



DOG TRAINING WORKFLOW ANALYSIS

STUDENTS GROUP

FRANCESCO GEMIGNANI, SAVERIO TELERA

BUSINESS PROCESS MODELING PROJECT

DATA SCIENCE & BUSINESS INFORMATICS

Academic Year 2021/2022

Contents

1	Diagramma BPMN	3
1.1	Client Pool	3
1.2	Dog Training Center Pool	4
1.2.1	Ufficio	5
1.2.2	Addestratore	5
2	Analisi della Workflow Net	9
2.1	Cliente	9
2.2	Centro di Addestramento	10
2.3	Dog Training System	12

Introduzione

Lo scenario si pone come obiettivo quello di modellare la gestione di un cliente che vuole iniziare un percorso di dog training con il proprio animale. I principali attori che abbiamo identificato in questo processo sono:

1. Cliente
2. Centro di addestramento

Nella prima parte abbiamo deciso di utilizzare la notazione grafica **BPMN** per descrivere la dinamica dell'intero processo, presentando i pool singolarmente. Questa notazione è stata scelta dato che si presta particolarmente alla multi-point view del nostro choreography diagram. Nella seconda parte, invece, sono state analizzate le Petri Nets risultanti in modo tale da fornire una descrizione qualitativa della struttura. I diagrammi BPMN sono stati progettati con il web tool **BPMN.io**, mentre le workflow nets sono state realizzate e analizzate con **WoPed** e **Woflan**.

Il collaboration diagram è stato descritto utilizzando 2 pools: il primo per rappresentare il **cliente** e l'altro per le attività del **centro di addestramento**. Per poter distinguere le responsabilità all'interno del centro di addestramento è stato applicato il metodo dello swimlanes, suddividendo il pool in due lanes: l'*ufficio amministrativo* e l'*addestratore*. La comunicazione tra i due attori nel diagramma è stata implementata con dei **message flow arrows**, così da rappresentare il flusso di informazioni tra i due pool. Sono stati aggiunti anche degli **artefacts**, per far sì che il processo potesse scrivere o leggere i dati che emergono dall'allenamento, in un *Data Store*. Quest'ultimo conserva le informazione oltre la vita dell'istanza del processo. In particolare, sono stati utilizzati per appuntare informazioni utili sull'esecuzione degli esercizi, per poter annotare i miglioramenti dell'animale e sui log dei pagamenti, per poter tracciare lo storico dei compensi ricevuti. I processi rappresentati nei pool sono strettamente dipendenti l'uno dall'altro, quindi, al fine di controllarne il flusso e creare una dipendenza reciproca, sono stati utilizzati dei **decision based gateway**, perciò il flusso di un attore dipenderà dalla decisione presa dalla sua controparte. Gli altri gateways utilizzati sono gli **XOR split** e **XOR join**. Nella sezione successiva verranno descritti la sequenza delle attività eseguite dal cliente e dal centro di addestramento.

1 Diagramma BPMN

1.1 Client Pool

Il pool principale che avvia l'intero processo è il cliente, il quale contatta il centro di addestramento per prenotare il primo incontro con l'addestratore. Questa attività è molto importante perché è quella che attiva il pool del centro. Abbiamo sviluppato una prenotazione alternata nel quale inizialmente il cliente riceve un appuntamento da parte dell'addestratore, che può accettare o no in base alle proprie disponibilità. In caso di rifiuto, il cliente fa una contro proposta che a sua volta può essere accettata o no dal centro. In caso di rifiuto, il cliente ripete il ciclo alternato ri-mettendosi in attesa di un altro appuntamento. Il loop termina quando viene stabilita la data e il luogo dell'incontro.

