UNIVERSITÀ DI PISA

P49: Dog Training

Business Processes Modeling
Project Report

Author: *Bombino Biancamaria 561745, Mastrorilli Alessandro 657939*

Professor:	*Roberto Bruni*
Master's degree:	Data Science & Business Informatics
Date:	26 novembre 2023

Indice

Dog Training Process

1. Introduzione

Il presente documento si pone come obiettivo la modellazione del processo *Dog Training*, in cui bisogna gestire la richiesta di eventuali clienti che richiedono dei corsi di addestramento per i loro animali. I principali attori che sono stati identificati sono:

- **Il Cliente**
- **Il Centro di Addestramento**

Come richiesto dalle linee guida, inizialmente è stata utilizzata la notazione grafica **BPMN** per descrivere la dinamica dell'intero processo, iniziando con la rappresentazione singola di ogni pool. Nella seconda parte sono state analizzate le **Petri Nets** ottenute, in modo da effettuare un'analisi qualitativa delle reti. I diagrammi BPMN sono stati ideati utilizzando **BPMN.io**, mentre le Petri Nets sono state create usando **Woped**. L'analisi di queste ultime è stata eseguita sia con **Woped** sia con **Woflan**. Il diagramma BPMN è stato descritto usando due pools: il primo per il cliente e il secondo per il centro di addestramento. In quest'ultimo pool è stato applicato il metodo delle *swimlanes*, in modo da suddividere il pool in due *lanes*: la segreteria del centro di addestramento e l'addestratore. La comunicazione tra i due attori principali del processo avviene tramite le *message flow arrows*. Inoltre, nel pool del centro di addestramento è stato associato un *DataStore* all'attività di pagamento, in modo da salvare in modo permanente i pagamenti effettuati dal cliente. In questo modo le informazioni riteneute più importanti per il centro vengono salvate oltre la durata dell'istanza del processo. I processi rappresentati nei due pool sono strettamente dipendenti tra di loro; quindi, per sottolineare questa dipendenza reciproca e per controllarne il flusso, sono stati utilizzati i *Event-based Gateway*. Tramite questi ultimi è stato possibile evidenziare come il flusso del primo attore dipende dalle decisioni del secondo e viceversa. Infine sono stati utilizzati anche gli *Exclusive Gateway*, in particolare *XOR split* e *XOR join*. Nelle Sezioni successive vengono descritte in maniera dettagliata le varie fasi di implementazione e analisi del processo, con i relativi risultati.

2. Diagramma BPMN

2.1 Cliente

Il pool del cliente, raffigurato in Figura 1, è composto da un business process model in cui egli, attraverso l'utilizzo di un **message flow**, contatta il centro di addestramento fornendo le caratteristiche del proprio cane. Dopo aver ricevuto una proposta di appuntamento da parte dell'addestratore associato dalla segreteria del centro, il cliente ha due opzioni definite tramite un **XOR split**:

- accettare la proposta ricevuta
- formulare la controproposta

Nell'ultimo caso si prosegue con un **Event-based Gateway** basato sulle decisioni esterne dell'addestratore, in cui le possibilità definite attraverso i **catching intermediate event** corrispondo alla ricezione di conferma o all'indisponibilità da parte dell'addestratore. In caso di rifiuto della controproposta è necessario ripetere il ciclo attraverso lo **XOR join**, tramite il quale si torna alla fase in cui il cliente riceve una proposta con la data e il luogo della lezione. Invece, in caso di conferma della proposta iniziale o della controproposta è possibile procedere con un **XOR join**

che conduce alla ricezione degli esercizi proposti dall'insegnante. Successivamente il cliente riceve gli esercizi che deve eseguire insieme al proprio cane, e dopo il task di esecuzione l'addestratore viene avvisato dall'allievo della conclusione dei compiti assegnati. Il giudizio dell'istruttore viene espresso tramite un **event-based gateway** con due possibilità:

- esecuzione corretta degli esercizi
- esecuzione scorretta degli esercizi

In caso di esecuzione errata il processo viene ripetuto mediante un **XOR join**, in caso contrario è previsto un ulteriore **event-based gateway**. Quest'ultimo consente all'addestratore di proporre nuovi esercizi oppure di concludere l'addestramento odierno. In particolare, se si verifica il primo caso il ciclo viene ripetuto con l'impiego di uno **XOR join** che permette di riniziare il ciclo di esecuzione e valutazione del nuovo esercizio.

