La Bibbia di Ingegneria del Software

Mario Petruccelli
Università degli studi di Milano

A.A. 2018/2019

Sommario

1 Introduzione

Per superare i metodi di produzione artigianale si studiano tecniche e metodologie che possano migliorare ed assicurare software di qualità. Per far ciò si utilizza un approccio diverso da quello scientifico, detto **approccio ingegneristico** che è decisamente più pragmatico: dato un **target** si definisce una **metrica** secondo la quale si misura la bontà del prodotto rispetto al target. Se il risultato è soddisfacente si tende a continuare nello sviluppo mentre se non lo è si ricomincia daccapo la fase appena conclusasi. L'obiettivo è quello di trovare una modalità di esecuzione del lavoro che faccia lavorare bene gli sviluppatori:

- Cambiando il numero e la mansione di tutti gli addetti al progetto. Il programmatore non è più cliente e questo crea problemi di comunicazione.
- Decidendo la dimensione del software.
- Software che per natura è malleabile ed è interessato da continue modifiche tanto da dare luogo a tantissime evoluzioni e versioni differenti.

Il software **deve funzionare** e **deve essere aderente alle specifiche del progetto**. Può essere che vengano richieste funzioni malevole, sbagliate ed incomplete, perchè non è sempre chiaro cosa viene richiesto e le aspettative del cliente spesso non coincidono con le specifiche del progetto. Lo sviluppatore, accortosi del danno, potrebbe sì correggere gli errori andando però così fuori specifiche. Raccogliere i requisiti del progetto è la prima e più importante cosa da fare. **Il software corretto è un software di cui ci si può fidare.** È difficile che un software sia completamente affidabile, tant'è che spesso si fa firmare un documento che scancisce l'inaffidabilità a prescindere del prodotto. L'affidabilità è difficile da misurare anche rispetto le aspettative che, tuttavia, non si possono fissare sulla carta. *Un software affidabile non è detto che sia anche corretto.* **Il software non dev'essere dannoso**; dev'essere garantita la sicurezza dell'utente e dev'essere robusto garantendo una certa performance anche sotto sforzo.

Dicendo che il software dev'essere bello s'intende che per il cliente dev'essere facile da usare. Per misurare l'usabilità si prende in esame un campione di utenti e si chiede loro di fare alcune cose servendosi del software osservando così le loro reazioni ad ogni richiesta. Attraverso lo studio del comportamento dell'utente (*dato dal suo sguardo, da ciò che pensa e ciò che prova mentre utilizza il software*) è possibile misurare il grado di usabilità del prodotto così da poterla qualificare. **Il software dev'essere reattivo** nell'interazione con l'utente ed efficiente nello sfruttamento delle risorse del sistema. **Il software dev'essere pulito** in modo tale che risulti facile metterci mano per rivederlo e modificarlo. La semplicità degli interventi post-consegna aiuta, quindi dovrà essere mantenibile perchè verranno sicuramente richiesti dei cambiamenti o delle aggiunte le quali avvengono più tranquillamente se il software è stato già predisposto alla cosa in fase di progettazione. Il 60% degli interventi post-consegna sono di manutenzione. Spesso il tutto comporta un'estensione dei requisiti di partenza così il software evolve per adattarsi a nuove situazioni che spesso esulano dal controllo del programmatore.

I processi che influiscono sulla qualità di un prodotto riguardano la resistenza agli imprevisti, specialmente a livello implementativo e decisionale. I processi che influiscono sulla qualità di un prodotto riguardano la resistenza agli imprevisti, specialmente a livello implementativo e decisionale.

Vaporwave L'annuncio di un prodotto che viene fatto per occupare una fetta di mercato ma che non vedrà mai luce per problemi tecnici o economici.

2 Produzione software

2.1 Modellare il ciclo di vita di un software

3 Reti di Petri

3.1 Definizioni

Le reti di Petri servono come linguaggio di specifica formale per sistemi concorrenti. Formalmente una rete di Petri è una quintupla $[P; T; F; W; M_o]$

- P è l'insieme dei posti.
- T è l'insieme delle transizioni.
- F è la relazione di flusso. $F \subseteq (P \times T) \cup (T \times P)$
- W è la funzione che associa un peso ad ogni flusso. $W : F \to \mathbb{N}^+$
- M è la marcatura iniziale. $M_0 : P \to \mathbb{N}$