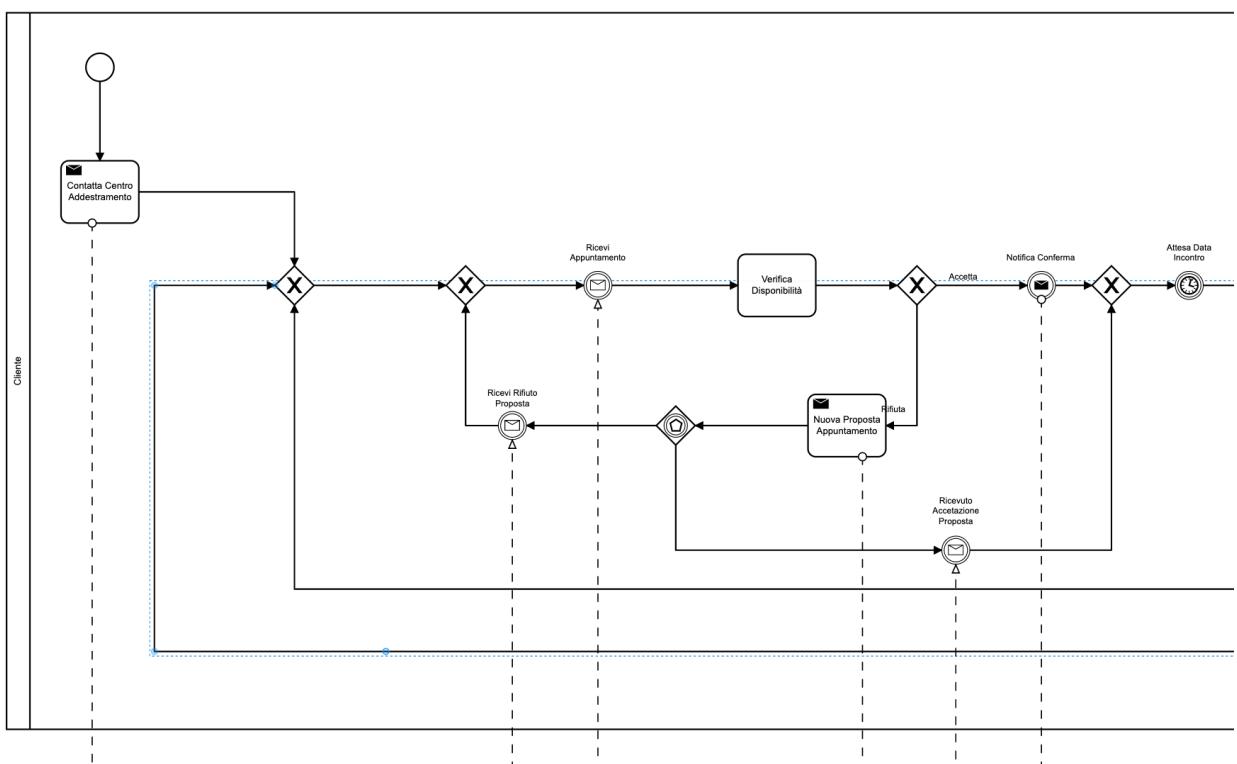


Figure 1.1: Diagramma BPMN della fase di prenotazione del cliente

In particolare, prima di ricevere la proposta dell'incontro, composta da data e luogo, sono stati inseriti due XOR join per gestire le ripetizioni delle eventuali prenotazioni che possono avere luogo al termine dell'incontro o durante la fase di prenotazione. Dopo aver ricevuto la proposta, il cliente verifica la disponibilità e utilizza uno XOR split per decidere se accettare l'appuntamento oppure rifiutare e inviare una contro-proposta. Nel primo caso la fase di prenotazione termina. Abbiamo inserito un **intermediate time event** per rappresentare il tempo che intercorre tra la fine della fase di prenotazione e il giorno dell'incontro. Nell'altro caso, il cliente invia una nuova proposta, passando il controllo del flusso all'addestratore, che a sua volta può accettare o ri-proporre un nuovo appuntamento. Per descrivere questa

dipendenza abbiamo inserito un **event-base gateway**. Se la contro-proposta del cliente viene accettata, la prenotazione termina passando al giorno dell'incontro. Altrimenti se non viene accettata si ripete l'intero loop, utilizzando lo XOR join iniziale, fondamentale per la procedura di prenotazione.

