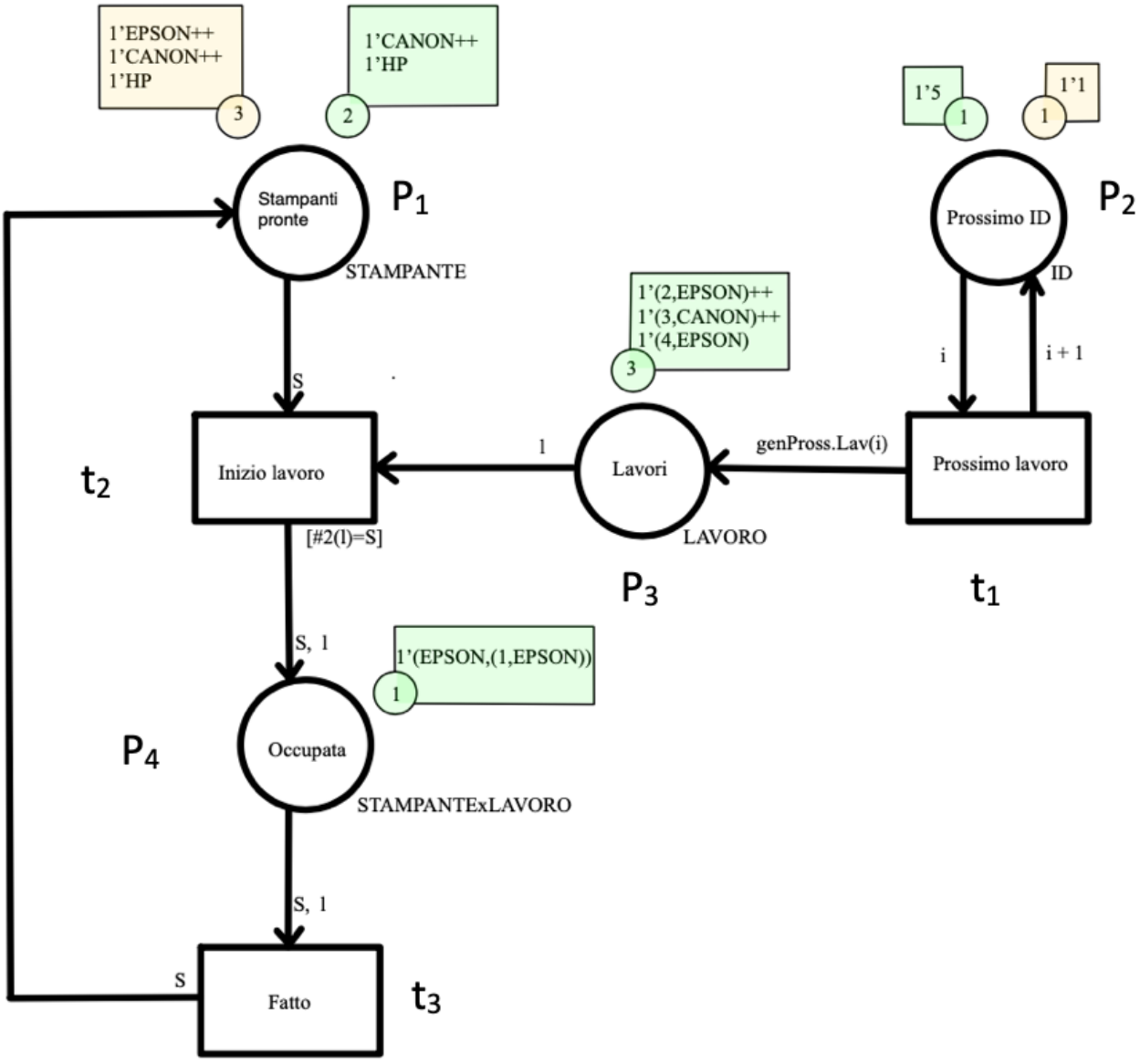
- color set STAMPANTE x LAV = STAMPANTE x LAV.

Grazie a questo possiamo esplicitare una richiesta di lavoro nel seguente modo (3, "HP") che ci permette di capire che il lavoro di stampa con ID=3 è abbinato alla stampante HP.

[12] From Petri NETS to Colored Petri NETS: A Tutorial Introduction to NETS Based Formalism For Modeling And Simulation
https://www.researchgate.net/publication/339409780_From_Petri_NETS_to_Colored_Petri_NETS_A_Tutorial_Introduction_to_NETS_Based_Formalism_For_Modeling_And_Simulation

Come già detto nella descrizione di questa estensione visto che i gettoni sono distinguibili ogni Posto deve indicare qual è il set di colori che ammette, e questo viene indicato scrivendo il set all'esterno del Posto in basso a destra.

Bisogna poi andare a definire le espressioni degli archi che vengono esplicitate attraverso una dichiarazione Var, ovvero: Var s = STAMPANTE; Var l = LAVORO; Var i = ID.



(figura 20, Rete di Petri Colorata per il processo di stampa)

Le Reti di Petri Colorate per le transizioni possono prevedere delle guardie facoltative che offrono un controllo aggiuntivo sul verificarsi di una transizione. Nella nostra rete la transizione "inizio lavoro" presenta la guardia g: [#2(l) = s].

Discutiamo quindi il significato di ciò: come possiamo vedere tale transizione ha due posti di ingresso, STAMPANTI E LAVORI, l'iscrizione dell'arco relativa alla prima pre-condizione è la variabile s=STAMPANTE a cui può essere associato CANON, EPSON o HP; l'iscrizione relativa all'arco dell'altra pre-condizione è la variabile l= LAVORO la quale è composta da ID e STAMPANTE. Chiarito questo possiamo spiegare la guardia la quale semplicemente va a stabilire che il secondo membro della coppia (nell'iscrizione d'arco l= Lavoro= IDxSTAMPANTE), ovvero la stampante, deve essere lo stesso della stampante associata alla variabile s.

Per terminare l'analisi del funzionamento di questa rete diamo un occhio alla transizione "prossimo lavoro", la quale ha come posto di ingresso "prossimo ID", di tipo INT; che genera gli ID per i lavori e quando la transizione scatta rimuove il numero intero ID (espressione arco i). Ciò che si verifica invece nelle-post condizioni è che in "prossimo lavoro" lo scatto della t rimette in sequenza il successivo numero intero grazie all'espressone d'arco "i+1", invece in "lavori" inserisce gettoni sulla base del risultato della funzione dell'espressione d'arco "genProssimoLavoro(i)" che genera una richiesta casuale di stampante per un certo ID lavoro.

Guardando la figura 20 vediamo come partendo dalla marcatura iniziale (in verde) una nuova marcatura (in giallo) viene raggiunta dopo alcuni passaggi. Al fine di rendere tutto il più chiaro possibile per prima cosa spieghiamo il significato delle marcature. Nella marcatura iniziale P_1 e P_2 sono marcati. Il Posto P_1 come possiamo notare dal contrassegno contiene tre gettoni (il cui color set è STAMPANTE) che vengono specificati nel multiset di valori, cioè 1 gettone = Epson (come si nota nel multiset 1'Epson), 1 gettone = Canon (1'Canon), 1 gettone = HP (1'HP). Il posto P_2 contiene un gettone (il cui color set è ID) specificato nel multiset come 1'1, cioè 1 gettone il cui valore corrispondente è 1 (essendo il primo ID generato). Nella marcatura finale invece P_1, P_2, P_3 e P_4 risultano marcati. P_1 contiene ora due gettoni il cui valore specificato nel multiset è 1 gettone = Canon e 1 gettone

= HP (1'Canon ++ 1'HP). P_2 contiene un gettone con valore 5, il che vuol dire che sono già stai generati gli ID =1, 2, 3 e 4. Il posto P_3 contiene tre gettoni come specificato dal contrassegno (con color set LAVORO= ID xSTAMPANTE, si abbina quindi un ID ad una stampante, come specificato dall'iscrizione d'arco in entrata) con i seguenti valori: un gettone con valore "2, Epson", che indica che per il lavoro con ID=2 è stata richiesta la stampante Epson; un gettone con valore "3, Canon"; un gettone con valore "4, Epson". Nel posto P_4 è presente un gettone il cui valore è "Epson, (1, Epson)", possiamo specificarne il significato notando che il color set per questo posto è STAMPANTExLAVORO, di conseguenza la stampante risulta essere la Epson, che viene associata al lavoro (color set LAVORO = IDxSTAMPANTE) con ID=1 per cui era stata appunto richiesta la stampante Epson.

Possiamo ora analizzare la situazione:

- nel posto Occupata troviamo la stampante Epson la quale è occupata con il lavoro con ID=1 (associato appunto alla stampante Epson), come possiamo notare dal contrassegno "1' (Epson, (1, Epson))". Solo dopo aver finito la stampante sarà libera e tornerà nel posto Stampanti pronte e quindi il loro con ID=2 potrà essere portato avanti;

-notiamo anche che due sono le stampanti libere, HP e Canon, in Stampanti libere e che nel posto Lavori ci sono tre lavori che attendono la loro stampante e infine vediamo che il l'ID=5 è il prossimo;

- veniamo alla parte più importante soffermando la nostra attenzione sulla transizione Inizio Lavoro. La variabile s ha due possibili legami, ovvero la due stampanti pronte HP e Canon; la variabili l invece ne ha 3, che corrispondono ai 3 elementi del contrassegno in Lavori. Essendo quindi le pre-condizioni marcate la transazione potrebbe essere abilitata e quindi scattare essendoci sei associazioni disponibili. Tuttavia, non è cosi perché è necessario soddisfare anche la guardia, quindi dal momento che la Stampante Epson non è disponibile l'unico elemento di associazione che permette di abilitare la transizione è (inizio lavoro; 3,

Canon) dato che l'elemento di associazione (inizio lavoro; Canon) è disponibile. Questi due consentono quindi lo scatto della transizione.

2.3 L'estensione del tempo

Come possiamo ben intuire il tempo è un aspetto di fondamentale importanza in molti sistemi reali. Infatti, tutte le varie parti di cui questi sono composti (ad esempio i vari step di cui si compone un flusso di lavoro) richiedono tempo per essere eseguiti, non sono delle azioni istantanee. L'introduzione del tempo quindi permette di aumentare l'espressività di queste reti e di modellare gli aspetti temporizzati dei sistemi in modo tale da poter trarre delle conclusioni circa le performance degli stessi.

Inizialmente gli studi sulle Reti di Petri basati su questa estensione hanno associato alle transizioni l'attributo del tempo (di cui ci occuperemo in seguito), successivamente sono emerse nuove estensioni in cui il tempo sotto forma di durata o di intervalli è stato associato ai Posti, agli Archi e ai Gettoni.

Rispetto ad una Rete classica in cui il suo stato è espresso esclusivamente dalla marcatura, in una Rete in cui viene introdotto il concetto del tempo la situazione è differente. Infatti, in questo caso lo stato attuale di una rete è espresso dalla marcatura esistente in un particolare momento temporale. Due sono i parametri da tenere in considerazione: la marcatura ed il tempo. Si tratta quindi rispettivamente di un parametro discreto e di uno continuo, che rendono quindi le Reti di Petri dipendenti dal tempo una sorta di strumento ibrido attraverso il quale è possibile descrivere dei sistemi.

Gli studi iniziali sull'introduzione del tempo in queste reti sono associati a Merlin (1976) e Ramchandani (1980) che hanno elaborato rispettivamente due modelli di Reti di Petri dipendenti dal tempo, in cui quest'ultimo viene associato alle transizioni. Si tratta delle Reti di Petri Temporali (Time Petri Nets) e delle Reti di Petri Temporizzate (Timed Petri Nets).

2.3.1 Le Reti di Petri Temporali

In una Rete di Petri Temporale un intervallo di tempo, $[a_t, b_t]$ viene associato ad ogni transizione. Quindi ogni transizione rappresenta un evento istantaneo, il suo scatto quindi non richiede tempo, che può verificarsi durante un determinato intervallo di tempo, all'interno del quale la transizione risulta quindi abilitata. Gli estremi di questo intervallo rappresentano rispettivamente:

- a_t è il primo tempo di abilitazione della transizione e indica il tempo minimo che deve trascorrere dall'istante in cui la transizione è divenuta abilitata prima che questa possa scattare ($a_t > 0$);
- b_t è l'ultimo tempo di abilitazione per la transizione e indica il massimo ritardo ammissibile, rispetto all'istante in cui diviene abilitata, entro il quale essa dovrebbe scattare, ($b_t < \infty$).

Sulla base di quanto appena detto possiamo fornire una definizione formale[13]: una Rete di Petri Temporale è una sestupla S (Z)=(P, T, F, W, M_0, I) tale che la quintupla (P, T, F, W, M_0) rappresenta una classica Reti di Petri così come descritta nel capitolo I, che qui prende il nome di "scheletro", mentre I è la funzione intervallo che associa ad ogni transizione un intervallo di abilitazione all'interno del quale può scattare, $I : T \rightarrow Q^+_0$ e per ogni $t \in T$, con $I(t) = (a_t, b_t)$ e $(a_t) \leq (b_t)$.

Con l'introduzione della variabile tempo cambia il concetto di marcatura la quale non è più sufficiente al fine di descrive in maniera integrale la situazione all'interno della rete[14]. Infatti, oltre agli ormai classici gettoni bisogna prendere in considerazione per ogni transizione del tempo trascorso dalla sua abilitazione. Infatti, in questa estensione viene introdotta un'ulteriore marcatura associata alle transizioni che risultano abilitate e il simbolo

[13] A causal semantics for time Petri nets https://www.sciencedirect.com/science/article/pii/S0304397599001140
[14] Time and Petri Nets https://link.springer.com/content/pdf/10.1007/978-3-642-41115-1.pdf

“#” per quelle che invece risultano non abilitate, si tratta di un vettore con tante componenti quante sono le transizioni all’interno della rete. Quindi in questo caso avremmo a che fare con due tipi di marcature che vanno ad esplicitare la situazione dei Posti (la marcatura classica) e delle Transizioni. Queste prendono rispettivamente il nome di marcatura-p e marcatura-t ed insieme consentono di definire in modo completo la situazione all’interno della rete. Una tale coppia prende il nome di “Stato”. Possiamo quindi andare a dare una definizione formale di questi tre concetti:

- sia P l’insieme dei Posti in una Rete Temporale Z, una marcatura-p all’interno di essa è una funzione m: $P \rightarrow \mathrm{N}$;

- sia T l’insieme delle Transizioni in una rete temporale Z, una marcatura-t al suo interno è una funzione h: $T \rightarrow R^{+}_{0} \cup \{\#\}$.

Quindi h(t) ci permette di sapere quanto tempo è trascorso dall’abilitazione della transizione. Si tratta in sostanza di un orologio interno della transizione;

-sia Z = (P, T, F, W, M_0, I) una Rete di Petri Temporale, m la marcatura-p ed h la marcatura-t. Si definisce Stato la coppia z = (m, h).

In questo caso se la transazione è abilitata h(t) è un numero razionale non negativo e minore dell’ultimo tempo di abilitazione di t, nel caso in cui fosse disabilitata h(t) = #.

Tuttavia, non ogni coppia (m, h) costituisce uno stato in quanto i due tipi di marcatura devono adattarsi l’uno all’altro e questo si verifica nel caso in cui il tempo passato dall’abilitazione della transizione è un numero, non #, e quando essa è abilitata dalla marcatura-p.

Dopo aver descritto questi concetti essenziali e distintivi rispetto ad una rete classica, possiamo andare a discutere quello che è il comportamento dinamico di questa estensione. Anche qui troviamo qualche differenza, infatti con lo scattare delle transizioni cambia la marcatura-p (sulla base della regola di scatto tradizionale) ma cambia anche la marcatura-t perché nuove transizioni potrebbero diventare abilitate ed altre disabilitate. Tuttavia, quest’ultima marcatura può cambiare anche solo con lo scorrere del tempo.