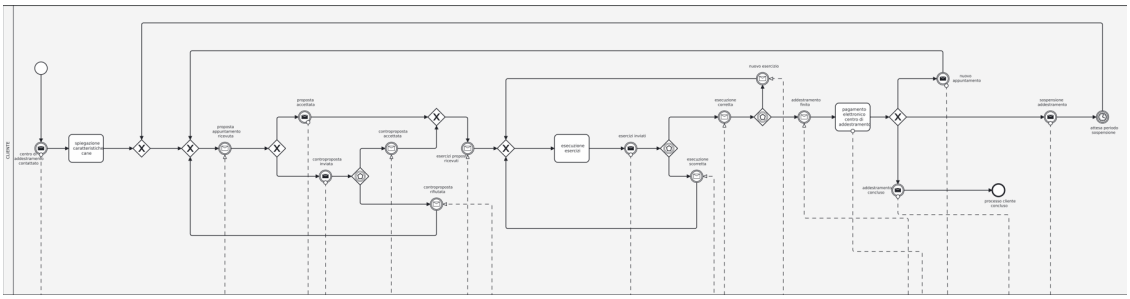

Figura 1: Diagramma BPMN del Cliente

Al termine dell'incontro attuale, il cliente può procedere con il task del pagamento elettronico, e attraverso uno **XOR split** sono rappresentate le differenti richieste dell'allievo:

- richiesta di un nuovo appuntamento
- conclusione dell'addestramento
- sospensione temporanea dell'addestramento

Per quanto riguarda la sospensione, l'evento è rappresentanto tramite un **intermediate time event** che indica la durata del periodo di inattività. Allo scadere del timer, il processo si ripete partendo dalla richiesta di una nuova prenotazione e la pianificazione dell'appuntamento. Mentre se l'allievo desidera un nuovo appuntamento, il ciclo rinizia dalla fase di programmazione della lezione (la data e il luogo). Infine se il cliente decide di concludere definitivamente gli incontri, notifica la sua scelta alla segreteria del centro. In questo ultimo caso l'intero processo viene terminato.

2.2 Centro di Addestramento

Il secondo pool di questo processo è il **Centro di Addestramento**, visualizzabile in Figura 2. Quest'ultimo viene attivato quando il cliente decide di contattare la segreteria del centro per scambiare informazioni riguardo un eventuale addestramento. Per rappresentare questo attore abbiamo usato le **swimlanes**, e per tale motivo il pool è stato diviso in due **lanes**:

- segreteria del centro di addestramento
- addestratore

Questa suddivisione è stata implementata poiché la segreteria si occupa della gestione delle informazioni del cliente e dei relativi pagamenti, oltre alle comunicazioni riguardo il termine effettivo di ogni addestramento. Invece l'addestratore, una volta contattato dalla segreteria del centro, gestisce l'organizzazione degli appuntamenti e degli esercizi da eseguire e correggere durante ogni incontro. Nella prima *lanes* è possibile osservare le attività della segreteria amministrativa del centro. Questa fase del processo inizia con la ricezione del contatto del cliente, rappresentata da un

catching start event che conduce a contattare l'addestratore adatto basandosi sulla spiegazione delle caratteristiche del cane. Il contatto dell'addestratore è rappresentato come un **throwing intermediate event**. Successivamente si passa direttamente al pagamento elettronico ricevuto, con il relativo salvataggio all'interno del *Datastore*. A questo evento segue con un **Event-based Gateway**, a cui sono associati degli eventi di catching. In particolare il gateway presenta tre output, di cui solo uno rientra tra le responsabilità della segreteria, ovvero: la notifica della conclusione definitiva dell'addestramento. Dopo questo *catching message* il processo del centro di addestramento viene terminato.