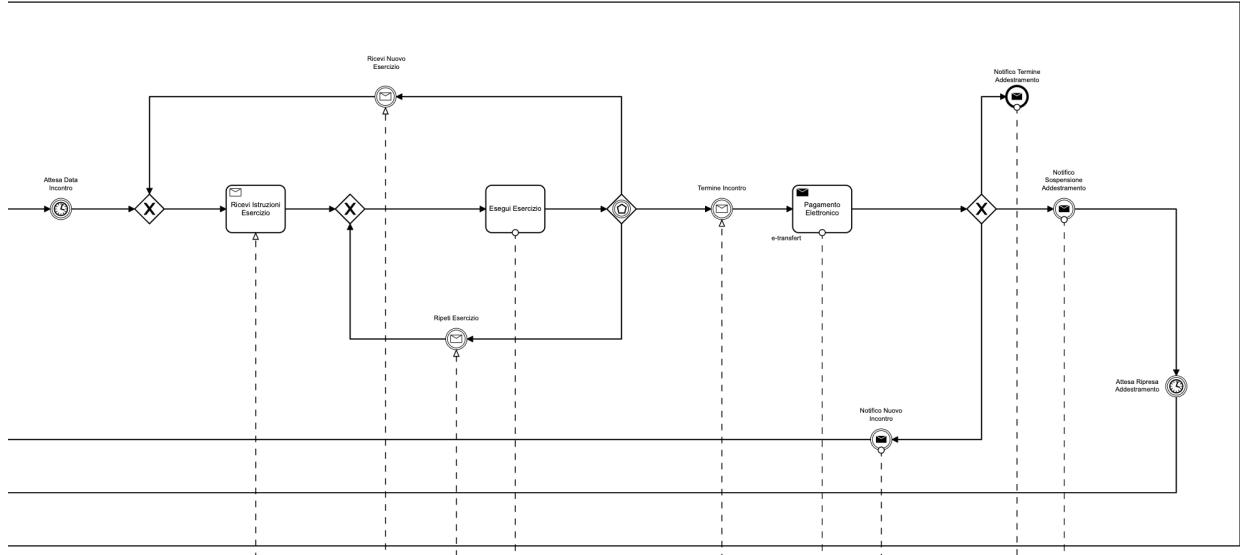


Figure 1.2: Diagramma BPMN dell'attività e dell'avanzamento della sessione

Nella fase successiva, una volta arrivato il momento dell'allenamento, il cliente riceve le istruzioni relativamente all'esercizio da svolgere e successivamente lo esegue. I due XOR join sono necessari per re-iterare la procedura di esecuzione di un nuovo esercizio o di ripetizione dello stesso, nel caso in cui non venga svolto correttamente. Quando l'esercizio è terminato, il cliente passa il controllo del flusso all'addestratore, aspettando di ricevere un riscontro. L'**event-based gateway** attende quindi il verificarsi di un evento, che può essere: l'esecuzione di un nuovo esercizio, la ripetizione dello stesso oppure la terminazione dell'incontro. Nei primi due casi il cliente viene avvisato della corretta o scorretta esecuzione dell'esercizio, passando rispettivamente alla spiegazione del successivo o alla ripetizione dello stesso. Altrimenti, se la sessione è terminata, il cliente provvede ad inviare un pagamento elettronico per poi decidere con l'ultimo XOR split la scelta successiva. Nel caso in cui intenda proseguire gli incontri, il cliente notifica la scelta all'addestratore re-iterando la procedura di prenotazione fino allo XOR join iniziale. La connessione tra lo XOR join iniziale e lo XOR split finale sono necessarie per garantire la ripetizione di un nuovo incontro. Inoltre, ci è stato richiesto di considerare la possibilità che il cliente possa sospendere gli incontri e riattivarli in un secondo momento. In questo caso abbiamo inserito un **intermediate time event** che rappresenta la durata di questo periodo di inattività. Allo scadere del timer, l'attività al solito procede con l'effettuare una nuova prenotazione. Altrimenti, se il cliente decide di terminare gli incontri, notifica la scelta all'ufficio facendo terminare l'intero processo.

1.2 Dog Training Center Pool

Il secondo attore del nostro modello è il centro di addestramento, quest'ultimo si attiva nel momento in cui il cliente decide di iniziare un allenamento con il proprio cane contattando il numero telefonico apposito. Per poter rappresentare questo attore abbiamo utilizzato il meccanismo delle **swimlanes**. Il nostro pool è stato suddiviso in due corsie, utilizzate per organizzare in categorie visive separate le responsabilità dell'intero centro di addestramento. Abbiamo identificato in particolare due corsie, una per l'**ufficio amministrativo** e l'altra

per l'**addestratore**. L'ufficio amministrativo si occuperà di gestire la richiesta iniziale del cliente, associando ad esso un allenatore che faccia al caso suo, ed infine il pagamento al termine di ogni sessione di allenamento. Per quanto riguarda l'allenatore, una volta selezionato dall'ufficio, viene messo in contatto con il cliente per concordare il giorno dell'appuntamento e successivamente si occupa dell'allenamento, fornendo degli appositi esercizi e valutando opportunamente la performance del cane.

1.2.1 Ufficio

In questa prima corsia possiamo osservare le attività dell'ufficio amministrativo del centro di addestramento

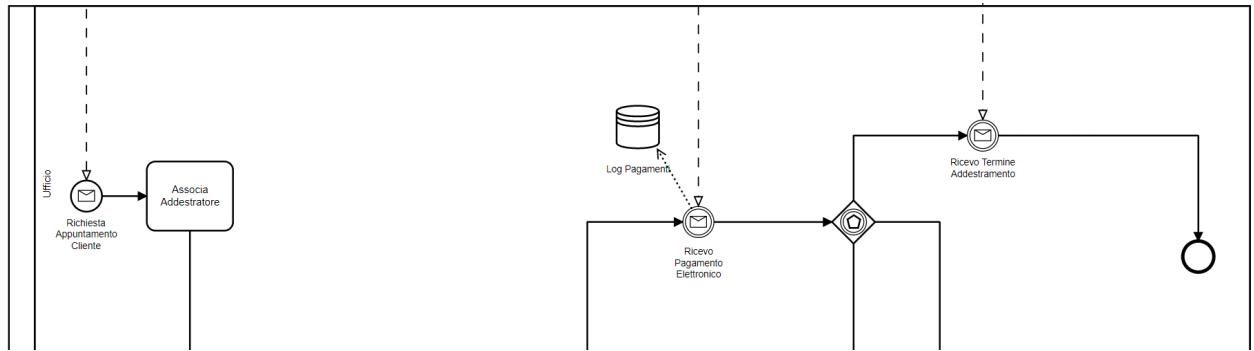


Figure 1.3: Diagramma BPMN dell'ufficio amministrativo

Come dato dalle specifiche fornite, una volta ricevuta la chiamata del cliente, l'ufficio provvede ad associalo un addestratore competente per lo specifico caso. Al termine di ogni sessione di allenamento, l'ufficio gestirà il pagamento, come in figura 1.3, registrando la transazione nell'apposito **log dei pagamenti**. Successivamente abbiamo un event-base gateway, seguito da catching events. Il flusso di sequenza viene suddiviso in 3, in particolare solo una delle opzioni è di responsabilità del ufficio amministrativo, la notifica della volontà del cliente di terminare l'addestramento. Infine, dopo il messaggio, il processo termina con l'attività finale.