Preset $Pre(a) = \{d \in (P \cup T) | < d, a > \in F\}$

Postset $Post(a) = \{d \in (P \cup T) | < a, d > \in F\}$

Una transizione t abilitata in M se e solo se $\forall p \in Pre(t)\ M(p) \geq W(< p, t >)$
Sintassi: $M[t >$

Lo scatto di una transizione t in una marcatura M produce una marcatura M'

- $\forall p \in Pre(t) - Post(t) \qquad M'(p) = M(p) - W(< p, t >)$
- $\forall p \in Post(t) - Pre(t) \qquad M'(p) = M(P) + W(< t, p >)$
- $\forall p \in Post(t) \cap Pre(t) \qquad M'(p) = M(p) - W(< p, t >) + W(< t, p >)$
- $\forall p \in P - (Pre(t) \cup Post(t)) \qquad M'(p) = M(p)$

3.2 Relazioni

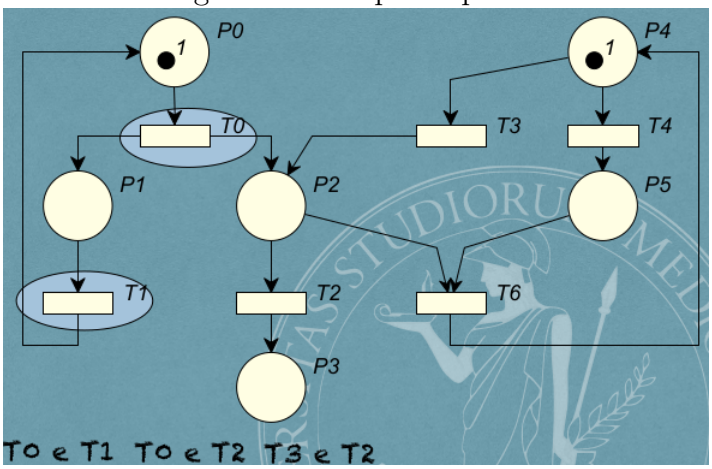
3.2.1 Sequenza

Una transizione t_1 è in sequenza con una transizione t_2 in una marcatura M se e solo se

$$M[t_1 > \wedge \neg M[t_2 > \wedge M[t_1 t_2 >$$

- t_1 è abilitata in M.
- t_2 **non** è abilitata in M.
- t_2 viene abilitata dallo scatto di t_1 in M.

Figure 1: Esempio sequenza



3.2.2 Conflitto

Due transizioni (t_1, t_2) sono in conflitto

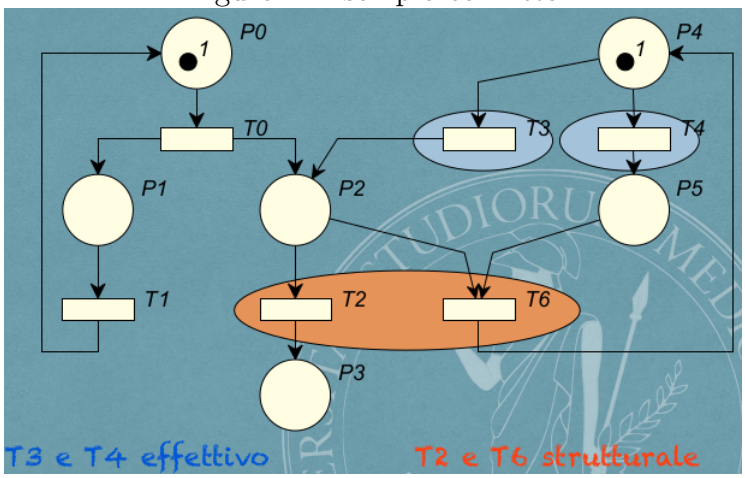
- **Strutturale** se e solo se $Pre(t_1) \cap Pre(t_2) \neq \oslash$
- **Effettivo in una marcatura M** se e solo se $M[t_1 > \wedge M[t_2 > \wedge$

$$\exists p \in Pre(t_1) \cap Pre(t_2) | M(p) < W(< p, t_1 >) + W(< p, t_2 >)$$

  - t_1 e t_2 sono abilitate in M.
  - Esiste un posto in ingresso ad entrambe che non ha abbastanza token per far scattare entrambe.

Il conflitto strutturale è condizione necessaria per quello effettivo.