Quindi per descrivere l'evoluzione della rete è essenziale definire quando una transazione è abilitata, e per fare ciò bisogna per prima cosa conoscere quali delle transizioni abilitate nella marcatura-p hanno, con il passare del tempo, raggiunto il loro intervallo di abilitazione. Quindi una transizione t è abilitata nello stato z= (m, h) se valgono entrambe due condizioni:

- t è abilitata nella marcatura m;

- $a_t \leq h(t) \leq b_t$.

Arriviamo quindi a come la rete evolve introducendo due regole, la prima per lo scatto delle transizioni e la seconda per lo scorrere del tempo, in entrambi i casi abbiamo due eventi che vanno a modificare lo stato della rete.

La regola di scatto prevede che una transizione t* nello Stato della rete z = (m, h) può scattare se è abilitata e questo comporta la sottrazione dei gettoni dalle pre-condizioni e la loro comparsa nelle post-condizioni. Si raggiunge quindi un nuovo stato z' = (m', h'); come detto sopra però lo scatto va ad influire su entrambe le marcature, in cui:

-m' = m + Δt*

- $\forall$ t (t $\in$ T $\rightarrow$ h'(t) =:

    - # se con lo scatto la transazione non è più abilitata;
    - h(t) se la transizione t abilitata in m lo è ancora in m' e non condivide nessuna precondizione con la transazione t* (in sostanza il suo orologio non viene azzerato);
    - 0 altrimenti.

La seconda regola prevede che dato un numero reale non negativo τ, che indica il tempo trascorso, e uno Stato z = (m, h), il trascorrere del tempo a partire da z è possibile se:

$\forall$ t (t $\in$ T $\wedge$ h(t) $\neq$ #) $\rightarrow$ h(t) + $\tau \leq$ ultimo tempo di abilitazione di t. Dopo che τ tempo è trascorso lo stato della rete cambia in z', con:

-m' = m, quindi non è scattata nessuna transizione;

-$\forall$t (t $\in$ T $\rightarrow$ h'(t) =:

- h(t) + τ, se la transizione è ancora abilitata all'interno del suo intervallo

- #, se invece non è più abilitata nel suo intervallo.

2.3.2 Le Reti di Petri Temporizzate

L'estensione che prende il nome di Rete di Petri Temporizzata prevede che ad ogni transizione venga associata una certa durata, d_t. Questo perché nel mondo reale, come è ben immaginabile, ogni evento o processo ha bisogno di un certo quantitativo di tempo per essere portato a compimento. Normalmente non vengono ammesse durate pari a zero, però nei casi in cui un evento abbia durata molto inferiore rispetto agli altri può essere utile prendere in considerazione questa possibilità.

Visto che ad ogni transizione viene associato un certo tempo per la sua esecuzione è chiaro che in questa tipologia di rete il suo scatto richiederà del tempo, la sua abilitazione invece, rispetto alle Reti di Petri Temporali, si verifica non appena tutte le pre-condizioni contengono gettoni a sufficienza, senza dover aspettare nessun lasso di tempo.

Quindi una Rete di Petri Temporizzata è una sestupla[15] $Z = (P, T, F, W, M_0, D)$ nella quale $D : T \rightarrow Q^+_0$ è la funzione di durata che, abbiamo già detto, assegna una certa durata ad ogni transizione.

Come detto in precedenza per le Reti di Petri Temporali, anche in questo caso al fine di poter capire la situazione all'interno della rete è necessario considerare due tipi di marcature, che insieme vanno a formare uno Stato che va ad esplicitare la condizione complessiva della rete. Quindi oltre alla classica marcatura m che associa ad ogni posto un certo numero di gettoni (marcatura p) consideriamo un ulteriore marcatura-t che è una funzione $u : T \rightarrow Q^+_0$ che per ogni transizione abilitata associa ad essa un timer che va ad

[15] Time and Petri Nets https://link.springer.com/content/pdf/10.1007/978-3-642-41115-1.pdf

indicare quanto tempo manca affinché la transizione abbia finito di scattare ($u_t \leq d_t$). Nel caso in cui u(t)=0 significa che la transizione non è in fase scatto, ma non sta a significare che essa non sia abilitata dalla marcatura. Quando una transizione viene attivata u(t)=0 viene settato sulla durata d(t) ed inizia quindi a contare alla rovescia il tempo finché lo scatto non sarà completato.

Detto questo possiamo capire che lo Stato iniziale all'interno di una rete, nel quale nessuna transazione è nella fase di scatto, è definito da $Z_0 = (m_0, u_0)$ dove u(t) = 0 per qualunque transizione, visto che nessuna di esse sta compiendo il suo scatto.

Per andare ad esaminare il comportamento dinamico della rete e capire quindi come evolve è opportuno andare a definire la regola che prende il nome di "maximal step". Sulla base di questa regola viene attivato un insieme massimo di transazioni non appena abilitate. Formalmente, Se z = (m, u) è uno stato della Rete di Petri Temporizzata Z = (P, T, F,W M_0, D), $M \subseteq T$ è un maximal step se:

- $\forall t\ (t \in M \rightarrow u(t) = 0$;
- $\sum_{t \in M} t^- \leq m$;
- $\forall\ t^*\ ((\ t^* \in T \wedge t^* \notin M \wedge t^{*-} \leq\ m \wedge u(t^*) = 0) \rightarrow (\sum t^- + t^{*-})$ non è $\leq m)$.

In sostanza le prime due condizioni definiscono il maximal step come un set di transazioni, delle quali nessuna è in fase di attivazione/scatto e tutte risultano essere abilitate durante il maximal step (abilitazione collettiva). L'ultima invece non consente l'esistenza di un super insieme, con altre transizioni, che soddisfi le altre due condizioni.

È quindi possibile ora definire, come fatto nel paragrafo precedente, le regole che definiscono lo scatto della transizione e lo scorrere del tempo per questa tipologia di estensione.

Lo stato all'interno della rete può cambiare come conseguenza dello scattare delle transizioni. Infatti, sulla base della regola del "maximal step" un insieme di transizioni può scattare sequenzialmente. Quando le transizioni scattano vengono rimossi i gettoni dalle

pre-condizioni per poi ricomparire nelle post-condizioni. Inoltre, considerando anche la durata delle transizioni man mano che queste scattano il loro timer viene caricato sulla durata necessaria, ovviamente solo per quelle che appartengono al “maximal step”. Formalmente:

se $z_1 = (m_1, u_1)$ è uno stato della rete temporizzata Z e $M \subseteq T$ è un maximal step, M può scattare in z_1 se è un maximal step e come conseguenza dello scatto la rete raggiunge lo stato $z_2 = (m_2, u_2)$ in cui:

- $m_2 = m_1 - \sum_{t \in M} t^- + \sum_{t \in M,\, d(t)=0} t^+$

- $u_2(t) = D(t)$ se $t \in M$ oppure $u_1(t)$ altrimenti.

La rete, essendo questa un’estensione basata sul tempo, può evolvere anche grazie al trascorrere del tempo. Infatti, una Rete di Petri Temporizzata può raggiungere lo stato successivo in seguito allo scatto solo nel momento in cui le transizioni hanno completato il loro scatto, ovvero dopo d(t) istanti temporali. Di fatto quindi, quando una transizione abilitata scatta, i gettoni vengono rimossi immediatamente dalle pre-condizioni ma vengono aggiunti alle post-condizioni solo quando il loro scatto è completo. Durante il processo di scatto inoltre il timer viene decrementato di volta in volta di un’unità fino appunto a quando il processo di scatto è completato.

2.4 L'estensione gerarchica

Le Reti di Petri si avvalgono della gerarchia per riuscire a ridurre la complessità di un modello di grandi dimensioni andando a suddividerlo in una serie di sotto modelli o moduli, i quali hanno il vantaggio di poter essere riutilizzati in posizioni diverse dello stesso sistema, riuscendo così a focalizzarsi su una parte specifica di un dato sistema senza dover per forza tenere in considerazione il sistema nella sua totalità[16].

L'estensione delle Reti di Petri attraverso la gerarchia permette di mettere in evidenza i vari livelli di un sistema, ovvero la sua struttura, andando a scomporlo in una molteplicità di moduli cosi da poter strutturare anche sistemi di dimensioni maggiori e più complicati. Infatti, per sistemi semplici la struttura è di facile rappresentazione, avendo pochi Posti e Transizioni tutte le varie parti possono essere mostrate con chiarezza. Per sistemi complessi invece una rappresentazione attraverso una rete classica sarebbe troppo vasta dovendo comprendere troppi Posti e Transizioni.

Calandoci nella realtà questo permette di rendere espliciti i vari sottoprocessi di cui si compone un processo di livello superiore. I moduli quindi sono una specie di sottorete la cui funzione è quella di "esplodere il contenuto" di una transizione di quella che è la rete principale, transizione che in questa estensione prende il nome di Transizione di Sostituzione. Così facendo è possibile andare a mitigare uno dei limiti delle classiche Reti di Petri che è appunto quello di rappresentare un sistema nella sua totalità ma senza mostrare la sua struttura gerarchica sottostante, infatti le reti tradizionali prendono il nome di "reti piatte".

Per poter modellare quindi questi sistemi a più strati è necessario avere diversi livelli di astrazione dal sistema e andare a perfezionare gli elementi di più alto livello attraverso

[16] Transactions on Petri Nets and Other Models of Concurrency VII
https://link.springer.com/content/pdf/10.1007/978-3-642-38143-0.pdf

elementi di livello inferiore che contengono maggiori dettagli. Questo è quello che si intende per approccio gerarchico. Quindi quello che si fa è suddividere il sistema principale in sistemi più piccoli, così da poter superare la complessità.

2.4.1 L'approccio gerarchico

Come già detto per introdurre la gerarchia nelle Reti di Petri è necessario suddividere un sistema in moduli, che rappresentano sottosistemi. Un modulo s risulta essere composto da Posti, Transizioni, Archi e Gettoni, nulla di nuovo rispetto alla versione classica, e viene rappresentato graficamente da un rettangolo con due bordi. Una rete di Petri Gerarchica è quindi composta da un insieme finito di moduli, S.

Tuttavia, quando si introduce il concetto di modulo è necessario chiarire che esistono due tipi di transizioni: Transizioni Elementari, quelle tradizionali, e Transizioni di Sostituzione. Queste ultime si riferiscono al modulo, essendo un sottoinsieme delle transizioni del modulo $T_{sos} \subseteq T$ la loro funzione è quella di astrarsi dal contenuto del modulo, facendo sembrare quest'ultimo una scatola nera all'interno della quale non si sa quello che avviene. Quindi ad ogni $t_{sos} \in T_{sos}$ è assegnato un modulo s attraverso una funzione $SF : T_{sos} \rightarrow S$.

Questo tipo di transizione viene rappresentata da un rettangolo disegnato all'interno del modulo. Sono proprio queste transizioni che permettono di implementare la gerarchia in quanto vanno a rappresentare un pezzo della struttura della rete. Si tratta di una particolare categoria di transizioni perché di per sé non si attivano ma contengono appunto una sottorete che definisce il comportamento che si verifica al suo posto. Quindi una transizione che può attivarsi o scattare appartiene alla rete principale o alla sotto rete, ma non sarà una transizione di sostituzione. Rispetto a quelle ordinarie le transizioni di sostituzione possono avere posti e transizioni al proprio interno e non hanno bisogno di consumare o produrre gettoni in un'azione atomica.

Inoltre, il numero di Transazioni di Sostituzione all'interno di un modulo non ha limiti e cosa ancora più interessante è che queste stesse potrebbero riferirsi ad altri moduli che possono contenere altre transizioni riferite ad altri moduli e così via a cascata. È questo che permette di mettere in evidenza i vari strati del sistema e mostrare le interdipendenze che sussistono al suo interno.

Al fine di poter comprendere effettivamente il funzionamento di questo tipo di estensione risulta utile introdurre i concetti di Socket e di Porta[17].

Le Socket, $p_{sock}^{(t)}$, costituiscono le interfacce delle Transizioni di Sostituzione con la rete e sono costituite dai Posti in ingresso e in uscita della T_{sos} nella rete di più alto livello. I posti socket non sono definiti in maniera esplicita come componenti di un modulo, ma vengono considerati esplicitamente tramite gli archi collegati alla transizione di sostituzione. Ecco perché andando ad esplorare il contenuto del modulo notiamo come anch'esso abbia delle interfacce, le Porte, p_{porta}. Queste permettono di fatto lo scambio dei gettoni tra il modulo e il resto dell'ambiente (gli altri moduli della rete).

Quindi un modulo può essere definito come una tripla s = (PN, T_{sos}, P_{porta}) nella quale:

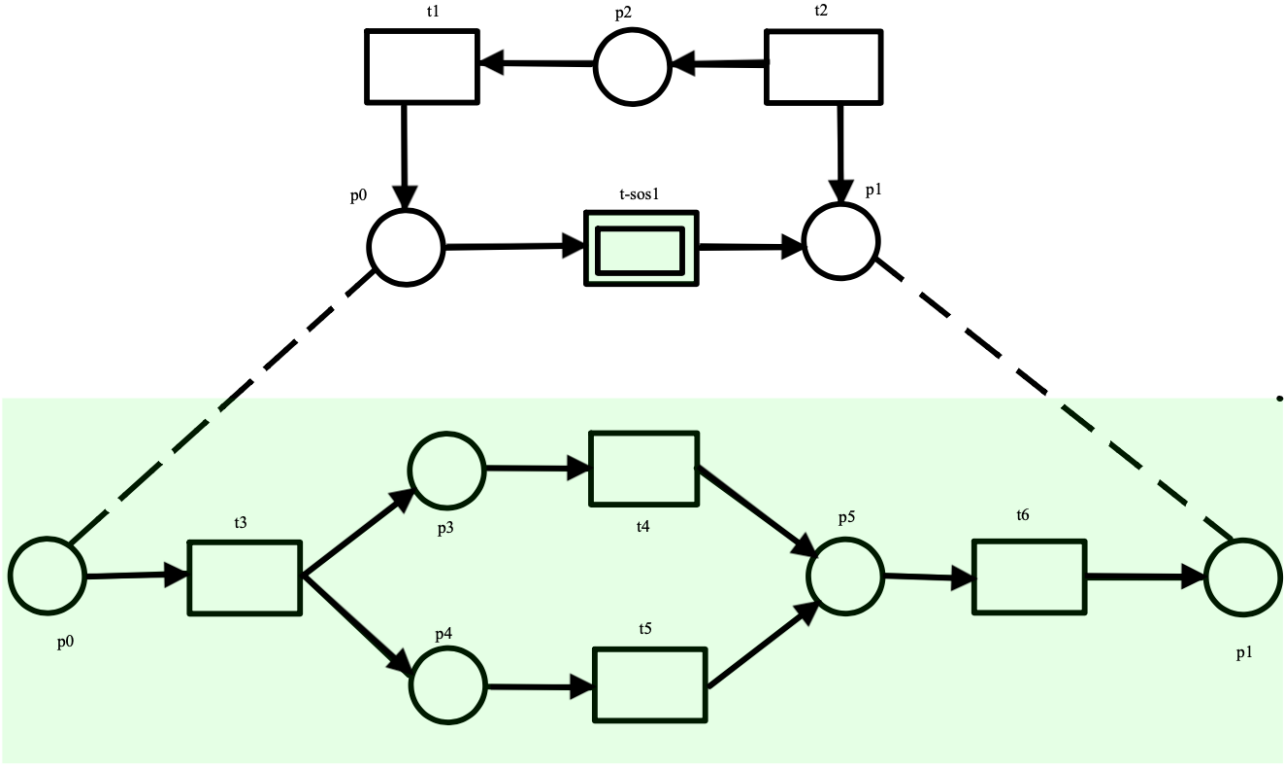
- PN rappresenta una Reti di Petri classica oppure gerarchica se il modulo a sua volta contiene t_{sos};
- T_{sos} è l'insieme delle transizioni di sostituzione;
- P_{porta} è l'insieme dei posti porta.