1.2.2 Addestratore

La responsabilità passa all'addestratore dopo essere stato associato al cliente dall'apposito ufficio.

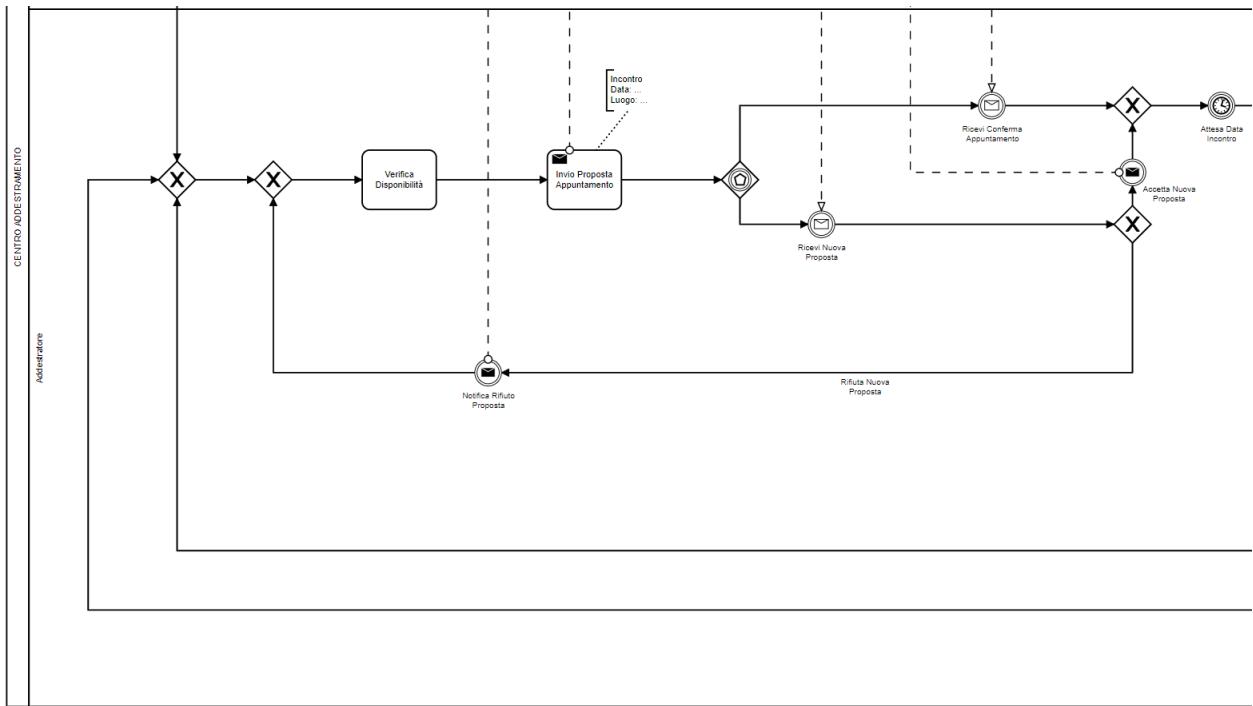


Figure 1.4: Diagramma BPMN fase di prenotazione dell'appuntamento

Nella corsia dell'addestratore, prima di verificare la disponibilità e di inviare la proposta della data e luogo, sono stati inseriti due XOR join per gestire le ripetizioni delle eventuali prenotazioni, nel caso in cui il cliente decida di prenotare una nuova sessione di allenamento oppure nel caso di una controproposta del cliente su una nuova data e luogo. Il meccanismo di prenotazione è simmetrico a quello spiegato precedentemente nel pool del cliente. Un **event base gateway** è stato posizionato, seguito da due catching events. Nel caso in cui si riceva conferma da parte del cliente sulla data e luogo proposti si procede all'evento dall'attesa della data dell'incontro con il rispettivo **intermediate time event**, nel caso di rifiuto, invece, l'allenatore può accettare la nuova proposta, oppure rifiutarla proponendone una nuova, fino a ché i due non sono d'accordo.