Figure 2: Esempio conflitto



3.2.3 Concorrenza

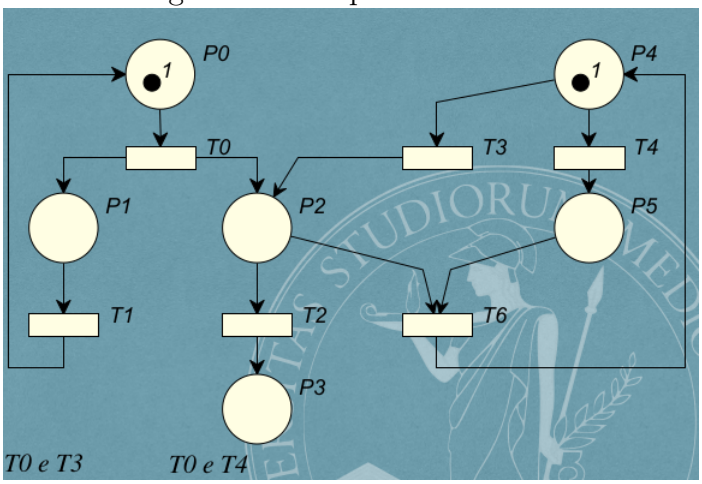
Due transizioni (t_1, t_2) sono in concorrenza

- **Strutturale** se e solo se $Pre(t_1) \cap Pre(t_2) = \oslash$
- **Effettivo in una marcatura M** se e solo se $M[t_1 > \wedge M[t_2 > \wedge$

$$\forall p \in Pre(t_1) \cap Pre(t_2) | M(p) \geq W(< p, t_1 >) + W(< p, t_2 >)$$

  - t_1 e t_2 sono abilitate in M.
  - Tutti i posti in ingresso ad entrambe hanno abbastanza token per far scattare entrambe.

Figure 3: Esempio concorrenza



3.3 Insieme di raggiungibilità

L'insieme di raggiungibilità di una rete a partire da una marcatura M è il più piccolo insieme di marcature tale che:

- $M \in R(P/T, M)$

  M è raggiungibile da M.

- $M' \in R(P/T, M) \wedge \exists t \in T \quad M'[t > M'') \rightarrow M'' \in R(P/T, M)$

  Se M' appartiene all'insieme di raggiungibilità a partire da M ed esiste una transizione $\in T$ tale per cui in M' con lo scatto di t arrivo in M'', allora anche M'' appartiente all'insieme di raggiungibilità a partire da M.

3.4 Proprietà di limitatezza

Una rete P/T con marcatura M si dice limitata se e solo se

$$\exists k \in \mathbb{N} \quad \forall M' \in R(P/T, M) \quad \forall p \in P \quad M'(p) \leq k$$

Esiste un $k \in \mathbb{N}$ tale per cui per ogni marcatura M' raggiungibile a partire da M, in questa marcatura M' per ogni posto $\in P$ il numero di gettoni è $\leq k$.
Cioè se è possibile fissare un limite al numero di gettoni della rete.

Se la rete è limitata:

- L'insieme di raggiungibilità è finito.
- È possibile definire un automa a stati finiti corrispondente i cui stati sono le possibili marcature dell'insieme di raggiungibilità *(può essere utile per sfruttare i tool presenti per le FSM)*.

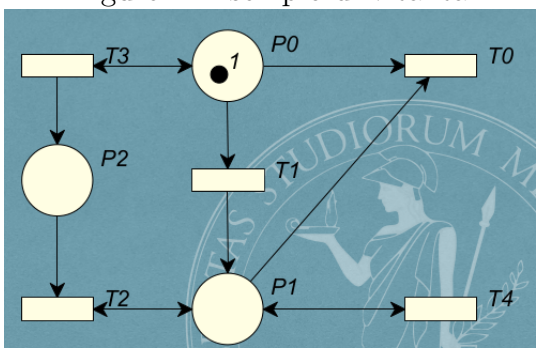
3.5 Vitalità di una transizione

Una **transizione** t in una marcatura M è viva a

- **Grado 0** (o morta) se non è abilitata in nessuna marcatura appartenente all'insieme di raggiungibilità.
- **Grado 1** se esiste almeno una marcatura raggiungibile in cui è abilitata.
- **Grado 2** se per ogni numero n esiste almeno una sequenza ammissibile di scatti in cui la transizione scatta n volte.
- **Grado 3** se esiste una sequenza di scatti ammissibile in cui scatta infinite volte.
- **Grado 4** se in qualunque marcatura raggiungibile, esiste una sequenza ammissibile in cui scatta *(è viva)*.

Una rete è viva se **tutte** le sue transizioni sono vive di **grado 4**.