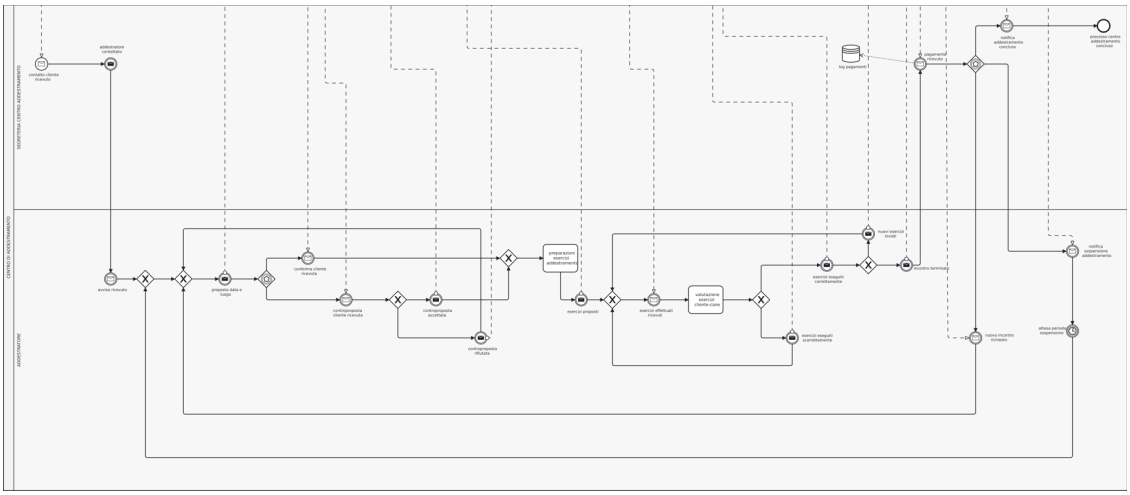

Figura 2: Diagramma BPMN del Centro di Addestramento

Nella seconda *lanes* invece, sono raffigurate le attività e gli eventi dell'addestratore contattato in base alle specifiche dell'animale. Questa parte del processo inizia con un **catching intermediate event** che segnala la ricezione dell'avviso da parte della segreteria. Successivamente, sono presenti due **XOR join**: il primo per gestire il caso della sospensione dell'addestramento e il secondo per gestire la controproposta rifiutata e la richiesta di un nuovo incontro. Una volta stabilita la data e del luogo dell'appuntamento con un **throwing intermediate event**, l'addestratore attende le due possibili risposte del cliente:

- conferma della proposta
- offerta di una controproposta

Queste due possibilità sono instradrate tramite un **event-based gateway**, poiché quest'ultimo è adoperato proprio per modellare decisioni che dipendono da alcuni eventi esterni. Nel caso in cui l'evento esterno corrisponda alla conferma dell'appuntamento, si prosegue direttamente con la fase della lezione che ha inizio tramite il task di preparazione degli esercizi. Se invece l'evento corrisponde ad una controproposta ricevuta, allora sono presenti due possibilità mediante un **XOR split**:

- controproposta accettata
- controproposta rifiutata

In caso di accettazione, si raggiunge un *XOR join* e si entra nella fase dell'incontro come in precedenza. In caso di rifiuto, si ripete il procedimento della predisposizione dell'appuntamento attraverso un *XOR join* che permette di ripetere la sequenza. Nella parte successiva sono poi esposte le attività effettuate durante gli incontri giornalieri. Questa fase incomincia con la preparazione degli esercizi a cui segue un evento di throwing con cui vengono proposti gli esercizi al cliente. Successivamente è presente l'evento di ricezione degli esercizi ultimati dal cliente con il suo animale e l'addestratore valuta le prove. La valutazione viene rappresentata con un task, ovvero un'attività atomica. In prosieguo, è presente un **XOR split** con due possibilità:

- throwing event messagge con notifica di esercizi corretti

- throwing event message con notifica di esercizi scorretti

In quest'ultimo caso gli esercizi vengono ripetuti ricollegando il flusso all'**XOR join** antecedente. Nel caso di corretta esecuzione degli esercizi, sono invece disponibili due strade diverse come output di un **XOR split**:

- nuovi esercizi inviati
- incontro terminato

Nel primo caso, si ripete il ciclo di ricezione e valutazione degli esercizi, nel secondo invece la lezione viene terminata e si prosegue con il pagamento. A seguire è presente l'**event-based gateway**, descritto anche nella prima *lanes*, in funzione della decisione presa dal cliente che può:

- richiedere un nuovo incontro, facendo tornare il flusso alla sequenza di proposta data e luogo.
- sospendere l'addestramento, in cui si attende il periodo di sospensione (delineato da un **intermediate time event**) e poi il flusso viene condotto nuovamente all'inizio della fase di organizzazione dell'appuntamento.
- terminare l'addestramento. il cui evento è gestito dalla segreteria.

Quest'ultima casistica guida il processo considerato alla sua conclusione.

2.3 Diagramma BPMN completo

In questa Sezione è possibile visualizzare il diagramma BPMN dell'intero processo *Dog Training* mediante la Figura 3.