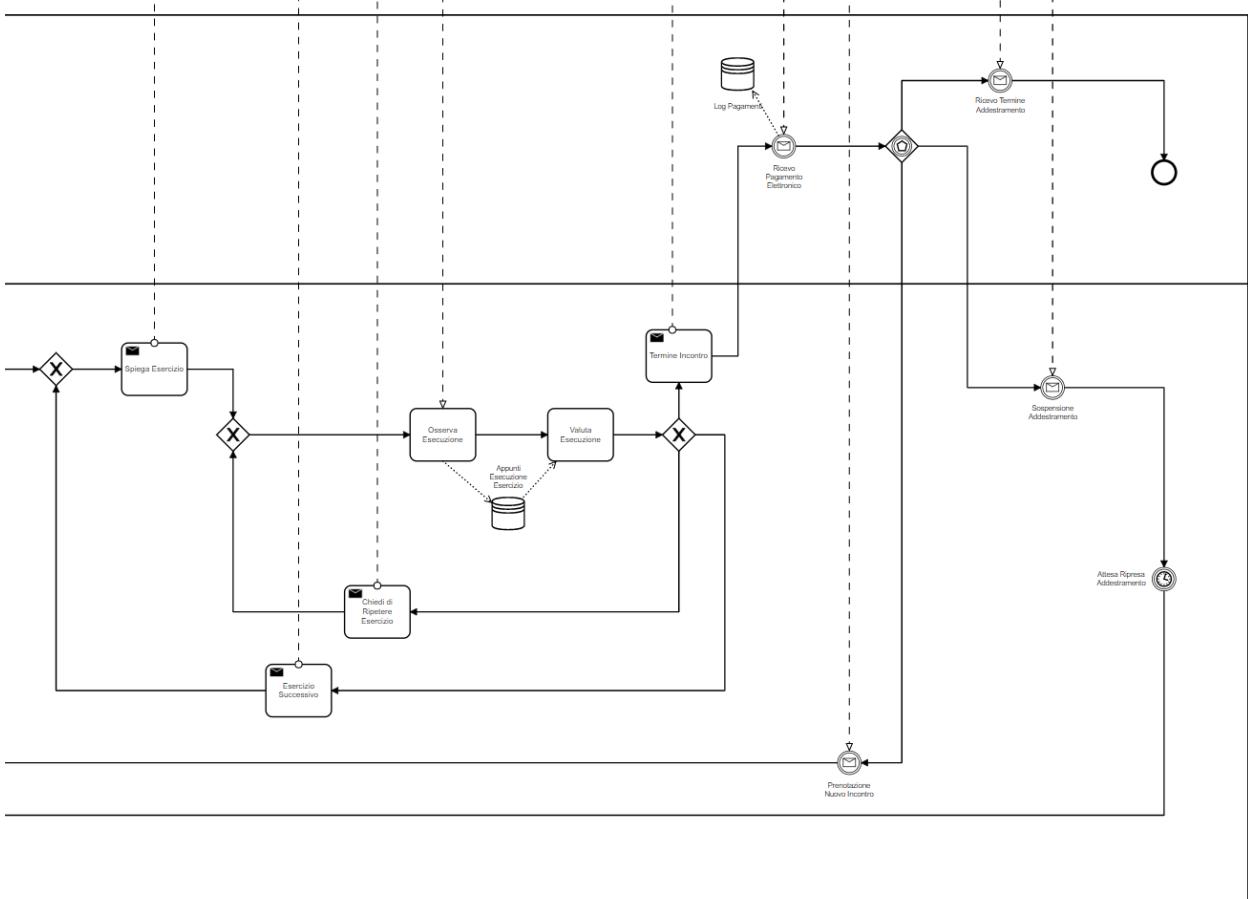


Figure 1.5: Diagramma BPMN allenamento

Una volta arrivato il giorno dell’allenamento, abbiamo posizionato uno XOR join per gestire la sequenza di esercizi in una sessione di allenamento. Dopo la spiegazione dell’esercizio troviamo un altro XOR join per gestire il caso in cui l’allenatore richieda la ripetizione dello stesso esercizio. A seguireabbiamo l’attività di osservazione e valutazione dell’esecuzione svolta dal cliente ed uno XOR split in cui il flusso si divide in 3:

- Nel caso di valutazione negativa l’allenatore chiede di ripetere il medesimo esercizio
- Nel caso di esito positivo, l’allenatore, in base alle condizioni fisiche del cane del cliente, può decidere di:
 - Proporre un nuovo esercizio
 - Terminare la sessione dell’allenamento

Una volta terminato l’allenamento e ricevuto il pagamento in base alla decisione del cliente, gestita dal event-base gateway nella corsia dell’ufficio amministrativo, l’allenatore in base alla decisione del cliente può:

- Procedere alla prenotazione del nuovo incontro, facendo tornare il flusso all’attività di proposta della data e luogo dell’appuntamento
- Procedere alla sospensione temporanea dell’allenamento, attendendo la data di ripresa, resa esplicita dal intermediate time event, prima di far tornare il flusso alla proposta della data e luogo