Figure 4: Esempio di vitalità



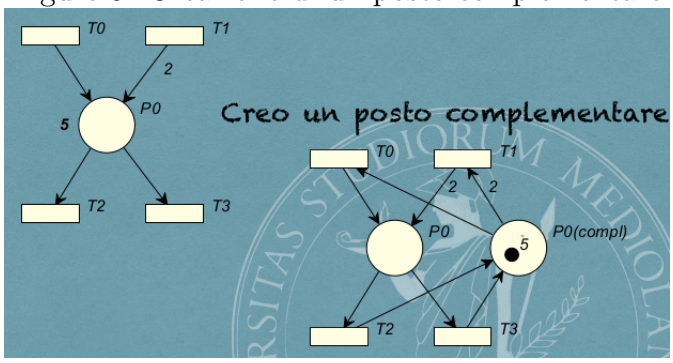
- $T0$ ha grado 0 perchè non può mai scattare.
- $T1$ ha grado 1 perchè può scattare solo una volta.
- $T2$ ha grado 2 perchè può scattare un numero di volte n grande a piacere: si fa scattare $T3$ n volte, poi si fa scattare $T1$ e a quel punto $T2$ può scattare n volte.
- $T3$ ha grado 3 perchè nella marcatura iniziale può scattare infinite volte. Notare che se scatta $T1$, $T3$ diventa di grado 0.
- $T4$ ha grado 4 perchè in qualunque marcatura raggiungibile, posso arrivare ad una marcatura che mi permetta di farla scattare.

3.6 Estensioni e riduzioni per le reti di Petri

3.6.1 Capacità dei posti

Una possibile estensione delle reti di Petri consiste nel fissare un massimo numero di token ammissibili in un posto. In questo modo si può **forzare** la limitatezza. Questa estensione non aggiunge nulla di nuovo alle nostre reti in quanto si può ottenere lo stesso risultato con la creazione di posti complementari. Un posto complementare è un posto che ha per ogni arco in ingresso del posto principale un arco con lo stesso peso in uscita che va nella stessa transizione e viceversa. La capacità del posto sarà la **somma** tra i token presenti nel posto principale e nel posto complementare.
NB La somma rimarrà sempre costante e il numero di stati raggiungibili sarà sempre lo stesso.

Figure 5: Creazione di un posto complementare



Definizione formale Un posto p_c è complementare di p **solo in una rete pura** se e solo se

$$\forall t \in T \quad \exists < p,t >\in F \leftrightarrow \exists < t,pc >\in F \quad | \quad W(< p,t >) = W(< t,pc,>)$$

$$\forall t \in T \quad \exists < t,p >\in F \leftrightarrow \exists < pc,t >\in F \quad | \quad W(< pc,t >) = W(< t,p,>)$$

In caso di reti con capacità sui posti la definizione di abilitazione sarebbe:
$t \in T$ è abilitata in M se e solo se

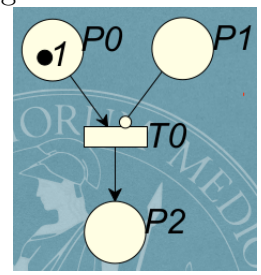
- $\forall p \in Pre(t) \quad M(p) \geq W(< p, t >)$
- $\forall p \in Post(t) \quad M(p) + W(< t, p >) < C(p)$
- $\forall p \in Post(t) - Pre(t) \quad M(p) + W(< t, p >) < C(p)$
- $\forall p \in Post(t) \cap Pre(t) \quad M(p) - W(p, t) + W(< t, p >) < C(p)$

3.6.2 Archi inibitori

Permettono di dire che non deve essere presente neanche un token perchè la transizione sia abilitata. Se il posto a cui è collegato l'arco inibitore è **limitato** si può rimappare lo stesso risultato su una rete normale:
Se in p non ci saranno mai più di n gettoni, si può costruire il posto complementare p_c mettendo $M(p) + M(p_c) = n$ e l'arco inibitore mappato su un arco d'ingresso e uscita p_c con peso $= n$. In questo modo se ci sono n gettoni in p_c, vuol dire che non ce ne può essere neanche uno in p.
In caso di rete con **posto non limitato** ne aumenta la potenza poichè non si riesce creare una rete equivalente.

Non sono considerate da tutti come estensione elegante poichè non si possono usare certe tecniche di analisi.