Porte e Socket svolgono inoltre un compito essenziale nella modellazione in quanto permettono la sostituzione della Transizione di Sostituzione con il contenuto del modulo a cui la transizione si riferisce. Affinché questa sostituzione possa avvenire, essenziale per ottenere un modello gerarchico, è necessario specificare come sono messe in correlazione le interfacce della transizione di sostituzione con le interfacce del modulo. Questo è reso

[17] Coloured Petri Nets, Modelling and Validation of Concurrent Systems http://artamonoviv.ru/wp-content/uploads/2016/08/Coloured_Petri_Nets_modeling_and_validation_of_concurrent_systems__2009.pdf

possibile grazie ad una relazione Porta-Socket, PS(t), la quale assegna ad ogni transizione di sostituzione una relazione PS(t) che mette in relazione una porta del modulo con la corrispondente socket della transizione di sostituzione. Quindi gli elementi di questa relazione sono coppie ($p_{sock}^{(t)}$, p_{port}).

Per facilitare la comprensione è utile assegnare alla porta e socket collegate lo stesso nome. Come conseguenza di questo meccanismo due posti (porta e socket) costituiscono due visioni diverse di un singolo posto. Questo significa anche che condivideranno anche la stessa marcatura e quindi concettualmente diventano un unico Luogo Composto.



(figura 21, Rete di Petri Gerarchica)

Come notiamo nella figura 21 la transizione di sostituzione t-sos$_1$ ha due socket P_0 e P_1. Questi due posti li ritroviamo poi nella sottorete come porte con archi da e verso transizioni diverse. Se quindi i gettoni vengono aggiunti al posto P_0 in seguito allo scatto di t_1 questi specularmente vengono visualizzati anche nella sottorete. La transizione sub_t_1 può quindi

scattare rimuovendo i gettoni da P_0 con conseguente rimozione degli stessi gettoni da P_0 nella rete principale.

Possiamo quindi arrivare a definire in modo completo una Rete di Petri Gerarchica attraverso la tripla (S, SF, PS(t)), cioè:

- S rappresenta l'insieme finito di moduli di cui si compone la rete gerarchica;
- SF: $T_{sos} \rightarrow S$ che identifica la funzione che assegna ad ogni transizione di sostituzione un modulo;
- PS(t) la funzione di relazione porta-socket che assegna ad ogni t_{sos} una relazione per collegare le porte del modulo con le corrispondenti socket della transizione di sostituzione.

Possiamo poi fare un'importante considerazione circa il vantaggio che può portare un approccio di questo tipo. Infatti, la suddivisione del sistema principale in moduli fa si che uno stesso modulo possa essere impiegato più di una volta all'interno dello stesso sistema. Di riflesso più di una transizione di sostituzione faranno riferimento allo stesso modulo e bisogna quindi distinguere tra Definizione di modulo, che ne contiene la specifica, ed Istanza di modulo, che altro non è che la copia del primo. Il risultato di ciò è che avremmo più transizioni di sostituzione riferite alla stessa definizione di modulo ma ognuna di esse sarebbe sostituita solo da una singola istanza di modulo.

Inoltre, partendo da una Rete di Petri Gerarchica è possibile arrivare ad avere una rappresentazione dell'intera rete su un unico livello[18]. Questo è possibile andando semplicemente a sostituire ogni transizione di sostituzione con il modulo che contiene. Questo meccanismo di sostituzione prende il nome di Flattening.

Quanto detto fino ad ora per questa estensione può essere modellato secondo un approccio top-down o all'apposto attraverso un approccio bottom-up. Nel primo caso partendo dal livello più alto a cascata si scompone il sistema nei vari moduli grazie all'uso

[18] Putting high-level Petri nets to work in industry https://pure.tue.nl/ws/files/2028061/612046.pdf

delle transizioni di sostituzione. In questo primo approccio ciò che è importante sono le relazioni tra i vari moduli, in quanto il loro contenuto viene visto come una scatola nera, che nei passaggi successivi (man mano che si scende di livello) prende forma grazie ai maggiori dettagli che vengono aggiunti.

L'altro approccio è l'opposto in quanto di parte dal livello più basso in cui i moduli vengono descritti dettagliatamente e man mano vengono composti fino a raggiungere il livello più elevato di astrazione.

CAPITOLO III

LE RETI DI FLUSSO DI LAVORO

3.1 L'esistenza dei flussi di lavoro e dei Workflow Management Systems

Come ben si sa le persone devono lavorare al fine di sopravvivere. Per mantenersi è infatti necessario disporre di una casa, di cibo, di vestiti e di tutto quanto risulta necessario per un'esistenza dignitosa.

Ovviamente sarebbe una cosa impossibile produrre tutto quanto da sé, considerando l'ammontare di tempo e sforzo necessario, le conoscenze e competenze richieste oltre che i macchinari necessari, tuttalpiù vivendo in una società sempre più complessa e sviluppata. Questa è la ragione che sta alla base del motivo per cui la nostra società/economia è suddivisa in *business unit* specializzate all'interno delle quali le persone lavorano per produrre determinate categorie di prodotti nel modo più efficiente possibile attraverso l'uso di macchinari. L'ormai nota divisione e specializzazione del lavoro.

Bisogna però soffermarsi su un dettaglio molto importante perché è vero si che oggi tutti i beni e servizi di cui disponiamo o che desideriamo possono essere prodotti però è altrettanto vero che questo è diventato sempre più complicato e complesso all'interno delle varie aziende e di conseguenza molte delle persone che vi lavorano non comprendono il motivo per cui fanno le cose che vengono loro dette di fare.

È proprio questo il motivo che spinge le aziende ad organizzare il lavoro interno in processi in modo che questo possa essere compreso senza troppi problemi dalle persone.

Inizialmente c'erano dei semplici sistemi informativi che supportavano la realizzazione di singoli compiti, ma appunto, con l'aumento delle dimensioni e della complessità delle imprese data dalla vasta gamma di prodotti e servizi offerti a cui si aggiunge una sempre

minor durata (vita utile) dei beni con conseguente necessità di rivedere spesso i processi aziendali, questo non è stato più sufficiente, l'intero processo aziendale e quindi il flusso di lavoro all'interno andava monitorato. Ecco perché si sono sviluppati i *workflow management systems*[19] con lo scopo di controllare il lavoro e facilitare lo scambio di informazioni tra le persone all'interno delle organizzazioni. Un sistema di gestione del flusso di lavoro (WFMS) altro non è che un software che rende possibile la definizione, l'esecuzione, la registrazione e il monitoraggio dei flussi di lavoro.

I sistemi di gestione dei flussi di lavoro vengono usati con l'obiettivo di semplificare e coordinare i processi aziendali[20]. Questi ultimi vengono rappresentati da flussi di lavoro, cioè modelli informatizzati del processo stesso, la cui utilità è quella di specificare quali task devono essere svolti, il loro ordine, le condizioni da rispettare, il flusso di dati tra i vari compiti, chi ha la responsabilità sull'esecuzione dei tasks, quali strumenti/risorse usare nello svolgimento dei compiti. Un WFMS è quindi uno strumento che rende possibile la progettazione e definizione dei flussi, la loro istanziazione ed esecuzione controllata a cui si aggiunge anche il coordinamento e l'integrazione di applicazione all'interno del workflow.

Data l'estrema importanza del controllo dei flussi di lavoro è di grande aiuto l'uso di modelli di riferimento ampiamente consolidati al fine di modellare e analizzare i processi dei flussi di lavoro. Uno di questi strumenti sono appunto le Reti di Petri di cui abbiamo trattato fino ad ora. Queste infatti risultano estremamente utili nella modellazione e analisi di tutti i tipi di processi, con applicazioni che spaziano da protocolli, hardware a sistemi di produzione e processi aziendali. Inoltre, le estensioni di queste reti, di cui già abbiamo parlato, facilitano notevolmente il dar forma a processi più complessi all'interno dei quali dati, tempo e gerarchia giocano un ruolo cruciale.

[19] The application of Petri Nets to Workflow Management
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.30.3125&rep=rep1&type=pdf
[20] A Research Perspective on Workflow Management Systems
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.28.3164&rep=rep1&type=pdf

L'uso delle Reti di Petri infatti grazie alla loro semantica definita in modo formale permette di definire in modo preciso e chiaro un workflow, e grazie alla loro potenza grafica sono molto intuitive e favoriscono la comunicazione. Dispongono inoltre di tutte le strutture modellistiche e di instradamento necessarie per la creazione di un flusso di lavoro.

3.2 Concetti base nell'ambito dei WFMS

I sistemi di gestione del flusso di lavoro (WFMS) consentono alle aziende di raggiungere i loro obiettivi tramite una sequenza di attività e con l'ausilio di risorse umane e informative. Visto che il ricorso alle reti di computer e ai sistemi informativi per supportare le attività lavorative è sempre più frequente, l'uso di questi sistemi attrae sempre più imprese.

Come già detto un WFMS è un sistema che supporta i flussi di lavoro grazie ad un software guidato da una rappresentazione informatica del flusso.

Ci sono alcuni importanti concetti che è necessario conoscere per comprendere maggiormente questo strumento[21]:

- **Modello del flusso di lavoro o definizione di processo**, è la rappresentazione computerizzata del processo e va a definirne le condizioni, le attività che lo compongono e come i dati fluiscono tra queste e le regole per il flusso. Sostanzialmente un buon modello dovrebbe avere una buona capacità di rappresentazione ed essere semplice e facile da modificare in modo da adeguarsi al variare delle condizioni;
- **Activity**, che consiste in un passaggio logico del flusso il quale comprende informazioni circa le condizioni da rispettare, gli utenti e risorse coinvolte, i dati ed i vincoli essenziali per portare a termine l'attività;

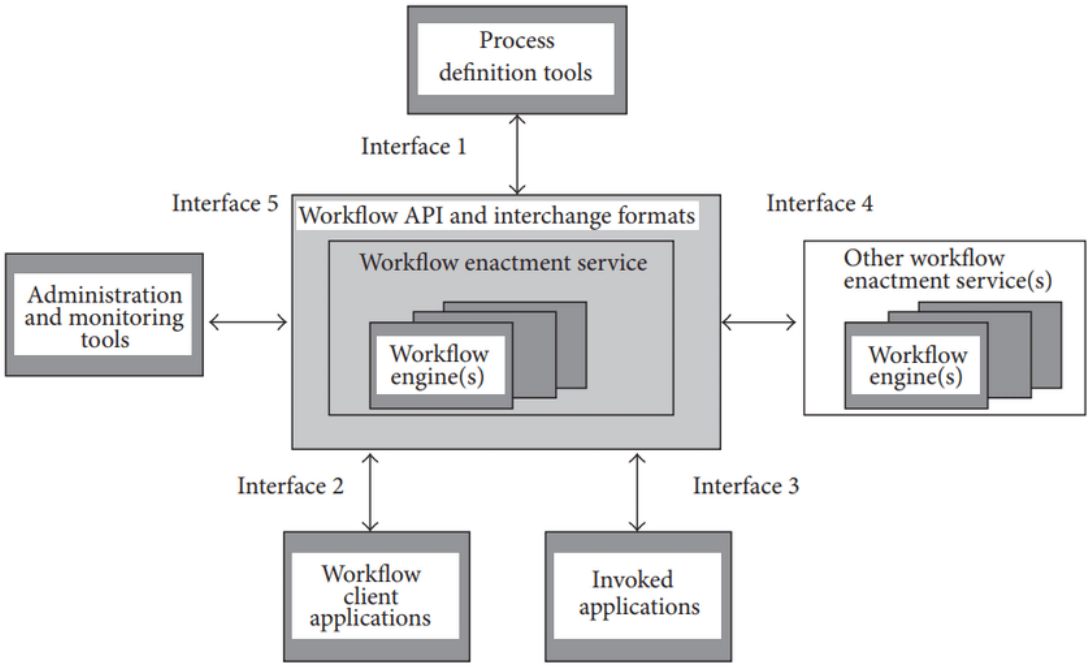
[21] Workflow Management Systems: A Survey https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=740974

- **Istanza di processo**, cioè l'esecuzione di una definizione di processo da parte del sistema di gestione del flusso di lavoro. Il sistema interpreta la definizione e monitora la sua istanziazione, le varie attività e la loro sequenza e aggiunge i work-item alle liste di lavoro.

Negli anni Novanta la Workflow Management Coalition ha creato un modello di riferimento (figura 22) per definire in maniera completa le varie componenti di questo tipo di sistema, di cui di seguito viene fornita una semplice spiegazione:

- lo strumento per la definizione del processo, un tool con cui gli utenti possono analizzare a dare forma ai processi aziendali e creare una rappresentazione computerizzata di essi. Questo a sua volta è in relazione con il servizio di attuazione del flusso di lavoro attraverso un'interfaccia che consente lo scambio di informazioni sulla definizione del processo;
- servizio di attuazione del flusso di lavoro, che fornisce un "ambiente" nel quale si verifica l'istanziazione e l'attivazione del processo. Ci possono essere anche più servizi di attuazione per completare l'esecuzione di un flusso di lavoro distribuito ed è quindi necessaria un'interfaccia per consentire scambi tra questi;
- le applicazioni client, le quali permettono agli utenti di elaborare i tasks che necessitano di interazioni con persone quando il processo è in esecuzione. Questi tasks prendono il nome di work-item, che include dei requisiti di elaborazione, risorse ed applicazioni coinvolte. Una lista dei lavori viene creata per ogni utente e contiene tutti i work-item che da esso devono essere eseguiti;
- applicazioni, che vengono richiamate dal servizio di attuazione per l'esecuzione di particolari attività, solitamente si tratta di applicazioni basate su server e prive di interfacce utente;
- strumenti di amministrazione e monitoraggio la cui funzione è quella di svolgere attività amministrative comuni, ad esempio la gestione degli utenti e dei ruoli o audit e controllo delle risorse, inoltre svolgendo anche funzioni di supervisione dell'istanza di processo interagisce con il servizio di attuazione del flusso di lavoro.

Il Workflow Application Programming Interface rappresenta la combinazione di tutte queste componenti e fornisce un insieme di chiamate API e dei formati di interscambio per supportarle.



(figura 22, framework per I WFMS della Workflow Management Coalition,

fonte: Workflow Management: Models, Methods and Systems)

3.3 Concetti base nell'ambito dei flussi di lavoro

Quando si parla di gestione dei flussi di lavoro si vuol fare riferimento alla logistica dei processi aziendali il cui scopo altro non è che quello di garantire che tutte le varie attività di cui si compone un processo vengano eseguite dalla persona giusta e al momento giusto. Risulta fondamentale citare la definizione di sistema di gestione del flusso di lavoro fornita dalla Workflow Management Coalition[22]: "*un sistema che definisce, gestisce ed esegue*

[22] Workflow Management Coalition Terminology & Glossary
https://healthcareworkflow.files.wordpress.com/2008/10/wfmc-tc-1011_term_glossary_v3.pdf

completamente i flussi di lavoro attraverso l'esecuzione di software il cui ordine di esecuzione è guidato da una rappresentazione al computer della logica del flusso di lavoro".
Quindi nel concreto questi sistemi concretizzano tecniche, concetti e metodi che risultano fondamentali per gestire i flussi di lavoro.

Un workflow o flusso di lavoro viene utilizzato per rappresentare il modello di un processo aziendale attraverso la sua razionalizzazione e suddivisione in diverse fasi con lo scopo di portare miglioramenti incrementali nelle performance del processo stesso ed efficientare l'uso delle risorse.

I flussi di lavoro sono basati su casi e quindi ogni pezzo del processo viene realizzato per un determinato caso. Questo rappresenta di fatto la sola cosa, tangibile o intangibile, che viene prodotta o modificata attraverso il flusso di lavoro. L'obiettivo ultimo della gestione dei flussi di lavoro è quello di completare i casi nel miglior modo possibile nel rispetto dei principi di efficacia ed efficienza. Esempi di casi possono essere il processo di assemblaggio di una bici da corsa, la progettazione di un edificio, raccogliere i risultati di un sondaggio per fare una statistica, eseguire un reclamo per un risarcimento ad un'assicurazione.

Portare avanti un caso risulta essere un'attività di natura discreta dato che un caso ha un inizio e una fine ben precisi e inoltre è distinguibile da qualsiasi altro caso.