Il BPMN model completo è descritto nell’immagine 2.9

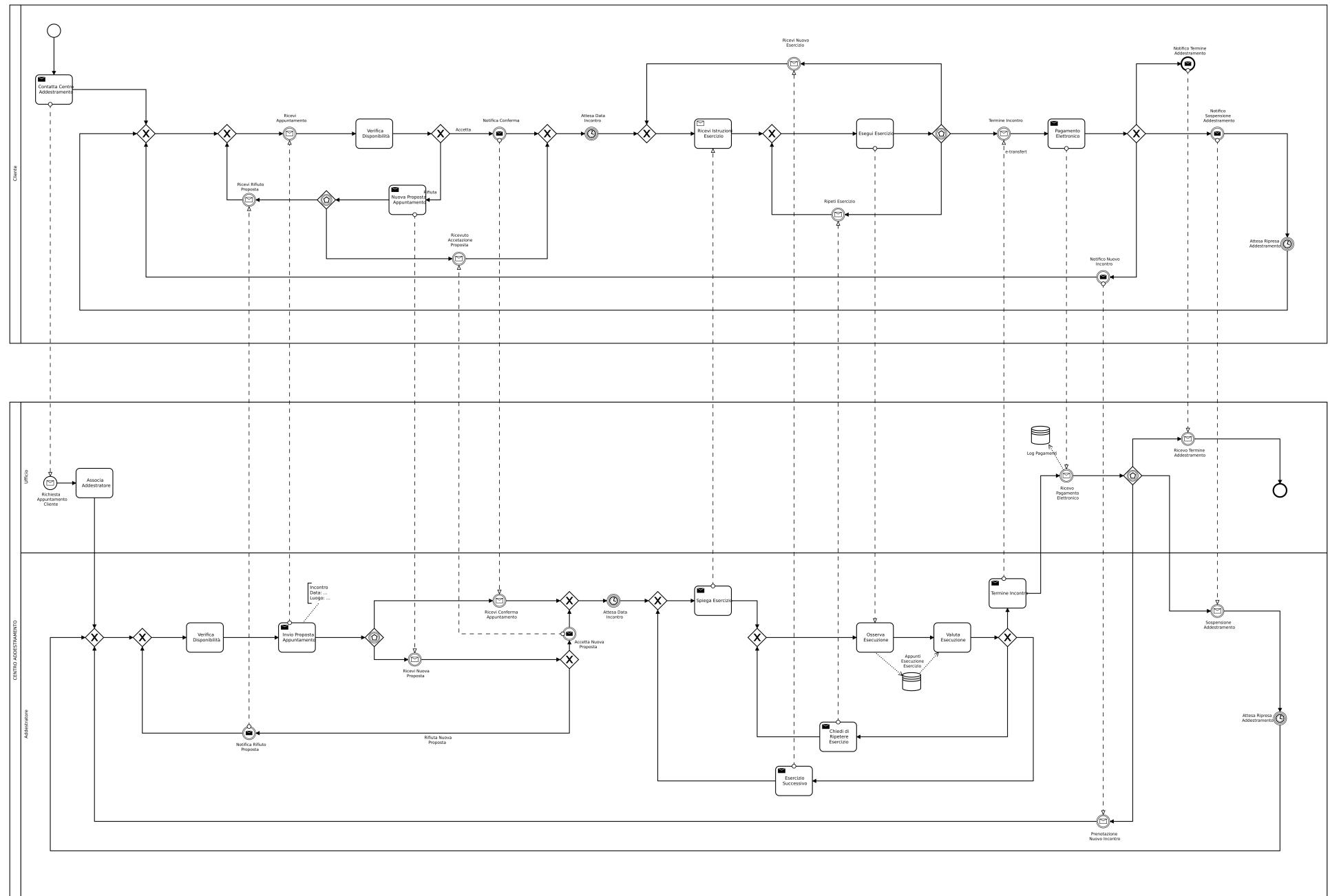


Figure 1.6: Diagramma BPMN completo

2 Analisi della Workflow Net

Abbiamo trasformato il diagramma BPMN in workflow net in modo tale da verificare le proprietà strutturali e analizzarla qualitativamente. Il processo di trasformazione è stato eseguito come segue: abbiamo prima aggiunto un place per ogni collegamento, rappresentando ogni evento, attività e gateway con una transition (prestando attenzione all'event-based gateway). Successivamente abbiamo eseguito il de-sugar della rete aggiungendo un unico initial place seguito da un AND split ed un unico sink place preceduto da un AND join, con un singolo token nel place iniziale. Successivamente riportiamo le Petri Net del cliente, del centro di addestramento e del workflow complessivo.

2.1 Cliente

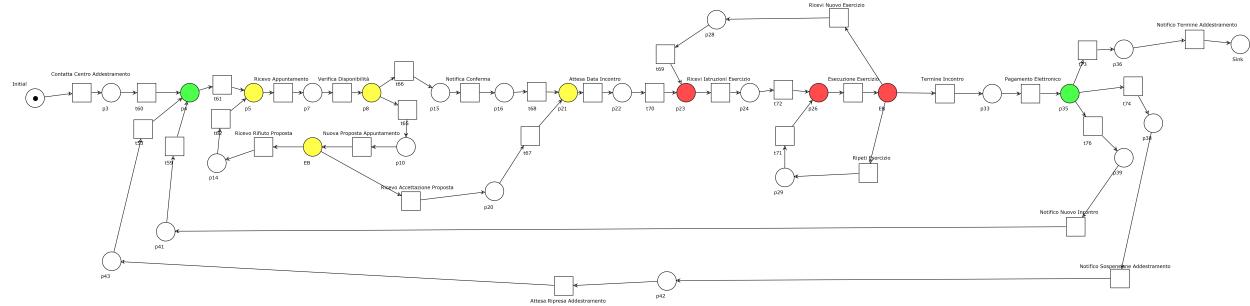


Figure 2.1: Workflow net del cliente.