Figure 6: Arco inibitore



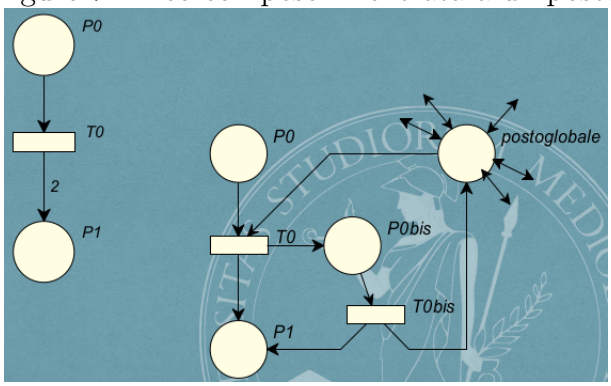
3.6.3 Eliminazione pesi archi

Perchè utilizziamo il peso sugli archi? Sarebbe possibile creare delle reti di Petri che non utilizzano il peso sugli archi?

Esempio Abbiamo un arco $(T0, P1)$ con peso 2, potremmo creare un posto e una transizione *bis* in uscita dalla transizione $T0$ che creerà in $P1$ un secondo gettone. I due gettoni non vengono generati atomicamente e prima di $T0_{bis}$ potrebbe scattare un'altra transizione; in questo modo si potrebbero avere marcature diverse dalla rete di partenza. Per mantenere le stesse marcature si dovrebbe creare un ulteriore posto

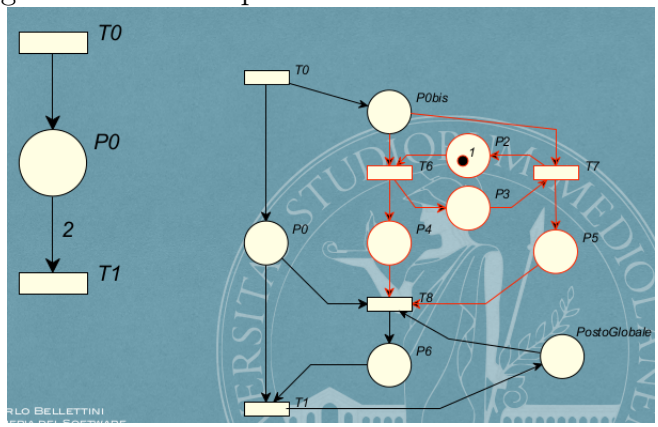
globale che è in ingresso a tutte le transizioni della rete originaria tranne che a $T0_{bis}$; in questo modo per andare avanti si dovrà per forza abilitare $T0_{bis}$. Ovviamente è un **pessimo** utilizzo delle reti di Petri.

Figure 7: Arco con peso in entrata a un posto



Quanto visto sopra non basta a risolvere il problema, cosa succede se un arco con peso è in ingresso a una **transizione**? La situazione è analoga, bisogna creare una parte di rete che aumenta di complessità con l'aumentare del peso. In questo caso non basta aggiungere un posto globale, posti e transizione *bis*. Bisogna costruire una parte di rete in cui si contano i gettoni che vengono messi nei posti che avrebbero arco con peso in uscita alla transizione per evitare che venga abilitata anche con un singolo gettone.

Figure 8: Arco con peso in entrata a una transizione



In sintesi: si possono fare delle reti senza peso analoghe a quelle con peso, ma la complessità aumenta parecchio.

3.6.4 Reti condizioni eventi

Una rete viene detta C/E se tutti gli archi hanno peso 1 e tutti i posti hanno capacità 1. Se i posti (condizioni) in ingresso a t contengono un token, la transizione t (evento) può scattare. Una rete P/T **limitata** ha sempre una corrispondente nella classe C/E.

3.6.5 Rete conservativa

Data una funzione di pesi H

$$H : P \to \mathbb{N}^+$$

Una rete P/T con marcatura M si dice conservativa rispetto a tale funzione se e solo se

$$\forall M' \in R(P/T, M)$$

$$\sum_{p \in P} H(p)M'(p) = \sum_{p \in P} H(p)M(p)$$

La sommatoria pesata (con la funzione H) del numeri di gettoni nei vari posti da sempre lo stesso valore per qualunque marcatura raggiungibile.

Se una rete è conservativa allora è anche limitata. Una rete limitata non è per forza conservativa.

$$conservatività \to limitatezza$$

Una rete P/T conservativa rispetto alla funzione che assegna pesi tutti uguali a 1 si dice **strettamente conservativa**. Il numero di gettoni in una rete strettamente conservativa non varia mai.

Verificare che una rete sia conservativa può essere un problema non banale, capire se è strettamente conservativa è molto più facile.

Se il numero di token consumati dallo scatto di una transizione **non morta** è uguale al numero di token creati, allora la rete è strettamente conservativa.