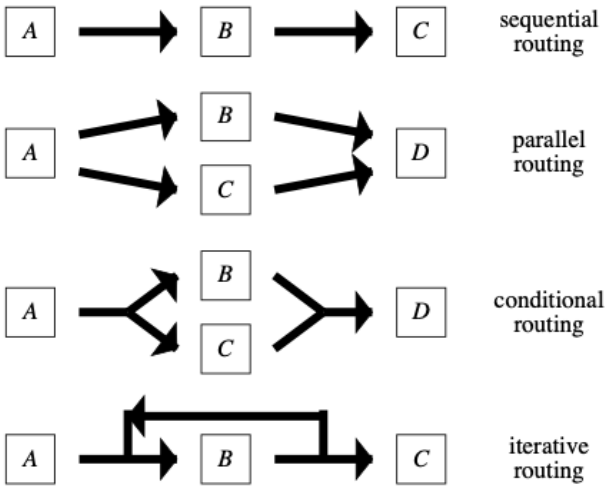
Ovviamente ogni caso prevede lo svolgimento di un processo o procedura, il quale si compone di tasks che devono essere compiuti e di un insieme di condizioni che, corrispondendo alle dipendenze causali tra i vari compiti, vanno a stabilire l'ordine di esecuzione dei tasks. Ognuno ha delle pre e post condizioni, la quali devono essere soddisfatte rispettivamente prima e dopo il compimento dei tasks.

Possiamo considerare i tasks come un'unità logica di lavoro svolta come un tutt'uno da una risorsa. Il termine risorsa risulta essere un qualcosa di generico che indica una persona o un gruppo di persone, un macchinario o un insieme di macchine oppure un insieme di risorse umane e macchine.

Nel definire tutti i vari tasks di un processo possiamo vedere come ci siano quattro meccanismi di base[23], o costrutti, nella struttura di quest'ultimo che prendono il nome di routing (figura 23):

- sequenza, si verifica quando due o più tasks devono essere eseguiti rigorosamente uno dopo l'altro. Nella figura qui sotto possiamo vedere come il task B deve essere eseguito dopo che il task A è stato portato a termine mentre il task C può iniziare solo dopo che il task B è stato completato;
- selezione, che troviamo nel caso in cui si deve compiere una scelta tra quale tasks deve essere eseguito. Nella figura possiamo vedere che una volta che il task A è stato eseguito bisogna scegliere se portare avanti il task B o il C, successivamente il task D potrà essere iniziato;
- parallelizzazione, in questo tipo di routing due o più task possono essere svolti contemporaneamente o in qualsiasi ordine, dato che l'esecuzione di un task non va ad influire su quella dell'altro. Inoltre, questi due compiti devono essere completati entrambi affinché il successivo possa cominciare, questo viene chiamato sincronizzazione;
- iterazione, in tutti quei casi in cui è necessario eseguire un task diverse volte (per esempio quando un compito deve essere reiterato fino a quando l'esito del test che viene eseguito non è positivo).

[23] The application of Petri Nets to Workflow Management
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.30.3125&rep=rep1&type=pdf



(figura 23, quattro costrutti di instradamento,

fonte: The application of Petri Nets to Workflow Management)

Un task può essere inteso anche come un processo atomico, cioè non scomponibile ulteriormente. Tuttavia, questo dipende da che punto di vista si effettua l'analisi, infatti "da fuori" un task può essere semplicemente visto come una parte del processo, ma d'altro canto per chi effettivamente lo deve svolgere può prevedere diversi step. Considerando il processo produttivo di un paio di occhiali ad esempio il momento in cui le aste vengono attaccate al pezzo frontale dell'occhiale appare come una singola azione, ma per chi la esegue diversi sono i passaggi, ad esempio controllare che le attaccature per le aste sul frontale corrispondano alle aste, controllare le cerniere, avvitare le aste alle cerniere ecc. Inoltre, lo svolgimento di un task da parte di una risorsa ad esso dedicata prende il nome di attività.

Un unico processo viene eseguito su ogni caso, vari casi possono condividere lo stesso processo anche se ognuno di questi può compiere un percorso diverso al suo interno. Il tipo di tragitto che un caso segue all' interno del processo dipende dalle sue caratteristiche specifiche (gli attributi del caso).

Tipicamente la quantità di processi interni all'azienda è in numero finito e comunque minore rispetto all'ammontare dei casi che devono essere gestiti. Di conseguenza un'impresa può sviluppare delle routine per lo svolgimento dei processi e guadagnare in efficienza. L'ideale sarebbe avere un piccolo numero di processi e tenere il più alto possibile il numero di casi che questi affrontano. Sviluppando sostanzialmente delle economie di scala.

Risulta si importante avere pochi processi però questi non devono essere troppo complicati, per non perdere efficienza, quindi delle volte può essere meglio avere un po' più processi che risultano però più semplici.

C'è però una sorta di eccezione al riguardo che può essere spiegata attraverso un semplice esempio: un sarto crea tutti gli abiti su misura, e di conseguenza serve un nuovo processo per ogni abito (il nostro caso). Vedendola da un'altra prospettiva in realtà il processo che viene seguito è sempre lo stesso, perché qualunque sia l'abito l'artigiano deve prendere le misure del cliente, scegliere il modello insieme ad esso e tutto quello che riguarda la fase di disegno e di finitura dell'abito. Quindi sostanzialmente viene seguito un approccio standard ma i vari task che dovranno essere eseguiti dipendono strettamente da caso a caso. Il grado di dipendenza dei tasks dal caso rappresenta quindi un parametro per valutare la complessità di un processo.

3.3.1 La relazione tra caso, task, work-item e activity

La qualità dei flussi di lavoro è una caratteristica essenziale per il successo di un sistema di gestione dei flussi di lavoro, è quindi necessario precisare un'importante relazione esistente tra alcuni concetti fondamentali in materia di workflow, ovvero quella che intercorre tra caso, task, attività e work-item[24].

[24] Workflow Management: Models, Methods and Systems
https://books.google.it/books?hl=it&lr=&id=O1xW1_Za-I0C&oi=fnd&pg=PR16&dq=%22Workflow+Management—Models,+Methods+and+Systems%22&ots=Ibdr76AiXN&sig=hK2TPI61uBLp5d-

Come abbiamo già accennato la ragion d'essere di un workflow è quella di trattare dei casi. Un semplice esempio può essere il processo di assemblaggio di un'automobile. Casi simili inoltre appartengono allo stesso tipo di caso, e vengono quindi trattati allo stesso modo, tuttavia proprio per poter distinguere un caso dall'altro ognuno di questi possiede una propria identità. Bisogna poi precisare che essendo il caso un evento discreto (quindi con un inizio e una fine) questo di conseguenza ha una durata limitata, quindi un caso che compare all'interno di un workflow deve anche poi scomparire una volta che il suo percorso all'interno del processo è giunto al culmine. All'interno di questa durata il caso ha sempre un particolare stato, che si compone di tre elementi: i valori degli attributi rilevanti del caso, le condizioni che sono state soddisfatte e il contenuto del caso. Gli attributi sono delle variabili che possono essere attribuite al caso e vengono usati per gestirlo, per esempio potrebbero indicare che un certo task su quel caso può essere saltato. Inoltre, i valori di questi attributi possono cambiare man mano che un caso avanza lungo il suo percorso. Per capire invece lo stato di avanzamento del caso bisogna guardare le sue condizioni, le quali permettono di comprendere quali tasks sono già stati completati e quali devono essere ancora eseguiti. Possiamo intenderle anche come dei prerequisiti necessari affinché un task possa essere portato avanti, quindi solo quando tutte le condizioni di un task sono soddisfatte questo può essere eseguito.

Infine, il contenuto del caso non è parte integrante del sistema del flusso di lavoro ma i dati riguardanti ad esso sono contenuti in file, documenti ecc.

Altro concetto importante è quello di task, in quanto la loro individuazione è fondamentale per poter strutturare un flusso di lavoro. Come già detto non sono altro che un'unità logica di lavoro che deve sempre essere svolta per intero, è quindi indivisibile nel senso che o viene eseguito tutto o non viene eseguito, infatti se nella fase di svolgimento

G6YBAlUwZHS4&redir_esc=y#v=onepage&q=%22Workflow%20Management—Models%2C%20Methods%20and%20Systems%22&f=false

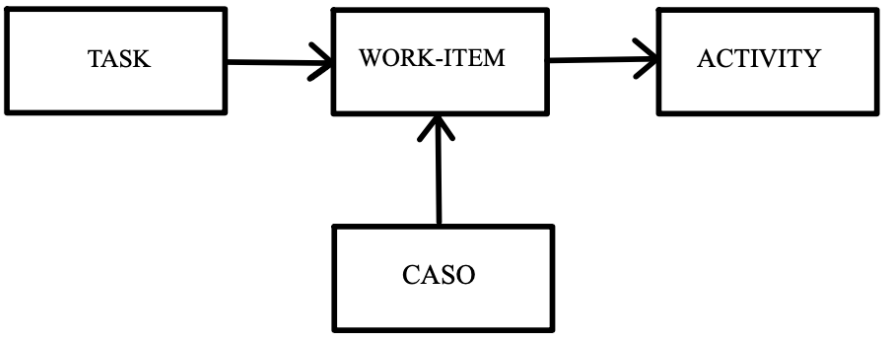
del task qualcosa non va per il verso giusto è necessario ricominciare da capo, quello che viene definito *rollback*.

Possiamo poi distinguere tre categorie di task: manuali, automatici o semiautomatici. I primi ovviamente sono quelli svolti esclusivamente da persone, come eseguire un controllo fisico, i secondi all'apposto sono portati a termine senza intervento umano, questo sta a significare che questo tipo di task può essere svolto, ad esempio da un programma, grazie a dati registrati in precedenza. Nell'ultimo caso vediamo l'intervento di persone e macchine.

Quindi dando una definizione semplice ma efficace un task rappresenta solamente una parte generica di un lavoro, non ha niente a che fare con l'esecuzione del task come parte di un determinato caso. A questo scopo abbiamo due altri concetti importanti. Con il termine "*work-item*" si è soliti indicare la combinazione di un caso e di un task che deve essere svolto e viene creato nel momento in cui lo stato di un caso lo consente. Sostanzialmente è una vera e propria parte di lavoro che può essere eseguita. Infine, con il termine "*activity"* si intende l'esecuzione effettiva di un work-item. Quindi nel momento esatto in cui inizia la lavorazione di un work-item siamo di fronte ad una activity.

Quindi rispetto ad un task che è un qualcosa di generico sia il work-item che l'activity risultano essere correlati/abbinati ad uno specifico caso.

Nella figura 24 possiamo notare la relazione esistente tra i concetti appena spiegati.



(figura 24, relazione esistente tra Caso, Task, Work-item e Activity)

3.4 La gestione delle risorse nei flussi di lavoro

Come abbiamo già detto un processo va a stabilire quali tasks devono essere svolti per una particolare categoria di casi e anche l'ordine in cui ciò deve essere fatto. Ma risulta di grande importanza, soprattutto considerando l'efficacia e l'efficienza del flusso, il modo in cui i work-item sono assegnati alle risorse, che possono essere sia persone che macchine. Possiamo quindi intuire che le risorse sono fondamentali per la buona riuscita di un flusso di lavoro, il quale supporta un processo[25].

La principale caratteristica di una risorsa è che essa è in grado di svolgere determinati tasks. Possiamo quindi precisare come una risorsa sia in grado di eseguire un numero limitato di tasks, ad esempio in una banca uno sportellista non può concedere un finanziamento, ma è anche vero che un task può essere compiuto solo da un numero limitato di risorse. Risulta quindi estremamente utile suddividerle in classi di risorse, una classe è un gruppo di risorse, inoltre una risorsa può appartenere a più classi.

Tipicamente esistono due modalità per classificare le risorse, una basata sulle proprietà funzionali e una basata sulla loro posizione all'interno dell'organizzazione. Una classe basata sulla classificazione del primo tipo prende il nome di ruolo (o funzione) e consiste in un gruppo di risorse ognuna delle quali ha specifiche competenze, e quindi sufficientemente qualificata per svolgere il task che le viene assegnato. Invece un insieme di risorse classificate in base alla loro posizione viene definito unità organizzativa. Questo assicura che un task venga eseguito nel giusto posto all'interno dell'organizzazione.

Per essere certi che ogni activity venga "lavorata" dalla giusta risorsa ad ogni task viene assegnato un principio di allocazione, rappresentante un ponte tra il processo e le risorse a

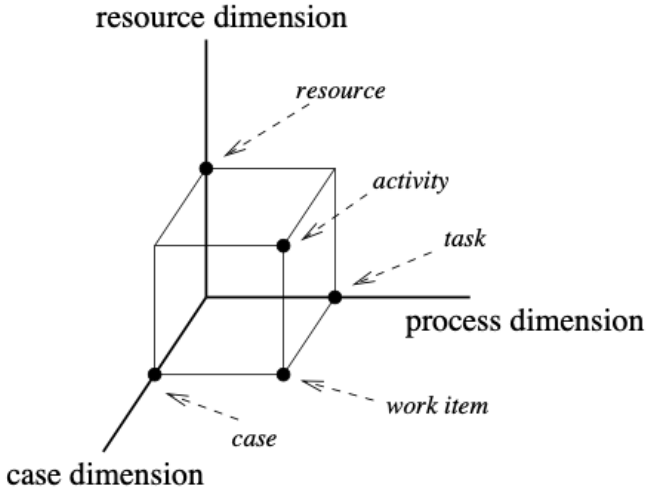
[25] The application of Petri Nets to Workflow Management
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.30.3125&rep=rep1&type=pdf

disposizione, che va ad esplicitare quali sono le pre-condizioni che la risorsa deve soddisfare. Va quindi a specificare il ruolo e l'unità organizzativa a cui la risorsa deve appartenere. L'allocazione in molti casi è anche strettamente correlata agli attributi specifici del caso per il quale il task va compiuto. Facendo quindi attenzione a questi attributi è possibile far sì che una activity sia eseguita da una specifica risorsa.

Inoltre, nel caso in cui una sequenza di tasks deve essere eseguita da/o sotto la supervisione di una singola risorsa/persona, questa prende il nome di "case manager". Questa figura è molto utile perché essendo responsabile di un caso sicuramente sarà molto coinvolta in esso e questo si traduce in un migliore servizio e una più rapida conclusione del caso grazie alla maggior dimestichezza con casi di quel tipo.

Ci sono però delle situazioni in cui ci sono più risorse che possono eseguire l'activity relativa ad un determinato work-item. Fortunatamente il motore del flusso di lavoro, che assicura l'attuazione del flusso, ha tra i suoi compiti quello di assegnare i work-item alle risorse tenendo conto ovviamente delle classi di risorse specificate come adatte.

Sulla base di quanto detto fino ad ora possiamo capire come un flusso di lavoro abbia tre dimensioni (figura 25): la dimensione caso; la dimensione processo; la dimensione risorse.



(figura 25, le tre dimensioni di un flusso di lavoro,

fonte: The application of Petri Nets to Workflow Management)

La prima dimensione sta ad indicare che ogni caso viene trattato individualmente, quindi nessun caso può influenzarne un altro. Bisogna però precisare che i casi si influenzano in maniera indiretta dal momento che condividono le risorse di cui dispone l'organizzazione. Nella seconda dimensione viene definito il processo del flusso di lavoro ovvero tutti i tasks e il percorso che i casi compiono al suo interno. Nell'ultima dimensione si specificano le classi di risorse che vengono assegnate ai vari work-item.

Dalla figura 23 notiamo come un work-item mette in relazione casi e tasks, invece un activity collega tutte e tre le dimensioni, caso-processo-risorse. Grazie alla rappresentazione di cui sopra un flusso di lavoro può essere visto come i vari punti presenti in tutte e tre le dimensioni.

3.4.1 I principi di allocazione delle risorse

Dato che l'obiettivo di flusso di lavoro è quello di completare i casi in modo efficiente ed efficace è di fondamentale importanza eseguire i vari work-item nel modo più rapido possibile, perché in caso contrario un ritardo nell'esecuzione comporta che il caso richieda più tempo per essere ultimato.