La rete è composta da 29 places, 34 transitions e 68 arcs. E' una **workflow net** in quanto è composta da un unico initial place i , un unico final place o e ogni place e transition appartengono ad almeno un cammino da i fino ad o . Poichè la rete N è **S-net** allora anche N^* (implicita in WoPed) è tale, di conseguenza dato che $M_0(P) > 0$ possiamo immediatamente concludere che è **live** e **bounded**, quindi **sound**. La rete è **safe**, infatti il numero totale di token nel marking iniziale è 1 ($M_0(P) = 1$) e nelle s-net il numero di token per ogni marking rappresenta un invariante. Le proprietà della soundness vengono tutte rispettate, non esistono **dead task** e le proprietà **proper completion** e **option to complete** sono sempre soddisfatte. La rete è **free-choice**, infatti per ogni coppia di transitions i rispettivi input places set non hanno mai un'intersezione parziale. Inoltre è **well-structured** in quanto non esistono tp-pt handles. La rete, essendo free-choice, live e bounded, conferma di essere **s-coverable** con un unico s-component che include 63 elementi. Il **coverability graph** è composto da 29 nodi e 34 vertici.

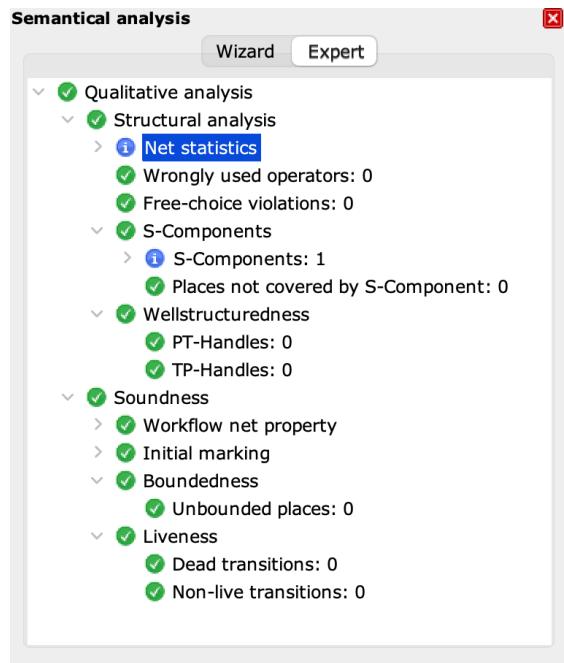


Figure 2.2: Analisi semantica del cliente (WoPed)

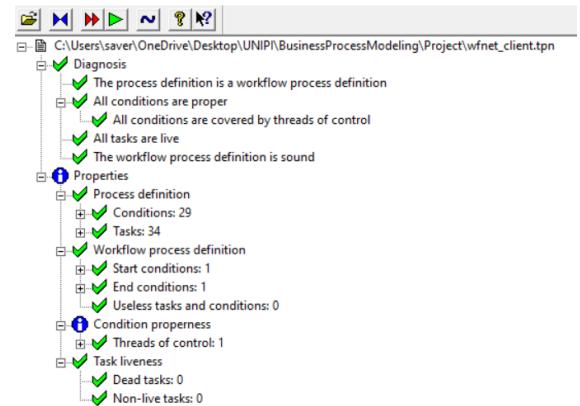


Figure 2.3: Analisi semantica del cliente (Woflan)

2.2 Centro di Addestramento

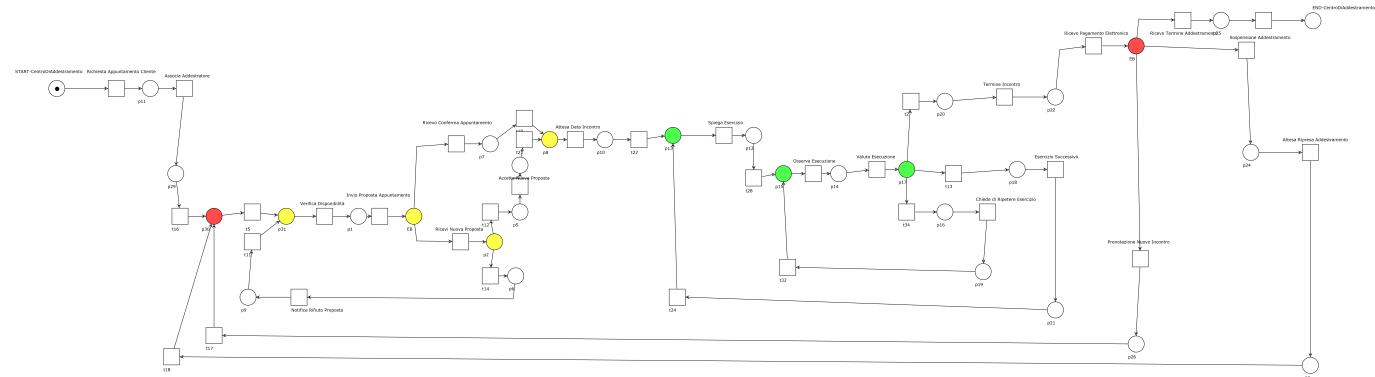


Figure 2.4: Workflow net del centro di addestramento.