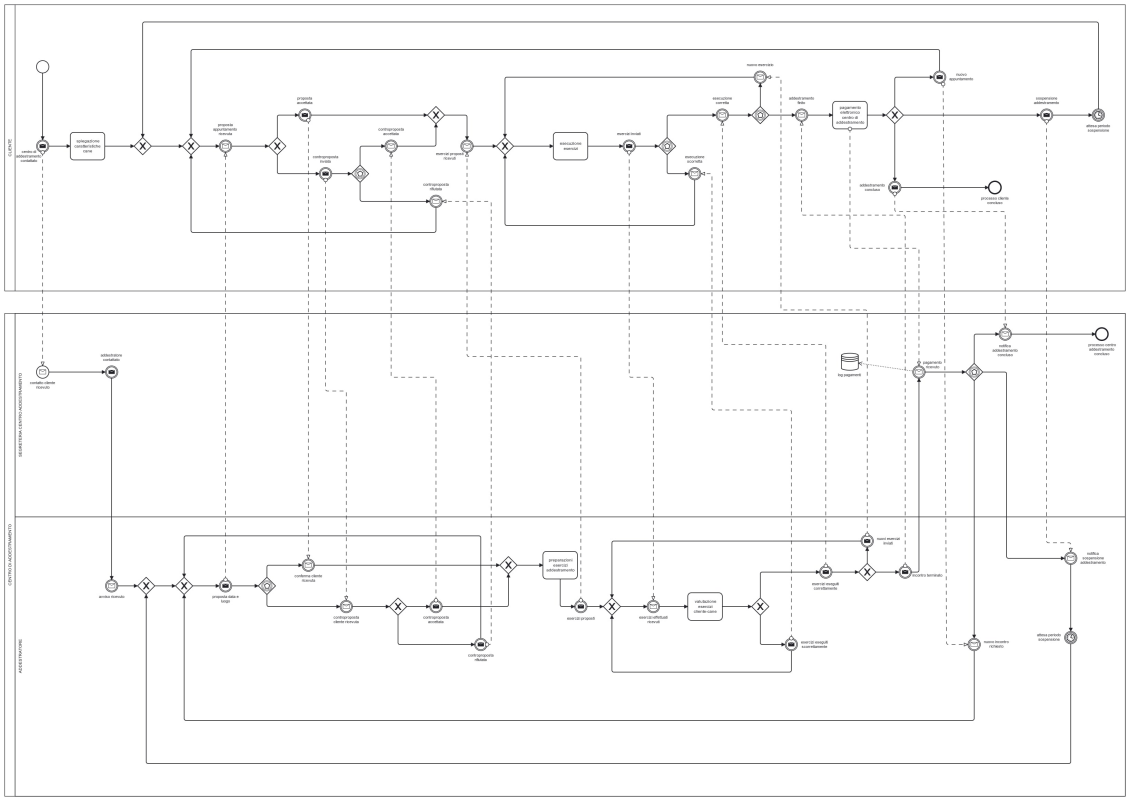

Figura 3: Diagramma BPMN completo

3. Petri Nets

In questa Sezione sono delineati i passaggi della trasformazione del diagramma BPMN, discusso nella Sezione 2, in una Workflow Net. Questa procedura è stata eseguita in modo tale da verificare le proprietà strutturali del processo ed effettuare un'analisi qualitativa di esso. La trasformazione è stata attuata come segue:

- è stato aggiunto un *place* per ogni collegamento
- gli eventi, le attività e gli exclusive gateway sono stati rappresentati come una *transition*
- è stata prestata particolare attenzione all'*event-based gateway*, per cui è stata impiegata un'unica piazza di input, la quale ha nel postset un numero di transizioni pari al numero di alternative possibili.
- in conclusione è stato eseguito il *desugar*

Nelle Sezioni successive sono riportate le Petri Net del Cliente, del Centro di Addestramento e del Workflow Net Complessivo.

3.1 Cliente

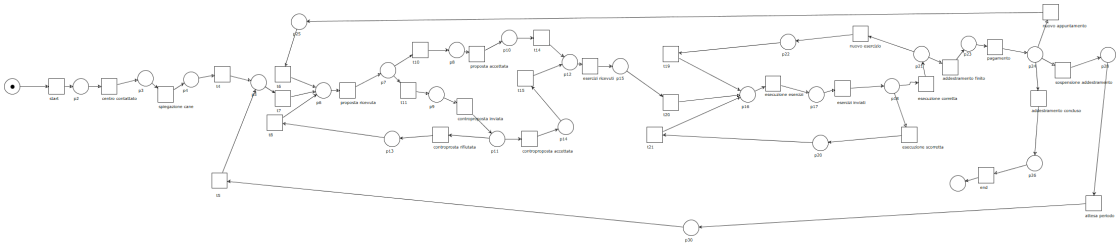

Figura 4: Workflow Net del Client

In Figura 4 è ritratto il Workflow Net del Cliente. Questa rete è composta da **28 places**, **33 transitions** e **66 arcs**. Dall'analisi emerge che è effettivamentee una Workflow Net, poiché è composta da:

- un unico place iniziale che non presenta archi in ingresso
- un solo place finale con nessun arco in uscita
- ogni place e transition appartengono ad almeno un cammino dal place iniziale a quello finale

In Figura 6 e 7, sono invece riportate le analisi ottenute adoperando sia Woped sia Woflan. Da queste immagini è possibile notare come la rete è **S-Coverable** poiché presenta un unico *S-Component* che copre tutta la rete ed è **strongly-connected** perché tutti i nodi hanno archi in entrata e quindi possono essere raggiunti. La rete non è una *T-Net* perché i place possono contenere più di una input transition e più di una output transition. Avendo un unico *S-Component* ed essendo i preset e postset delle transizioni composti da un unico place, la rete è una **S-Net**. Di conseguenza è **safe** e **sound**. Essendo una S-Net, la rete è **free-choice**, in quanto ogni coppia di transizioni ha preset equivalenti o disgiunti. Conseguentemente è anche **live** e **bounded**. Non essendo presenti PT-Handles e TP-Handles, la rete è **well-structured**. Infine, le proprietà della *soundness* sono tutte rispettate e quindi non sono presenti deadlocks poiché ogni nodo del grafo ha un arco uscente, escludendo ovviamente quello relativo alla fine del processo. Il **Coverability Graph** del Cliente, visualizzabile in Figura 5, è finito ed è composto da 33 archi e 28 vertici.