È quindi cruciale definire l'ordine in i work-item vengono trasformati in activity e da quali risorse queste ultime sono eseguite. Si tratta di decisioni che si influenzano a vicenda perché l'ordine da stabilire è importante per la selezione delle risorse ma la scelta di queste va ad influire sull'ordine in cui i work-item vengono trasformati.

Diverse sono le modalità da applicare al fine di stabilire l'ordine di avanzamento, le quali prevedono l'accodamento dei vari work-item:

- FIFO (First-In, First-Out), quando i work-item vengono portati avanti in base all'ordine in cui sono stati creati, si tratta del tipo di allocazione più utilizzato;

- LIFO (Last-In, First-Out), l'opposto del precedente e prevede di trattare per primi I work-item create più recentemente;
- Tempo di elaborazione più breve/lungo, andando a stimare in anticipo quanto tempo richiede l'esecuzione delle varie activity, sulla base degli attributi del caso, è possibile distinguere tra casi facili che richiedono poco tempo per essere completati e casi più difficili e dispendiosi in termini di tempo. Affrontando prima i casi più facili è possibile ridurre il tempo medio del flusso dei casi, mentre in alcuni casi può essere conveniente trattare prima quelli difficili;
- Tempo di elaborazione residuo più breve/lungo, se si conosce il tempo per svolgere le varie activity e l'instradamento di un caso è possibile stimare il suo tempo netto di elaborazione residuo. Scegliendo le activity con il tempo più breve è possibile ridurre al minimo la quantità di lavori in corso
- Prima Data di Scadenza, l'ordine di esecuzione dipende dal momento (data di scadenza) in cui il caso deve essere completato.

L'ordine come abbiamo detto è correlato alla selezione delle risorse. Nel selezionare le risorse adatte per lavorare sui vari work-item è utile seguire alcuni principi. Le risorse possono eseguire vari tasks, ma su certi sono più specializzate e quindi ogni risorsa dovrebbe poter lavorare sui tasks su cui meglio può performare. Una risorsa dovrebbe poi poter svolgere tasks simili in successione, in modo da ridurre i tempi di set-up e ottenere guadagni di tempo sviluppando delle routine. È fondamentale poi cercare di essere il più flessibili possibile nel senso che disponendo di più risorse per svolgere un work-item è utile farlo eseguire a quella che sa eseguire meno elementi di lavoro di altri tipi, si cerca quindi di "preservare" le risorse generaliste.

3.4.2 Il diagramma ER

I concetti di caso, attributi del caso, processo, tasks, work-item, activity, risorse e classi di risorse sono strettamente connessi l'uno con l'altro (come mostrato dalla figura 24). Di conseguenza l'allocazione delle risorse alle activity non è cosa semplice.

Al fine di rendere tutto un po' più chiaro si può ricorrere all'uso di un modello che consente di riassumere tali concetti e le loro connessioni: un diagramma ER (entity relatioship)[26]. Si tratta di un modello abbastanza semplice, contiene infatti due soli elementi, tipi di entità e tipi di relazioni, indicati rispettivamente da un rettangolo e da un rombo. Per esempio, il tipo di entità "task" può contenere tutti i tasks che compongono un processo. Invece i tipi di relazioni possono precisare l'appartenenza di una risorsa ad una certa classe o indicare a quale task un work-item è associato, in questo senso possiamo capire come un work-item abbia una relazione con un solo task e di contro come un task possa avere più di un work-item ad esso associato, questo viene rappresentato attraverso i simboli 1 e N che rappresentano le cardinalità della relazione. Una relazione precisa anche che un work-item è correlato ad un caso specifico, e che di contro un caso specifico può essere messo in relazione con più work-item.

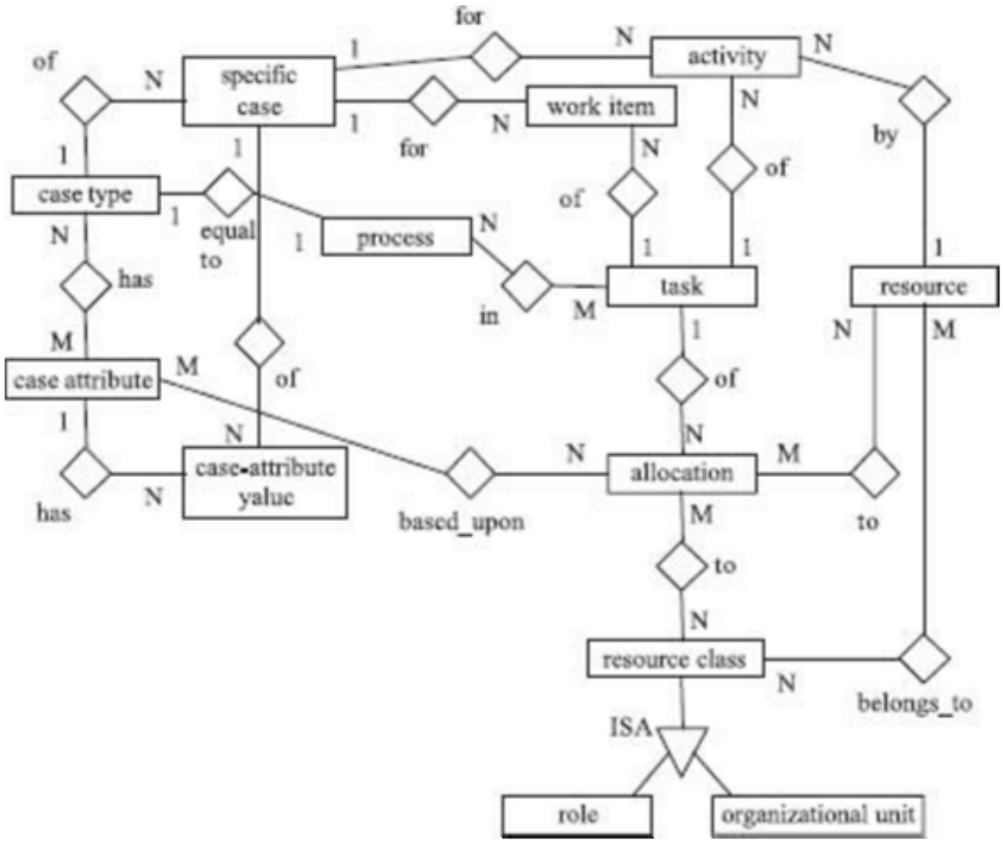
Il tipo di entità "activity" si riferisce all'effettiva esecuzione di un work-item e di conseguenza si riferisce ad un singolo caso e un singolo task. Inoltre, a ciascuna activity vengono assegnate delle risorse, le quali possono appartenere a diverse classi di risorse, la quali a loro volta possono contenere diverse risorse, si tratta quindi di una relazione molti a molti (M a N).

[26] Workflow Management: Models, Methods and Systems
https://books.google.it/books?hl=it&lr=&id=O1xW1_Za-I0C&oi=fnd&pg=PR16&dq=%22Workflow+Management—Models,+Methods+and+Systems%22&ots=Ibdr76AiXN&sig=hK2TPI61uBLp5d-G6YBAlUwZHS4&redir_esc=y#v=onepage&q=%22Workflow%20Management—Models%2C%20Methods%20and%20Systems%22&f=false

All’interno di questo tipo di diagramma è anche possibile osservare la distinzione tra un tipo di caso e un caso specifico. Il primo coincide con il processo e fa riferimento alla categoria di casi che possono essere eseguiti da esso, quindi esiste una relazione uno ad uno tra il tipo di caso e il processo.

Un ulteriore differenza che il diagramma permette di esplicitare è quella tra attributi del caso, che altro non sono che i nomi che esprimono appunto una particolare proprietà del caso, e attributi associati ad uno specifico caso, che sono i valori relativi agli attributi di un determinato caso. Questo consente di allocare, attraverso una relazione, alle activity risorse che soddisfano determinate condizioni. L’allocazione risulta sempre difficoltosa dato che coinvolge diversi elementi e in aggiunta un task può avere anche più allocazioni che dipendono a loro volta da uno o più attributi del caso.

Di seguito forniamo un esempio di diagramma ER (figura 26).



(figura 26, diagramma ER, fonte: Workflow Management: Models, Methods and Systems)

3.5 Le Reti di Flusso di Lavoro

Gli strumenti per la gestione dei flussi di lavoro tendono a scindere la modellazione del processo del flusso dalla modellazione della struttura organizzativa e delle sue risorse; come possiamo infatti notare dalla figura 25.

Questo avviene principalmente per due ragioni: si può ridurre la complessità e si possono mutare i processi senza cambiare il modello organizzativo.

In questo senso le Reti di Petri costituiscono uno strumento valido per la modellazione dei processi.

Di fatto grazie alla loro potenza espressiva e grafica risulta abbastanza semplice definire un processo con le Reti di Petri, i vari tasks che compongono il processo sono modellati dalle Transizioni, le condizioni vengono rappresentate dai Posti e infine i casi che un processo deve trattare altro non sono che i Gettoni.

Come abbiamo già detto molti sono i casi che possono essere gestiti da uno stesso processo, quindi i gettoni relativi ai vari casi risiederanno nella stessa rete e per non correre il rischio di fare confusione e mischiarli è possibile ricorrere all'estensione del colore che consente di attribuire ad ogni gettone un'informazione sull'identità del caso corrispondente (identificatore del caso, ad esempio caso1, caso2 ecc). Quindi di conseguenza le transizioni possono essere attivate solo se le loro precondizioni contengono lo stesso identificatore del caso.

Dopo aver fornito questi concetti semplici ma essenziali possiamo andare a definire una Rete di Petri che modella il processo di un flusso di lavoro, chiamata anche Rete di Flusso di Lavoro. Una rete di questo tipo ha due requisiti[27]:

[27] A Petri Net-based Workflow Modeling for a Human-centric Collaborative Commerce System https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.121.9183&rep=rep1&type=pdf

- è dotata di un posto di input “i” (detto anche posto sorgente) e di un posto di output “o”, quindi un gettone nel posto “i” significa che c’è un caso che deve essere gestito, viceversa un gettone nel posto “o” corrisponde ad un caso che ha finito il suo percorso all’interno del processo ed è stato quindi completato;
- se viene aggiunta alla Rete una transizione t* che collega il posto di output a quello di input la rete che ne deriva è fortemente connessa.

I gettoni in una rete di flusso di lavoro indicano lo stato del flusso di lavoro di un determinato caso. Un caso possiede poi attributi relativi al flusso di lavoro e dati. I primi corrispondono all’informazione relativa all’indirizzamento dei casi, possono essere quindi intesi come delle variabili di controllo o parametri logistici. Si usa poi il termine attributo del caso per indicare l’attributo del flusso di lavoro relativo al caso. I dati invece non servono per gestire il flusso ma per eseguire i vari tasks.

Infine, lo stato generale di un flusso di lavoro è rappresentato dalla distribuzione dei gettoni lungo i posti della rete, ovvero la marcatura.

3.5.1 Costrutti di instradamento

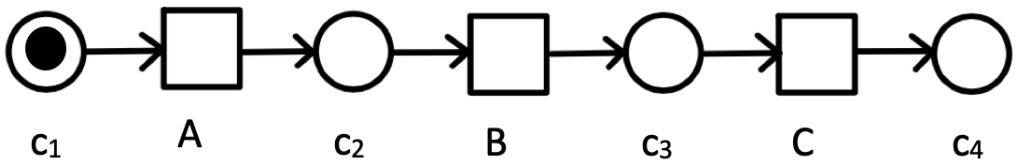
Modellando un processo con Reti di Petri si ricorre ad alcuni building blocks per dare forma ai quattro tipi di instradamento (sequenza, selezione, parallelizzazione e iterazione), ovvero AND-split, AND-join, OR-split, OR-join[28].

Di seguito si spiega come questi costrutti vengono modellati.

L’instradamento sequenziale all’interno di un processo viene usato quando si ha a che fare con delle relazioni causali tra tasks, ovvero quando un task può essere eseguito solo dopo che il precedente è stato portato a termine. La modellazione attraverso Reti di

[28] The Application of Petri Nets to Workflow Management
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.30.3125&rep=rep1&type=pdf

Petri (figura 27) prevede l'aggiunta di un posto tra le transizioni che rappresentano i tasks in questione in modo che rappresenti al tempo stesso la post-condizione di uno e la pre-condizione dell'altro andando così a creare la relazione causale.



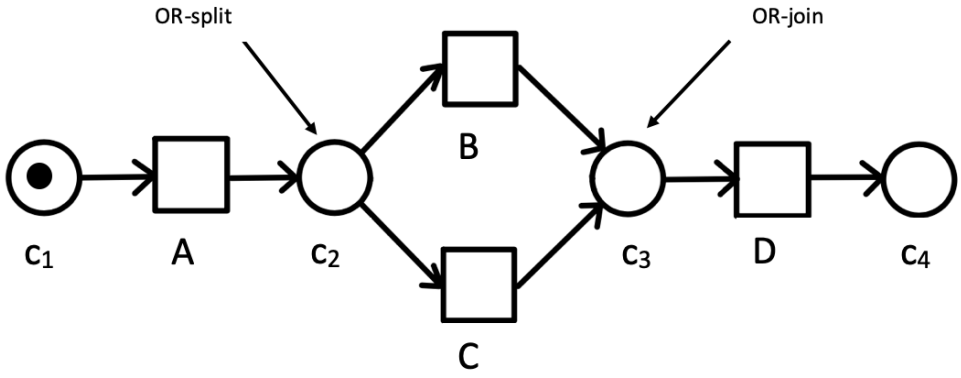
(figura 27, modellazione con Rete di Flusso di lavoro dell'istradamento sequenziale)

Possiamo notare dalla figura come il posto/condizione c2 modella la relazione tra il task A e il B, solo con lo scattare di A si genererà un gettone in c2 che permetterà di abilitare B.

All'instradamento parallelo viene fatto ricorso quando l'ordine in cui devono essere eseguiti i tasks non è così stringente o arbitrario. Per fare questo si ricorre a due building blocks di cui abbiamo già accennato, l'AND-split e l'AND-join. Possiamo vedere questi due blocchi, in questo caso modellati da transizioni, come delle attività di controllo che vengono inserite per motivi di instradamento. Un AND-split può quindi abilitare due tasks la cui esecuzione sarà parallela, invece un AND-join viene abilitato solo dopo che i due tasks sono stati ultimati, la sua funzione è quindi quella di sincronizzare i due sotto flussi che si sono venuti a creare. Questi due sotto flussi, ognuno dei quali serve per eseguire un certo task, potrebbero prevedere un processo più o meno complesso per portare a termine i tasks e sarebbero modellabili in modo abbastanza semplice attraverso l'estensione gerarchica.
Come abbiamo detto l'ordine in cui i tasks posso essere portati avanti è arbitrario, quindi l'esecuzione si potrebbe sovrapporre nel tempo e quindi coincidere oppure nel caso in cui non sia permesso eseguire due tasks relativi al medesimo caso contemporaneamente (ad esempio per evitare problemi tecnici derivanti dalla condivisione dei dati da parte dei tasks) si parla di routing semi-parallelo.

L’instradamento condizionale (o scelta) è previsto quando il tipo di percorso che un caso deve compiere all’interno del processo varia a seconda del caso. Questo ad esempio può dipendere dagli attributi del caso o dal carico di lavoro dell’organizzazione.

In questa situazione si ricorre ad altri due blocchi, l’OR-split e l’OR-join (figura 28), che vengono modellati rispettivamente con un posto con una molteplicità di archi in uscita e con un posto con più archi in entrata. Nel caso dell’OR-split un task può essere seguito dall’esecuzione di uno tra due (o più) tasks. Essendo questo costituito da un posto con più archi in uscita verso altri tasks è evidente come questi ultimi condividano una pre-condizione (c2). Quando quest’ultima contiene un gettone allora una scelta deve essere compiuta, la quale può essere non deterministica o deterministica nel caso in cui dipenda dagli attributi del flusso di lavoro.



(figura 28, modellazione con Rete di Flusso di lavoro dell’istradamento condizionale)

Esistono due alternative per modellare una scelta deterministica tra B e C.

La prima, usando l’estensione del colore, prevede di aggiungere un requisito di abilitazione aggiuntivo basato sugli attributi del flusso di lavoro ad ogni tasks in modo che solo uno dei due sia abilitato se c2 è marcato.