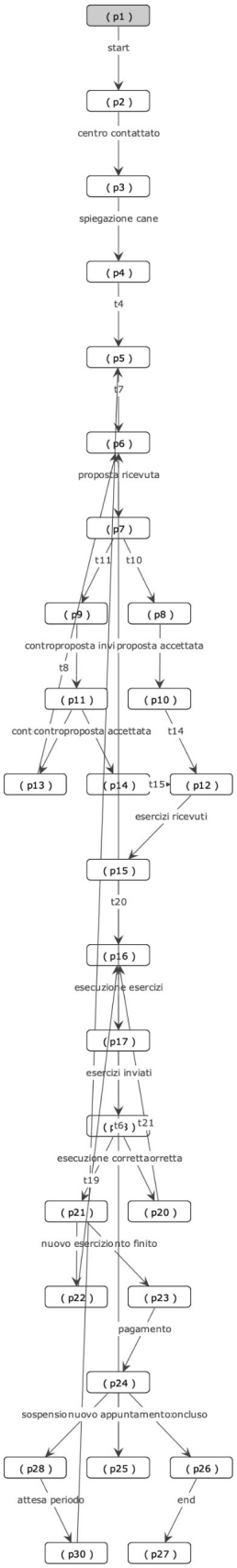

Figura 5: Coverability Graph del Cliente

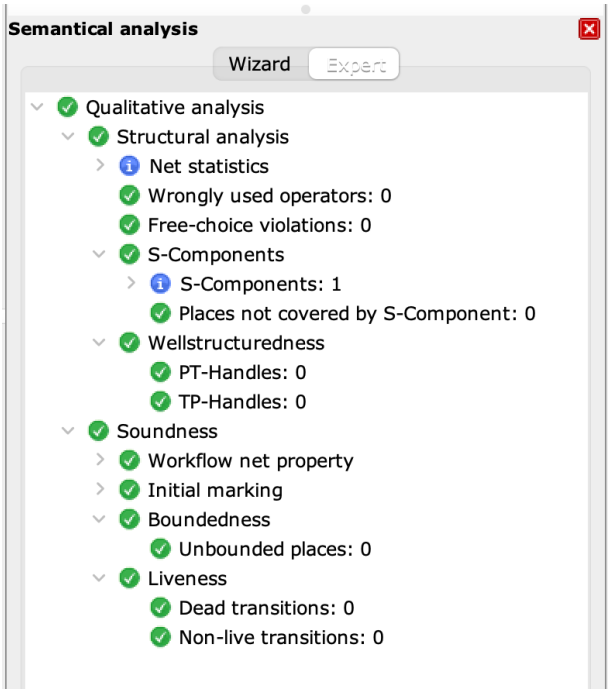

Figura 6: Analisi semantica del cliente con WOPED

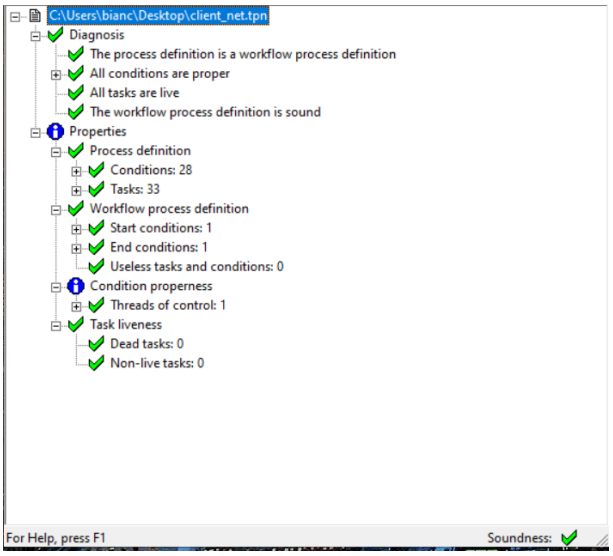

Figura 7: Analisi semantica del cliente con WOFLAN

3.2 Centro di Addestramento

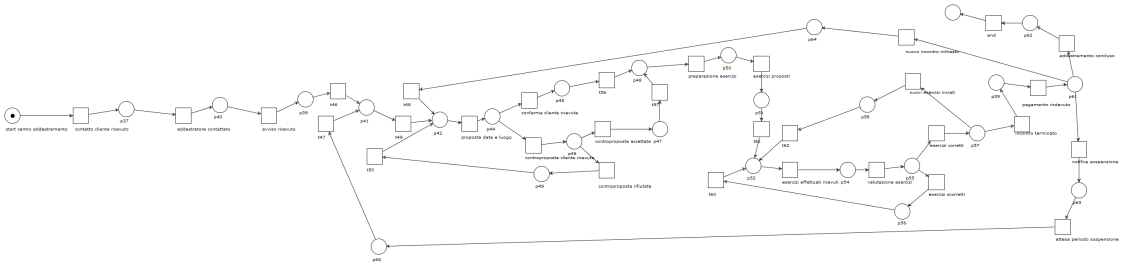

Figura 8: Workflow Net del Centro di Addestramento

Nell'immagine 8 è illustrata la Workflow Net del Centro di Addestramento. La rete è composta da **27 places**, **32 transitions** e **64 arcs**. Le analisi conseguite sul workflow net del centro di addestramento sono le stesse del cliente e i risultati, riportati nelle Figure 10 e 11, sono anche equivalenti. Infatti, la rete rispetta le proprietà di *Liveness*, *Boundness* e conseguentemente di **Soundness**. Inoltre è **well-structured** poichè non sono presenti PT-Handles e TP-Handles, ed è anche **free-choice**. Dagli esiti nelle immagini sottostanti è possibile notare che anche questa rete è **S-Coverable** poiché presenta un unico *S-Component* che copre tutta la rete ed è **strongly-connected**. Infine la rete non è T-Net, ma è una **S-Net**. Il **Coverability Graph** del centro di addestramento è visibile nella Figura 9. Quest'ultimo è composto da 32 archi e 27 vertici. Anche in questo caso il grafo risulta finito, dato che la rete è bounded.
Sia per il cliente sia per il centro è possibile verificare anche tramite i coverability graph che non esistono *dead task*, infatti sussiste sempre almeno un arco etichettato con una transizione delle rispettive reti. Inoltre sia la **option to complete** che la **proper completion** sono soddisfatte. Una volta che un token raggiunge i token finali non rimangono token in nessun place della rete.