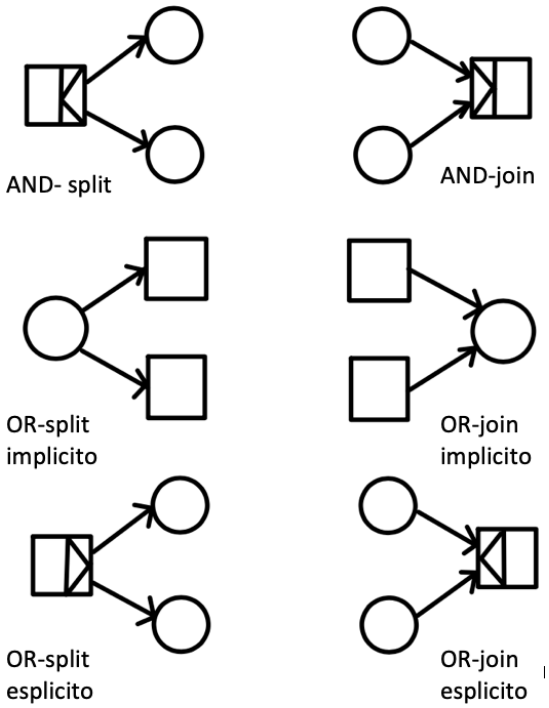
La seconda prevede invece l’uso di una transizione, l’OR-split esplicito, con più posti in uscita. Si tratta di una transizione che produce un gettone solo in una delle sue post-condizioni, che viene selezionata tenendo conto degli attributi del flusso di lavoro. In questo caso c’è una differenza rispetto al caso precedente, in cui la scelta veniva compiuta nel

momento in cui o B o C dovevano essere eseguiti, infatti qui la scelta viene effettuata nel momento in cui A viene completato.

Detto ciò possiamo capire che l'OR-split viene usato per modellare una scelta quando questa deve essere presa il più tardi possibile, invece si fa ricorso all'OR-split esplicito quando la scelta è compiuta sulla base degli attributi del flusso di lavoro. Si tratta di una distinzione fondamentale quando si parla di gestione del flusso di lavoro.

L'altro building block presente è l'OR-join il quale a sua volta può essere implicito e quindi rappresentato da un luogo o esplicito e quindi rappresentato da una transizione la quale viene abilitata se una delle sue pre-condizioni contiene gettoni.

La figura 29 qui sotto riporta la rappresentazione dei building blocks appena descritti.

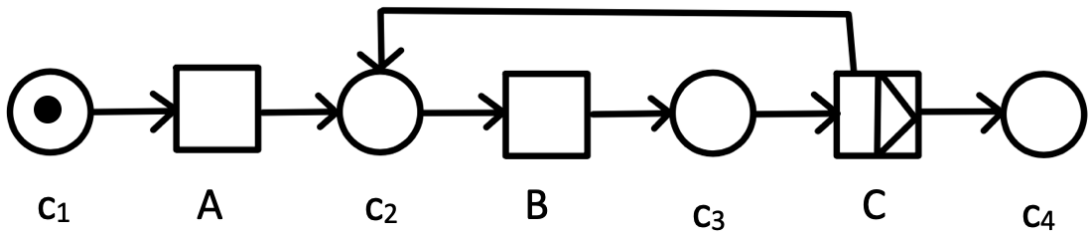


(figura 29, building blocks)

Infine, l'ultimo tipo di instradamento è costituito dall'iterazione, che consiste nella ripetizione di un determinato task. Proprio per questo motivo viene considerato un instradamento indesiderato dato che prevede l'esecuzione di un task per più volta e di fatto non permette l'avanzamento del flusso di lavoro. Nonostante questo, in determinate situazioni risulta essenziale per la buona riuscita del processo, ad esempio nel caso in cui

le informazioni fornite da un cliente sono incomplete e quindi il task va ripetuto finché queste non sono soddisfacenti, oppure si può pensare anche ad un controllo qualità che qualora dia esito negativo rende necessario ripetere i tasks precedenti.

Come possiamo notare dalla figura 30 qua sotto l'interazione può essere modellata con un OR-split esplicito. Il task C rappresenta un controllo che viene eseguito sul risultato del task B. L'esito del controllo stabilisce se l'attività B debba essere ripetuta o meno.



(figura 30, modellazione con Rete di Flusso di lavoro dell'iterazione)

3.5.2 L'esecuzione dei tasks

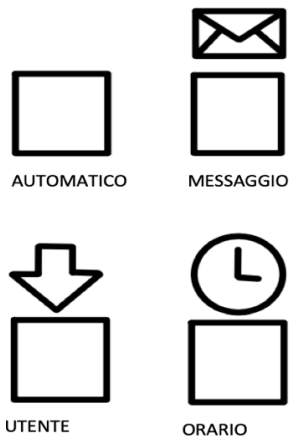
Come abbiamo già più volte specificato un flusso di lavoro definisce il percorso che un caso deve seguire e di conseguenza quali tasks dovranno essere svolti e in che ordine. Per un determinato caso che si trova in uno specifico stato viene indicato quali tasks possono essere performati ma questo non implica necessariamente che questi vengano eseguiti direttamente, nel senso che se un task deve essere compiuto da un dipendente questo deve essere disponibile (non impegnato in altre attività), oppure se è necessario un macchinario deve essere funzionante e non occupato. Capiamo come di fatto l'esecuzione dei vari tasks richieda delle ulteriori condizioni e che questa non può essere forzata dal sistema di gestione del flusso.

È doveroso quindi distinguere tra l'abilitazione di un task dalla sua esecuzione e questo viene fatto grazie ad un concetto fondamentale: il trigger, cioè una condizione esterna che

consente l'esecuzione di un task abilitato, quindi un task abilitato può essere eseguito/attivato sole se il caso è in uno stato che ne consente l'attivazione. In base a questo distinguiamo quattro tipi di task[29]:

- automatico, si tratta di un task che viene attivato nell'esatto momento in cui viene abilitato dato che viene eseguito da un'applicazione che non necessità di un intervento umano. Di fatto l'abilitazione coincide con l'esecuzione;
- utente, in questo caso il task abilitato richiede l'intervento di una risorsa umana per essere attivato. È quindi la risorsa che seleziona l'attività abilitata da eseguire dal suo "in-basket" che contiene i tasks che possono essere da essa performati effettivamente;
- messaggio, si tratta di un trigger che prevede un evento esterno per attivare e quindi eseguire un task abilitato. Si può pensare a delle telefonate, dei fax, dei messaggi o delle mail che permettono di "sbloccare" un task;
- orario, il task che risulta essere abilitato viene attivato da un orologio e quindi la sua esecuzione si verifica dopo un lasso di tempo predeterminato.

Queste quattro tipologie hanno una loro rappresentazione grafica nella modellazione tramite Rete di Petri, nella figura 31 qua sotto se ne fornisce un esempio.



(figura 31, rappresentazione grafica delle tipologie di trigger)

[29]Three Good Reason for Using a Petri-net-based Workflow Management System
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.33.960&rep=rep1&type=pdf

Dopo aver precisato la differenza tra abilitazione di un task e la sua attivazione possiamo anche tradurre quanto appena detto in termini di work-item ed activity. Infatti un task, inteso in senso generale, corrisponde ad una transizione, invece un work-item, ovvero la combinazione di un task e un determinato caso, essendo una parte di lavoro che può essere eseguita corrisponde ad una transizione abilitata per un caso specifico. È quindi la combinazione di un caso, un task e un trigger.

Infine, un activity rappresenta l'effettiva attivazione di una transizione, ovvero l'effettiva esecuzione di un work-item da parte di una risorsa. Può quindi essere inteso come la combinazione di un caso, un task, un trigger e una risorsa.

3.5.3 La proprietà di solidità

Tipicamente quando si parla di flussi di lavoro ci si concentra principalmente sull'analisi delle performance di quest'ultimo dato che l'obiettivo di un workflow è quello di completare i casi in maniera efficace ed efficiente. Di contro viene prestata una minor attenzione alla correttezza del processo che il flusso rappresenta e di riflesso questi falliscono non riuscendo a dare il giusto supporto. È quindi importante svolgere la verifica al fine di constatare l'assenza di errori logici all'interno del flusso, come ad esempio deadlocks o livelocks[30]. Bisogna poi sottolineare che a differenza della convalida, che dipende dal contesto e la quale è necessaria per attestare che il flusso si comporta esattamente come previsto, la verifica non dipende dal contesto poiché un processo che si blocca non è corretto in alcune situazione. Quindi gli errori logici possono essere individuati anche senza conoscere il funzionamento di un processo. Sulla base di ciò possiamo capire come per la

[30] Challenges in Business Process Management: Verification of business processes using Petri nets http://www.padsweb.rwth-aachen.de/wvdaalst/publications/p209.pdf

verifica non siano necessari concetti di correttezza dipendenti dal contento di applicazione, uno di questi è la proprietà di solidità.

Quando abbiamo definito una Rete di Flusso di Lavoro abbiamo elencato alcuni requisiti essenziali correlati alla sua struttura. Esiste però un altro importante requisito che deve essere soddisfatto: per ogni caso, il processo si concluderà e nel momento in cui questo risulta concluso c'è un gettone nel posto "o" e tutti gli altri posti risultano vuoti (questo in relazione ad ogni singolo caso, infatti una Rete di Flusso di Lavoro descrive il comportamento di un solo caso in isolamento) e non devono esserci transizioni (tasks) "morte", dovrebbe quindi essere possibile eseguire un task arbitrario seguendo un percorso appropriato all'interno della rete. Questo corrisponde alla proprietà di solidità.

Possiamo dare quindi una definizione formale, un processo modellato da una Rete di Flusso di Lavoro è solido se[31]:

- per ogni stato M raggiungibile a partire dallo stato "i" (ovvero lo stato con un solo gettone nel posto i) esiste una sequenza di scatti che dallo stato M permettere di raggiungere lo stato "o";
- lo stato "o" è l'unico stato raggiungibile dallo stato "i" con almeno un gettone nel posto "o";
- non ci sono transizioni "morte".

Dal primo requisito possiamo quindi capire che partendo dallo stato "i" è sempre possibile arrivare allo stato con un solo gettone sul posto "o", di fatto questo implica che alla fine si raggiunga lo stato finale del processo (in cui il caso risulta concluso). Il secondo invece ci dice che nell'istante in cui il posto "o" è marcato con un gettone tutti gli altri posti devono essere vuoti. Questi due requisiti insieme vengono definiti "terminazione corretta", ovvero che il caso ha completato il suo percorso all'interno della rete e non ci sono tasks da svolgere

[31] Verification of Workflow Nets
https://www.researchgate.net/publication/298226676_Verification_of_Workflow_nets

dato che l'unico gettone è presente nel posto "o". L'ultimo requisito afferma l'assenza di transizioni "morte" nello stato di inizio "i".

Dopo aver definito la proprietà di solidità è importante capire quando una Rete di Flusso di Lavoro risulta essere solida. Per fare ciò bisogna estendere la nostra Rete di Flusso di Lavoro aggiungendoci una transizione t* che collega il posto "o" e il posto "i". Per non fare confusione chiameremo WF la rete di base e WF-ex la rete estesa.

Detto questo, se la rete WF-ex è viva e limitata (con k=1) allora la rete WF è una rete solida (ricordiamo a tal proposito queste due proprietà. Una Rete di Petri si dice viva se per ogni stato raggiungibile M_1 e per ogni transizione t esiste uno stato M_2 che può essere raggiuto da M_1 in cui t è abilitata. Una Rete di Petri è limitata se per ogni posto p esiste un numero k tale che per ogni stato raggiungibile il numero di gettoni in p è minore o uguale a n. La Rete è sicura se il numero massimo di gettoni in un posto è k=1)

Vediamo quindi come questo può essere verificato. Se WF-ex è viva significa che per ogni possibile stato raggiungibile M esiste una sequenza di scatti che conduce ad uno stato in cui t* è abilitata. Dato che "o" è pre.condizione di t* possiamo capire che per ogni stato M raggiungibile da "i" è possibile raggiungere uno stato con almeno un gettone in "o". Se prendiamo in esame uno stato raggiungibile M_0+ "o", cioè uno stato con almeno un gettone in "o", la transizione t* è abilitata e attivandosi raggiunge lo stato M_0* "i". Inoltre, dato che WF-ex è limitata (con k=1), M_0 dovrebbe essere uguale allo stato vuoto.

Possiamo quindi dedurre che i requisiti uno e due della solidità sono rispettati e quindi la terminazione corretta è assicurata. L'ultimo requisito invece è soddisfatto dal momento che la rete WF-ex è viva. La Rete WF è quindi solida.

3.5.4 Esempio di una Rete di Flusso di Lavoro per la gestione degli ordini

Di seguito viene fornito un semplice esempio trovato in letteratura di come una Rete di Flusso di Lavoro può essere usata appunto per rappresentare un flusso di lavoro. A tal proposito ricordiamo in breve che un workflow consiste in un insieme di tasks organizzati allo scopo di eseguire processi aziendali.

In questo esempio verrà modellata una Rete per rappresentare un processo di un'azienda che prevede la gestione degli ordini fatti dai clienti[32]. Al suo interno daremo una descrizione del processo e dei vari tasks che lo compongono fornendo anche una veste grafica, per poi analizzarne il funzionamento attraverso la Rete di Flusso di Lavoro per capire come effettivamente si svolga.

Come già detto il processo in questione riguarda la gestione degli ordini, questi rappresentano quindi il "caso" che viene trattato. L'ordine una volta arrivato all'azienda per prima cosa viene registrato e successivamente due sono le "macro attività" che devono essere svolte:

- quella relativa all'evasione fisica dell'ordine;
- quella che prevede la sua fatturazione.

Possiamo già capire quindi come all'interno di questo flusso ci sia una parallelizzazione dei tasks da svolgere. Per quanto riguarda l'evasione dell'ordine dopo la registrazione si è a conoscenza di quali prodotti sono stati ordinati ed è quindi possibile verificarne la disponibilità a magazzino. Se questi sono disponibili allora possono essere preparati per la spedizione. Qualora invece non fossero disponibili due sono le possibilità: o viene fatto un ordine di rifornimento o si attente in quanto l'ordine per il riassortimento è già stato fatto. Quindi quando la merce sarà disponibile si potrà procedere alla sua spedizione.

[32] Making Work Flow: On the Application of Petri nets to Business Process Management https://link.springer.com/chapter/10.1007/3-540-48068-4_1

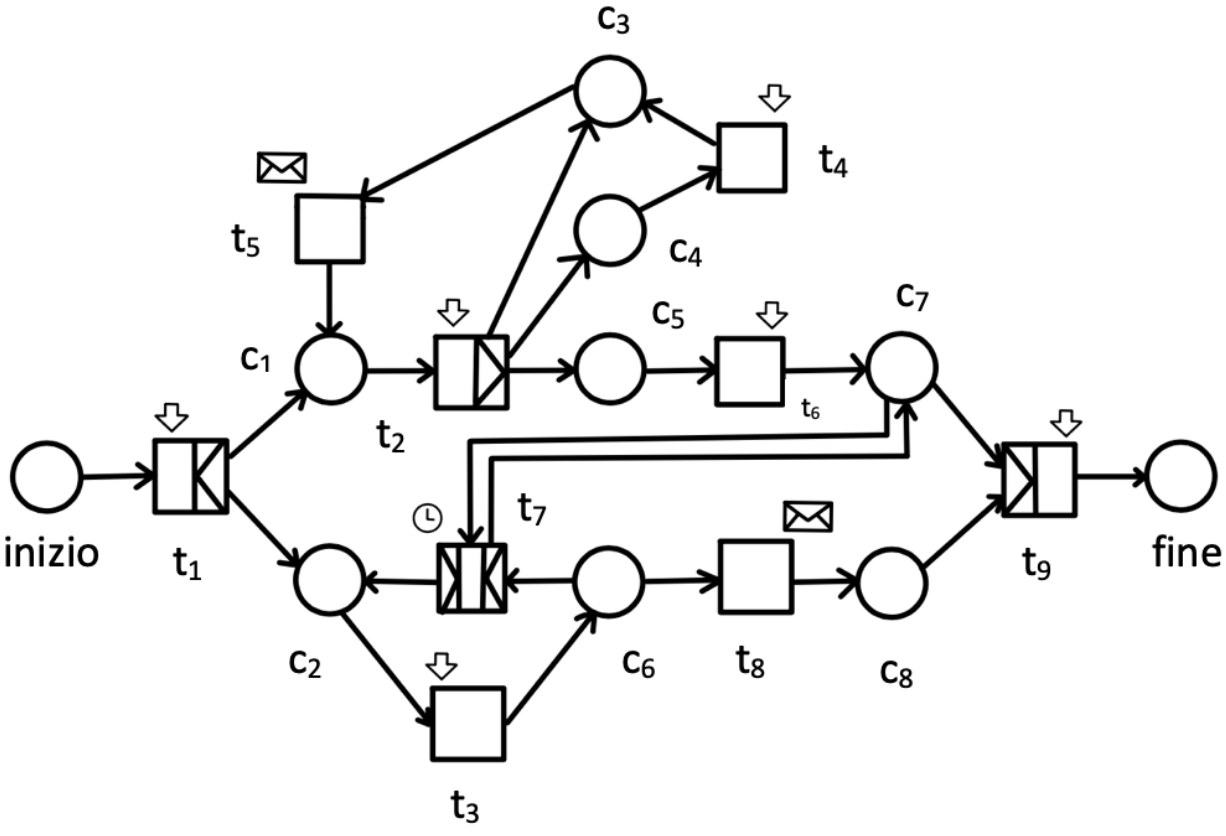
Invece la fatturazione come è ovvio prevede l'emissione della fattura e una volta fatto ciò è necessario ricevere il pagamento da parte del cliente. Nel caso in cui quest'ultimo tardi ad arrivare è possibile inviare un promemoria per sollecitarlo.

Infine, una volta che il pagamento è avvenuto e la merce è stata spedita l'ordine risulta ultimato e viene quindi archiviato, il caso risulta quindi concluso.

Il nostro flusso si compone quindi di nove tasks, qui elencati:

- registrare l'ordine;
- controllo della disponibilità;
- emissione della fattura;
- rifornimento;
- aggiornamento;
- spedire;
- promemoria;
- ricezione pagamento;
- archiviare.

Dopo averlo descritto a parole forniamo una rappresentazione grafica (figura 32) del processo per la gestione degli ordini in modo da rendere tutto più chiaro.



- inizio
-c_1 prodotti ordinati
-c_2 fatturazione
-c_3 prodotti riordinati
-c_4 prodotti da ordinare
-c_5 prodotti disponibili
-c_6 fattura emessa
-c_7 prodotti spediti
-c_8 pagamento ok
- fine

-t_1 registrare l'ordine
-t_2 controllo della disponibilità
-t_3 emissione della fattura
-t_4 rifornimento
-t_5 aggiornamento
-t_6 spedire
-t_7 promemoria
-t_8 ricezione pagamento
-t_9 archiviare

(figura 32, Rete di Flusso di Lavoro per la gestione degli ordini)

Come possiamo vedere dalla figura essendo una Rete di Flusso di lavoro è dotata di un posto di inizio e di uno di fine. Questi risultano fondamentali per capire quali casi devono ancora essere gestiti e quali invece sono già stati completati.

Per quanto riguarda i tasks possiamo fare alcune considerazioni:

- i tasks "registrare l'ordine", "emissione della fattura", "controllo della disponibilità", "rifornimento", "spedire" e "archiviazione" necessitano di un trigger per la loro esecuzione, in particolare non essendo task automatici necessitano l'intervento di una risorsa umana. Per esempio, nel controllo di disponibilità è necessario un dipendente che controlli nel database se quel prodotto è disponibile oppure nell'emissione della fattura è necessario un impiegato che se ne occupi;

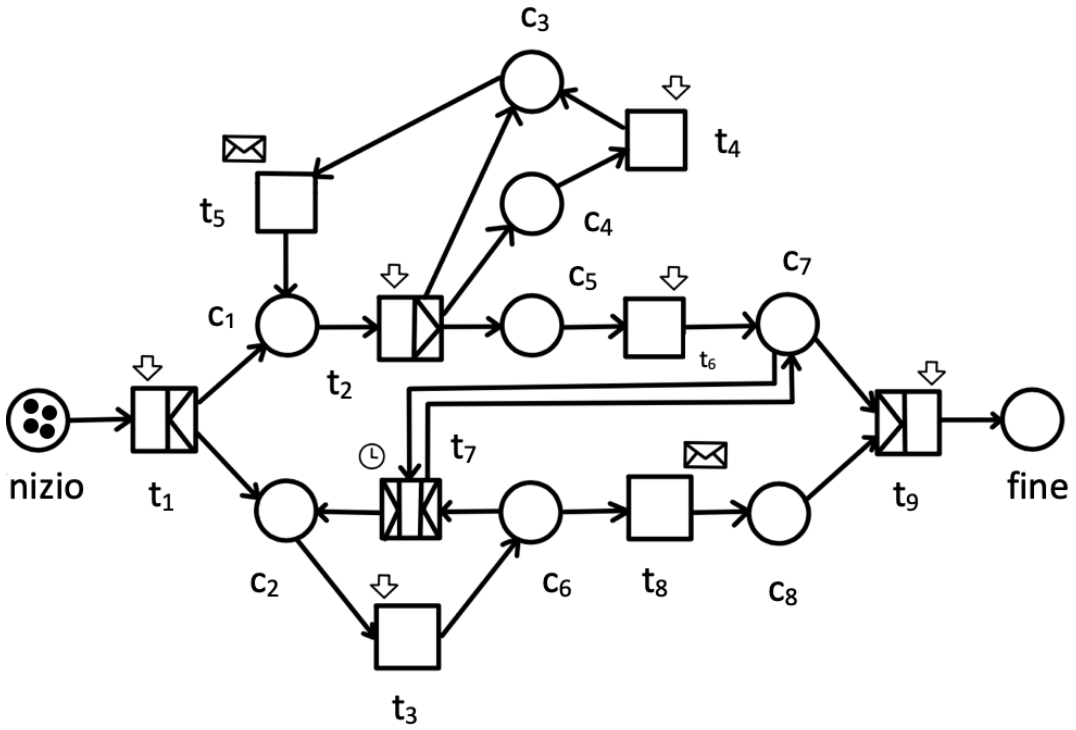
- i tasks "aggiornamento" e "ricezione pagamento" vengono invece attivati da un evento esterno. Ad esempio, arriva una notifica del pagamento del cliente oppure arriva al magazzino l'ordine di rifornimento contenente i prodotti mancanti;

- il task “promemoria” invece è associato ad un trigger di tempo, nel senso che un promemoria può essere mandato solo dopo che un determinato periodo di tempo è trascorso senza ricevere alcun pagamento.

Possiamo inoltre osservare dalla struttura della Rete che:

- il task “registrare ordine” rappresenta un AND-slip che va quindi ad abilitare l’esecuzione di due task paralleli, ovvero “controllo di disponibilità” ed “emissione fattura”;
- “controllo disponibilità” consiste in un OR-split esplicito che produce un gettone solo per una delle sue post-condizioni e quindi questo implica la scelta tra quali dei tasks successivi andrà eseguito;
- il task “archiviare” è di tipo AND-join e il suo compito è quello di sincronizzare le due parti del flusso di lavoro, quella che prevede l’evasione dell’ordine e quella che si occupa della fatturazione, per poter completare il caso;
- il promemoria invece rappresenta un AND-join/AND-split. Questo task risulta bloccato fino all’invio della fattura e alla spedizione delle merce, quindi quando viene abilitato sincronizza i due sotto flussi e permette il rinvio della fattura;
- possiamo notare anche la presenza di un OR-split implicito nel posto c_6 “fattura emessa. È interessante notare che in questo caso la scelta non viene effettuata dal processo in se ma dall’ambiente in cui esso si trova, nel senso che non si può sapere se inviare il promemoria o ricevere il pagamento sarà il task successivo.

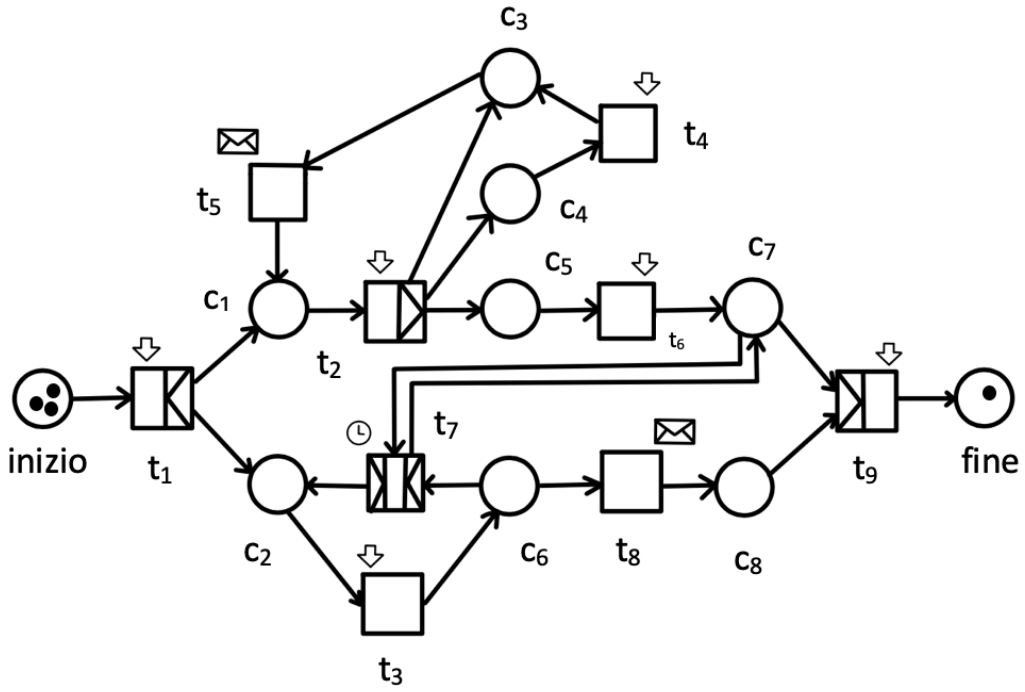
Vediamo ora il funzionamento di questo flusso attraverso la Rete di Flusso di Lavoro. Come possiamo vedere dalla figura 33 nello stato iniziale vediamo la presenza di quattro casi (gettoni)nel posto “inizio” pronti per essere gestiti.



(figura 33, funzionamento Rete di Flusso di Lavoro per la gestione degli ordini)

Il primo ordine, ovvero il primo caso, può quindi essere registrato dato che il relativo task, “registrare l’ordine”, è abilitato. Essendo questo modellato attraverso un AND-join con il suo scatto (che indica che l’ordine è stato registrato) vengono prodotti gettoni in c_1 e in c_2 che abilitano quindi lo svolgimento di due task in parallelo, “controllo della disponibilità” e “emissione della fattura”. Supponiamo che venga eseguito per primo il task che prevede il controllo della disponibilità. Il controllo da esito positivo, ovvero i prodotti ordinati sono disponibili, viene quindi prodotto un gettone in c_5 che a sua volta abilita il task “spedire” che verrà eseguito e quindi i prodotti saranno spediti (gettone in c_7). Nel frattempo, potrà essere stato eseguito il task “emissione della fattura”, quindi la fattura verrà emessa. A questo punto, essendoci un gettone in c_6, il task “ricezione pagamento” sarà abilitato ma potrà essere eseguito solo dopo che sarà arrivata la notifica dell’avvenuto pagamento (trigger). Quindi quando questo si verificherà verrà prodotto un gettone in c_8. Vediamo quindi come ora entrambe le pre-condizioni del task “archiviare” siano marcate (essendo questo un AND-join è necessario che entrambe siano marcate) e dato che risulta abilitato questo può quindi

essere eseguito. Con l'archiviazione del caso (l'ordine) quest'ultimo ha completato il suo percorso all'interno del flusso, rappresentato dalla presenza di un gettone in "fine" (figura 34).



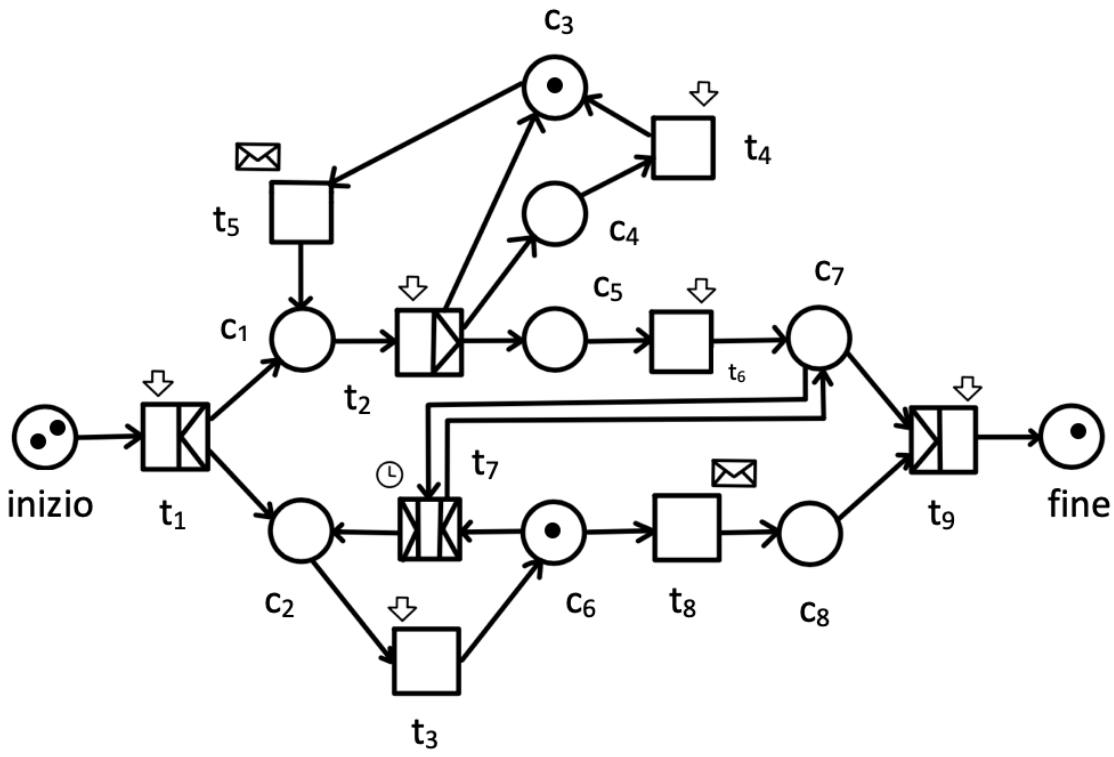
(figura 34, funzionamento Rete di Flusso di Lavoro per la gestione degli ordini)

Questo rappresenta il migliore dei casi possibili nel senso che le merci ordinate erano disponibili e che la fattura è stata pagata subito senza necessità di emettere un promemoria. Proponiamo ora un esempio di funzionamento un po' più complicato.

La situazione di partenza ora vede tre gettoni nel posto "inizio" e uno in "fine" (si riparte dalla situazione finale precedente in figura 34). Come già detto prima il task "registrare ordine" risulta abilitato, con il suo scatto viene quindi eseguito e come risultato vengono prodotti gettoni in c_1 e c_2. Ora risultano abilitati in parallelo i tasks che prevedono il controllo della disponibilità della merce ordinata e quello che porta all'emissione della fattura. Viene quindi eseguito il "controllo di disponibilità" che essendo un OR-split produrrà un gettone o in c_3 o in c4 o in c5. Questa volta però l'esito del controllo è negativo, ovvero i prodotti non sono disponibili. Dal controllo può risultare che la merce è già stata ordinata e quindi verrà prodotto un gettone in c_3 oppure che i prodotti dovranno essere ordinati e quindi c_4 risulterà

marcato. Supponiamo si verifichi quest'ultima ipotesi, questa abilita il task "rifornimento" che potrà scattare e produrre quindi un ordine per il rifornimento, quindi il posto c_3 conterrà un gettone in quanto i prodotti sono stati ordinati. Questo abilita il task "aggiornamento", il quale però essendo associato ad un trigger esterno potrà essere eseguito solo quando arriverà un messaggio che conferma la disponibilità della merce riordinata.