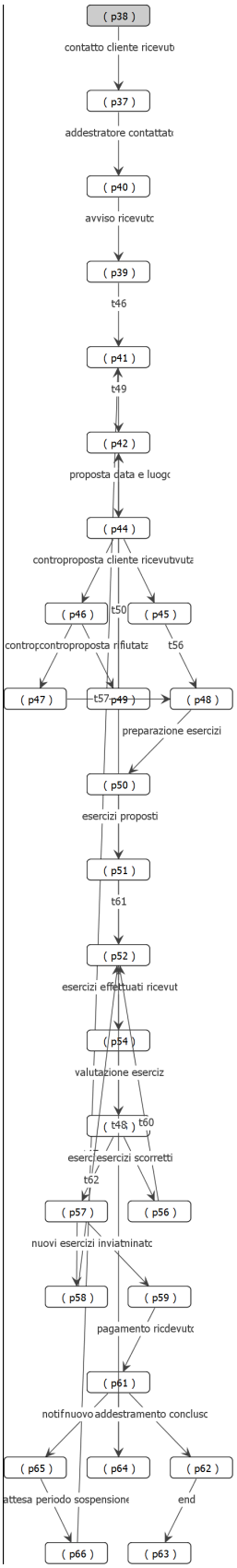

Figura 9: Coverability Graph del Centro di Addestramento

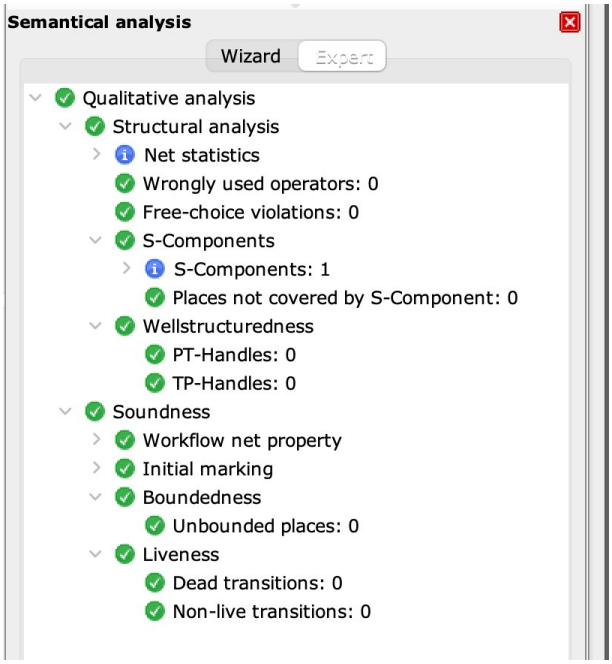

Figura 10: Analisi semantica del centro di addestramento con WOPED

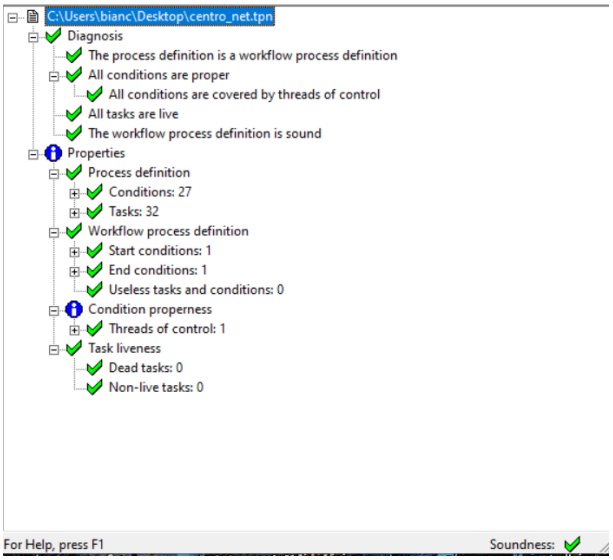

Figura 11: Analisi semantica del centro di addestramento con WOFLAN

3.3 Workflow Net Completo

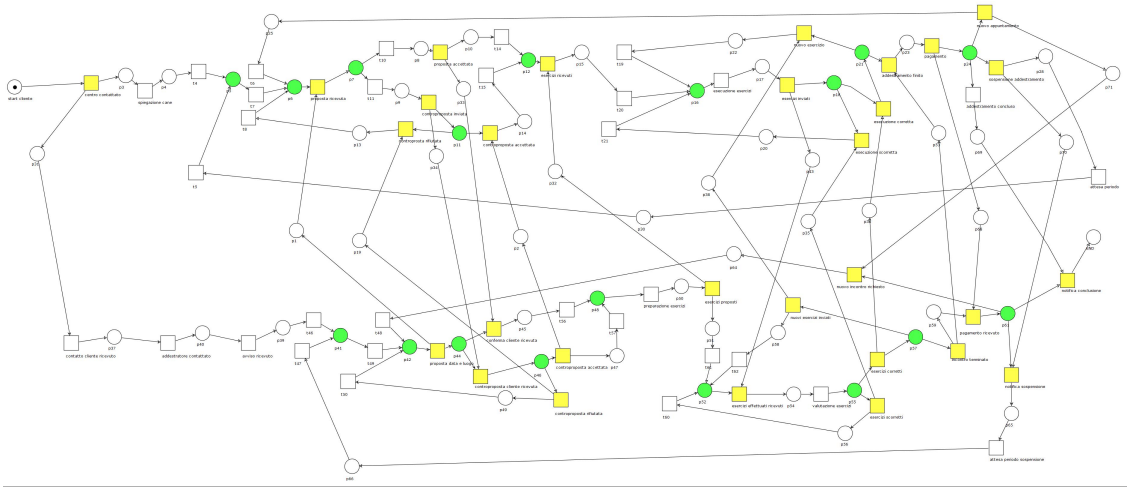

Figura 12: Dog Training System

La traduzione da BPMN a Petri Net dell'intero sistema è mostrata in Figura 12 ed è l'esito della combinazione dei due workflow modules descritti in precedenza. L'unione è avvenuta attraverso le piazze di interfaccia che permettono la comunicazione e lo scambio di messaggi tra essi. Queste piazze di interfaccia sono collegate attraverso archi alle transizioni corrispondenti in modo tale da tradurre le attività di ricezione ed invio inserite nel diagramma BPMN. Per garantire un'unica piazza iniziale è stato considerato come punto di inizio quello della Petri Net del Cliente, mentre per assicurare un'unica piazza finale è stato scelto come punto di fine quello del Centro di Addestramento. La rete completa è quindi composta da un unico *initial place* contenente un token che abilita la chiamata del cliente alla segreteria del centro, e da un unico *end place* raggiunto nel momento in cui la segreteria riceve la notifica di