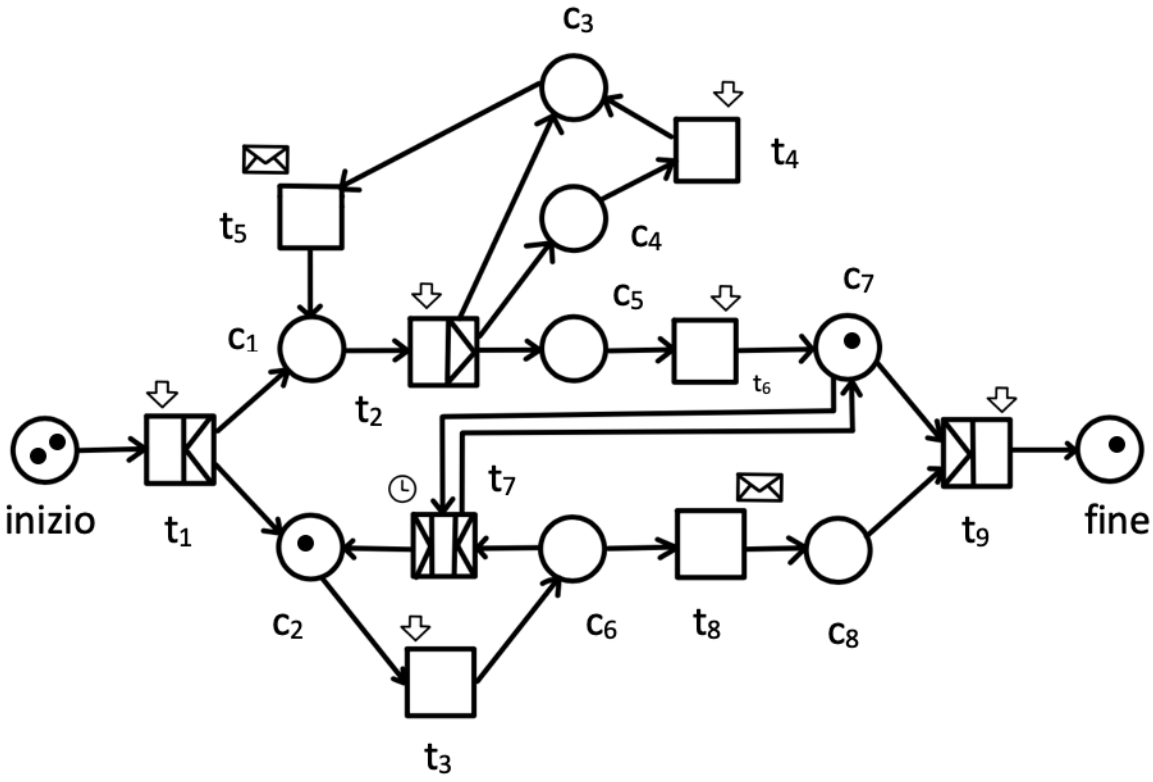
Nel frattempo, dato che anche il task "emissione della fattura" era abilitato potrà scattare e con questo verrà emessa la fattura, a questo corrisponde la comparsa di un gettone in c_6. Come conseguenza di ciò solo il task "ricezione pagamento" risulta abilitato ma non potrà essere eseguito fino a quando il trigger che prevede la notifica della ricezione del pagamento sarà soddisfatto (figura 35).



(figura 35, funzionamento Rete di Flusso di Lavoro per la gestione degli ordini)

Durante ciò se sarà arrivato il messaggio dal magazzino che informa che la merce ordinata è disponibile potrà essere eseguito il task "aggiornamento". Questo a sua volta renderà abilitato il task "controllo disponibilità" che ora darà esito positivo producendo un gettone in c_5, il quale abilità il task "spedire" e di conseguenza i prodotti verranno spediti, c_7 risulterà

quindi marcato dato che ora i prodotti sono stati spediti. Durante tutto ciò se la notifica del pagamento non sarà ancora arrivata e se sarà trascorso un certo periodo di tempo (ad esempio tre giorni) il task “promemoria” diventerà abilitato, dato che ora entrambe le sue pre-condizioni sono marcate (c_6 e c_7). Si tratta di AND-join/AND-split quindi per scattare sottrae i gettoni da entrambi i suoi posti di input, invece con il suo scatto produce un gettone nei posti c_7 (da cui era stato precedentemente sottratto, ricompare immediatamente poiché i prodotti erano già stati spediti) e in c_2 per far si che la fattura possa essere riemessa dato che il task “emettere fattura” è abilitato (figura36).



(figura 36, funzionamento Rete di Flusso di Lavoro per la gestione degli ordini)

La fattura viene quindi emessa, segnalata dalla presenza del gettone in c_6. Supponiamo che dopo il rinvio della fattura arrivi la notifica della ricezione del pagamento così che il task “ricezione pagamento” possa ora essere eseguito. Ora la rete di trova nello stato in cui i prodotti risultano spediti e il pagamento è stato ricevuto, ovvero c_6 ec$_8$ sono marcati. Questi due abilitano il task “archiviare” e una volta che viene eseguito il caso è arrivato alla fine del

suo percorso all'interno del flusso di lavoro a cui corrisponde la presenza di un gettone in "fine".

Come possiamo notare dall'esempio tutti i concetti relativi alle Reti di Flusso di Lavoro finora esposti sono presenti. Il flusso di lavoro modellato grazie a questo tipo di Rete soddisfa infatti i requisiti per essere considerato una Rete di Flusso, ovvero la presenza di un posto di inizio e di fine. Al suo interno inoltre troviamo le varie tipologie di costrutti di instradamento e si vede anche come l'esecuzione dei vari tasks sia dipendente dalla presenza dei vari trigger. Per ultima, ma non meno importante, possiamo constatare che si tratta di una Rete solida, infatti come possiamo capire dagli esempi di funzionamento è sempre stato possibile raggiungere lo stato "fine" e quando questo si verificava nelle Rete tutti gli altri posti non contenevano gettoni. Inoltre, non sono presenti transizioni "morte" in quanto tutte contribuiscono al completamento dei casi che vengono gestiti dal flusso di lavoro.

Grazie a questo esempio siamo quindi riusciti a rappresentare in maniera semplice un flusso di lavoro, a descriverne il funzionamento, così da capire quale percorso possono seguire i casi all'interno del flusso, e verificarne la correttezza grazie alla proprietà di solidità. Questo ci permette di apprezzare, attraverso un semplice lavoro, la potenza espressiva e di modellazione di questa tipologia di Reti.

CONCLUSIONI

Questo lavoro ha affrontato l'argomento delle Reti di Petri, partendo dalla descrizione delle Reti classiche, la reti P/T, l'analisi è stata approfondita presentando le estensioni di cui gode questo strumento, cioè quella del colore, del tempo e della gerarchia, per poi arrivare in fine ad introdurre la ragione per cui esistono i flussi di lavoro e le loro caratteristiche in modo da poter discutere dell'applicazione di queste Reti all'interno della gestione dei processi aziendali.

L'obiettivo era quello di riuscire a dimostrare l'utilità e validità di questo strumento nel rappresentare con grande chiarezza e fedeltà il funzionamento di sistemi reali. Come è stato mostrato nel Capitolo I, in cui sono state fornite semplici definizioni circa la struttura, le regole e il funzionamento di queste Reti, è stato possibile modellare e descrivere in maniera completa il funzionamento di un sistema reale, ovvero il distributore automatico. Allo stesso modo con l'introduzione delle Reti di Petri di Alto Livello, ottenute estendendo le reti classiche, è stato possibile modellare e analizzare sistemi più complessi, anche a livello industriale. Come si è dimostrato nell'esempio del capitolo II anche questa volta è risultato semplice mappare, attraverso l'estensione del colore, un processo di stampa nel quale grazie proprio alle informazioni e alle condizioni logiche aggiuntive previste della relativa estensione è stato possibile descrivere il funzionamento di quest'ultimo in maniera ancora più completa e realistica permettendo ancora una volta di apprezzare la natura grafica e la semantica formale che consentono descrizioni molto precise e prive di ambiguità.

Quindi l'estensione del colore consente di specificare e calare ancor di più nella realtà il funzionamento di un sistema/processo. Inoltre, le estensioni del tempo e della gerarchia sono quantomeno essenziali al fine di descrivere gli aspetti temporali, dato che le varie fasi di cui si compone un processo richiedono tempo per essere svolte, ed evidenziare la

struttura a livelli (processo principale e sotto processi di cui si compone) che un reale processo presenta. È quindi evidente che senza il ricorso a queste estensioni risulterebbe difficile plasmare e rappresentare alcuni importanti aspetti di molti sistemi reali.

Nell'ultimo capitolo poi si è affrontato il tema delle Reti di Flusso di Lavoro e di come il loro utilizzo per la modellazione di un flusso di lavoro sia di grande aiuto.

La situazione corrente dal punto di vista dei sistemi di gestione dei flussi di lavoro, nonostante i grandi sforzi della Workflow Management Coalition, vede la presenza di una moltitudine di questi sistemi ma ciò che ancora manca è uno standard concettuale. Come conseguenza di questa mancanza molte organizzazioni non sono inclini ad utilizzare questi tipi di software. Ecco perché, come è stato esposto nel corso di questo lavoro, la teoria delle Reti di Petri rappresenta una solida base da cui partire per procedere nella standardizzazione di un modello dato che le caratteristiche di queste Reti trovano una facile applicazione e si adattano con successo nel contesto dei flussi di lavoro.

In particolare, la rappresentazione e modellazione dei flussi è possibile grazie alle Reti di Flusso di Lavoro che dispongono infatti di un linguaggio grafico che permette la modellazione delle "primitive" di un workflow e inoltre godono di una semantica definita in maniera formale, permettendo di rappresentare la logica aziendale sottostante alla definizione di un processo.

Come è stato possibile vedere nel processo per la gestione degli ordini attraverso queste Reti è possibile dare vita ad un modello di flusso di lavoro che rispecchia tutte le caratteristiche che i flussi hanno. Grazie a queste reti nell'esempio si è potuto costruire un modello fedele ad un reale processo con tutti i tasks e condizioni di cui si compone, capendo se fossero necessarie risorse per eseguirli o se fossero automatici o se la loro esecuzione dipendesse da condizioni esterne, riuscendo quindi a gestire in maniera appropriata i trigger e le scelte esterne, e grazie alle varie primitive (sequenza, parallelizzazione, scelta e reiterazione) si è riusciti a definire in modo realistico il percorso che un caso può seguire

all'interno del flusso. Possiamo quindi ben capire che le caratteristiche di un flusso basato sulle Reti di Petri sono di grande aiuto nella fase di progettazione e analisi.

Inoltre, la grande capacità di rappresentazione di questo strumento è dovuta anche dal fatto che risulta basato sugli stati e non sugli eventi, infatti lo stato di un caso che sta seguendo un percorso all'interno del flusso può essere modellato in maniera esplicita dai Posti. Altre tecniche invece, come i diagrammi di flusso, risultano essere basate sugli eventi, nel senso che le Transizioni vengono plasmate esplicitamente mentre gli stati sono impliciti e quindi soppressi. In questo modo però viene persa dell'informazione, quindi la distinzione tra queste due tecniche è cruciale. Risulta quindi particolarmente importante sottolineare che una rappresentazione basata sugli stati permette di differenziare l'abilitazione di un task dalla sua esecuzione, come sappiamo infatti l'abilitazione non implica che il task sia eseguito immediatamente (come descritto nel paragrafo 3.5.2).

La loro validità circa l'applicazione nei flussi di lavoro è dimostrata anche dalla disponibilità di tecniche di analisi. In particolare, è stata discussa una proprietà di base che dovrebbe essere soddisfatta da ogni flusso di lavoro, ovvero la proprietà di solidità, la quale garantisce flussi corretti e assicura la buona riuscita dei casi trattati dal processo. Proprietà che nelle Reti di Flusso di Lavoro corrisponde alla vivacità e limitatezza.

Quindi questo consente di dimostrare come lo strumento delle Reti di Petri può dare il suo contributo, grazie alla sua base teorica formalizzata, alla sua facilità di rappresentazione e alla sua capacità di analisi, nello sviluppo dei prossimi sistemi di gestione dei flussi di lavoro e nella definizione dei processi/flussi di lavoro.

BIBLIOGRAFIA

A. Venkateswarlu, Abhishek Halder, A study of Petri Nets: Modeling, Analysis and Simulation, 2006

C. Mohan, G Alonso, R. Gunthor, M Kamath, Exotica: A Research Perspective on Workflow Management Systems, Data Engineering Bullettin, Volume 18, 1995

Gehlot Vijay, From Petri Nets to Coloured Petri Nets: A tutorial Introduction to Nets based formalism for modeling and simulation, Proceedings of the Winter Simulation Conference, 2019

Kurt Jensen, Lars M. Kristensen, Coloured Petri Nets: Modelling and Validation of Concurrent Systems, Speinger-Verlag Berling Heidelberg, 2009

Kurt Jensen, Wil M. P. Aalst, Gianfranco Balbo, Maciej Koutny, Karsten Wolw, Transactions on Petri Nets and Other Models of Concurrency VII, Lecture Notes in Computer Science, Springer-Verlag Berlin Heidelberg, 2013

Luca Ferrarini, Automazione industriale: Controllo Logico con Reti di Petri, Pitagora Editrice Bologna, 2001

Louchka Popova-Zeugmann, Time and petri Nets, Springer-Verlag Berlin Heidelberg, 2013

Manuel Silva, Half a century after Carl Adam Petri's Ph.D thesis: A perspective on the field, Annual Reviews in Control, Volume 37, Issue 2; 2013, pages 191-219

Shi meilin, Yang Guangxin, Wu Shangguang, Workflow management systems: a survey, 1998 International Conference on Communication Technology, Proceedings IEEE, volume 2, 1998

Seung-yun Kim, Waleed W. Smari, A Petri Net-based Workflow Modeling for a Human-Centric Collaborative Commerce System, 2006

T. Murata, Petri nets: Properties, analysis and applications, in Proceedings of the IEEE, vol. 77; no. 4, pp. 541-580, April 1989

Tuomas Aura, Johan Lilius, A casual semantics for time Petri nets, Theoretical Computer Science, Volume 243, Issues 1-2, pages 409-447, 2000

W.M.P van der Aalst, Challenges in Business Process Management: Verification of Business Processing Using Petri Nets; Bulletin of the EATCS, volume 80, 2003

W.M.P. van der Aalst, Making Work Flow: On the Application of Petri Nets to Business Process Management. Application and Theory of Petri Nets 200. Lecture Notes in Computer Science, volume 2360. Springer, Berlin, Heidelberg, 2002

W.M.P. van der Aalst, Putting high-level Petri nets to work in industry, Computers in Industry, Volume 25, Issue 1, pages 45-54, 1994

W.M.P van der Aalst, The Application of Petri Nets to Workflow Management, The Journal of Circuits, Systems and Computer. Volume 8, pages 21-88, 1998

W.M.P van der Aalst, Three Good Reasons for Using a Petri-Net-Based Workflow Management System, Information and Process Integration in Enterprises, the Springer International series in Engineering and Computer Science, volume 428, 1998

W.M.P van der Aalst, Verification of workflow nets, in: Application and Theory of petri Nets. Lectures Note in Computer Science, vol. 1248, Springer, Berlin, Heidelberg, 1997

W.M.P. van der Aalst, Kees M. van Hee, Workflow Management: Models, Methods and Systems, Cooperative information systems, 2002

Wolfgang Reisig, Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies, Springer-Verlag Berlin Heidelberg, 2013

Workflow management Coalition, Terminology and Glossary, Document Number WFMC-TC-1011, 1999