conclusione definitiva dell'addestramento dell'allievo. La rete finale è composta da **66 places**, **62 transitions** e **154 arcs**, e a differenza dei singoli moduli **non è free-choice**. Infatti contiene cinque free-choice violations. Queste ultime sono causate dagli event-based gateway, poiché le transizioni in uscita da questi gateway (trasformati in place) non hanno un preset totalmente diverso o totalmente uguale. Infatti queste transizioni condividono lo stesso place relativo all'event-based gateway, ma hanno un diverso place dell'interfaccia utilizzato per comunicare con l'altro pool. Per tale motivo la workflow net non è una S-Net e nemmeno una T-Net. Infine **non è well-structured** perché presenta 150 PT-Handles e 120 TP-Handles. Nonostante queste variazioni, visibili in Figura 13 e 14, le proprietà della *Soundness* risultano convalidate. Concludendo, in Figura 15, è raffigurato il **Coverability Graph Completo** composto da 208 archi e 130 vertici. Quest'ultimo è finito e concide con il *Reachability Graph* dato che la workflow net risulta essere *bounded*. Il grafo conferma le proprietà della reta, infatti esiste sempre un arco etichettato con una transizione e quindi non abbiamo dead task. E' deadlock-free poiché non esistono nodi privi di archi uscenti, tranne quello finale. Infine non abbiamo token rimanenti nella rete e questo conferma le proprietà di soundness.

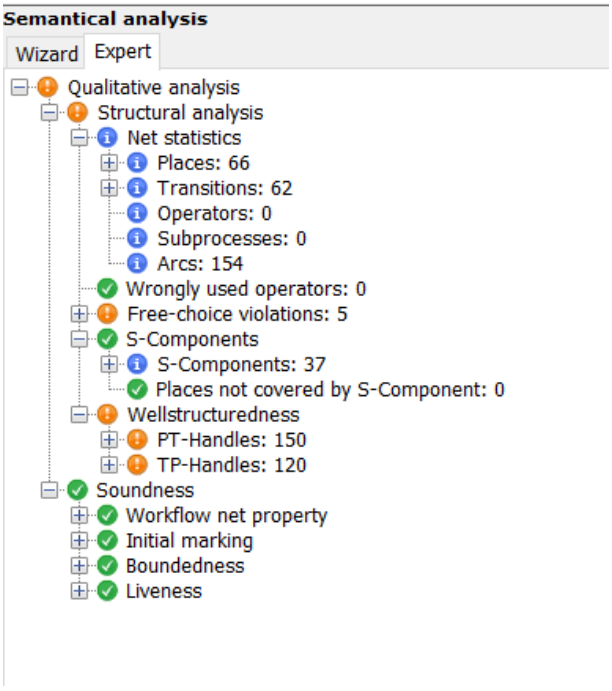

Figura 13: Analisi semantica completa con WOPED

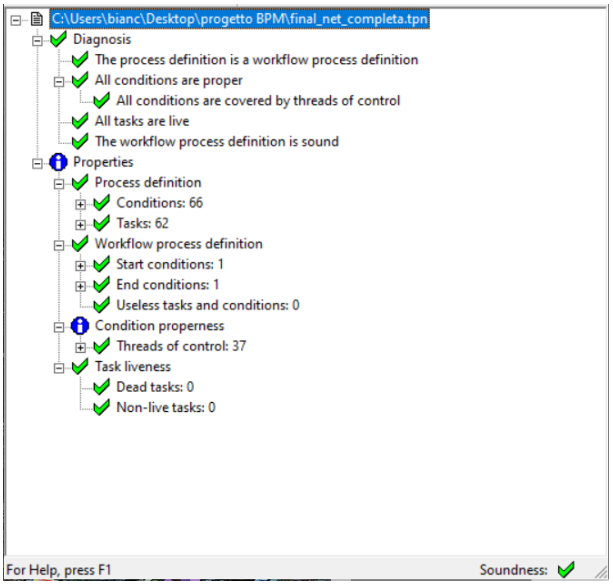

Figura 14: Analisi semantica completa con WOFLAN

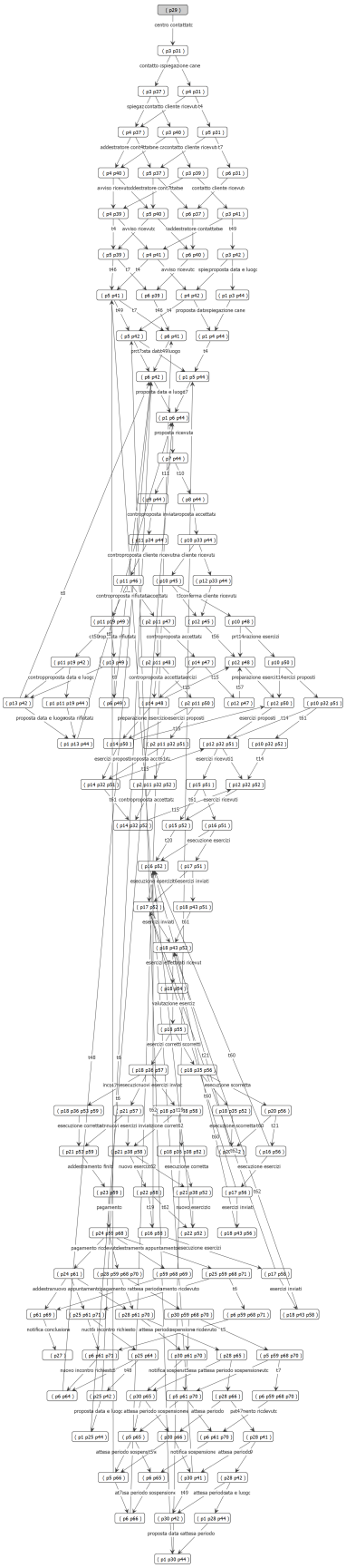

Figura 15: Coverability Graph Completo