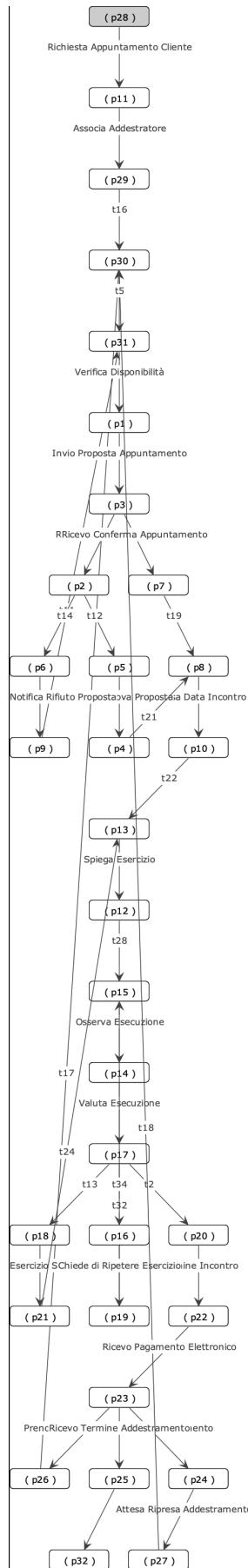


Figure 2.5: Coverability graph del centro di addestramento.

Il grafo è composto da 32 places, 37 transitions e 74 arcs. La definizione di workflow net è rispettata per le stesse motivazioni descritte nel cliente. La rete, come quella precedente, è **S-net** e rispetta tutte le proprietà della **soundness**. Come possiamo verificare anche dal **coverability graph**, non esistono **dead task**, infatti esiste sempre almeno un arco etichettato con la transizione della rete. Inoltre possiamo notare che sia la **option to complete** che la **proper completion** sono soddisfatte. Una volta che un token raggiunge p32 non rimangono token in nessun place della rete. Il coverability graph è composto in totale da 32 nodi e 37 archi e coincide col reachability graph. Poiché la rete è **bounded**, il grafo risulta finito. Per motivi di spazio non abbiamo inserito il coverability graph del cliente. In ogni caso gran parte delle considerazioni fatte sono le stesse. La rete, inoltre, è **free choice** e **well-structured**. Ha una **S-invariant** positiva della forma $[k, k, \dots, k]$ ed una semi-positive **T-invariant**. Infine la net è **s-coverable** da un unico s-component che include 69 elementi.

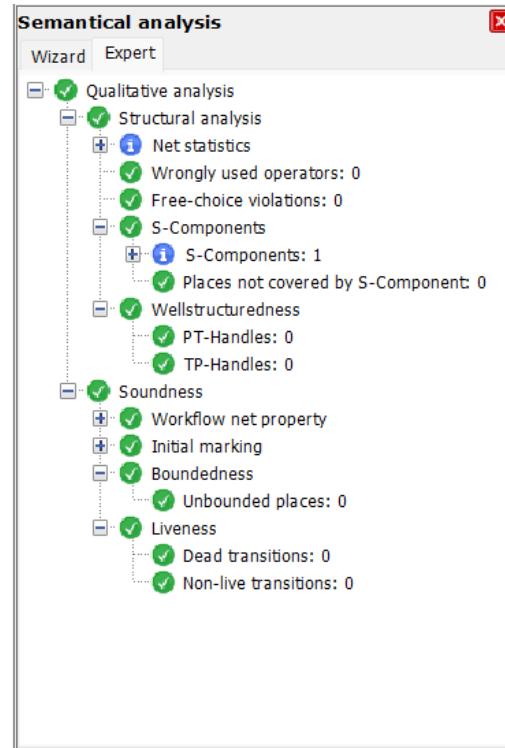


Figure 2.6: Analisi semantica del centro di addestramento.

2.3 Dog Training System

Per descrivere il workflow complessivo, abbiamo prima trasformato le precedenti reti in moduli, aggiungendo un insieme di input/output place per ogni interfaccia. Il workflow system lo abbiamo ottenuto unendo i due moduli strutturalmente compatibili. E' composto da un unico initial place contenente un token, che abilita la chiamata al centro di addestramento da parte del cliente e da un unico sink place raggiunto nel momento in cui l'ufficio riceve la notifica di terminazione degli incontri da parte del cliente. La strongly connection è garantita dalla reset transition, la quale collega il sink place al place iniziale. Tale collegamento non è visibile in quanto implicito in WoPed.

Il sistema è composto da **74 places**, **71 transizioni** e **170 archi**. A differenza dei singoli moduli, non abbiamo una rete **free-choice** ne **well-structured**. Dopo una breve analisi abbiamo notato che le violazioni che non la rendono free-choice sono generate dagli **event-based gateway**. In particolare le transizioni in uscita da tali gateway hanno un input place set con un'intersezione parziale; entrambe condividono lo stesso place dell'event-based gateway ma un diverso place dell'interfaccia (per comunicare con l'altro pool). Per questo motivo non si tratta nemmeno di una **S-net** (ne T-net). La figura 2.8 mostra il workflow system nella sua interezza.

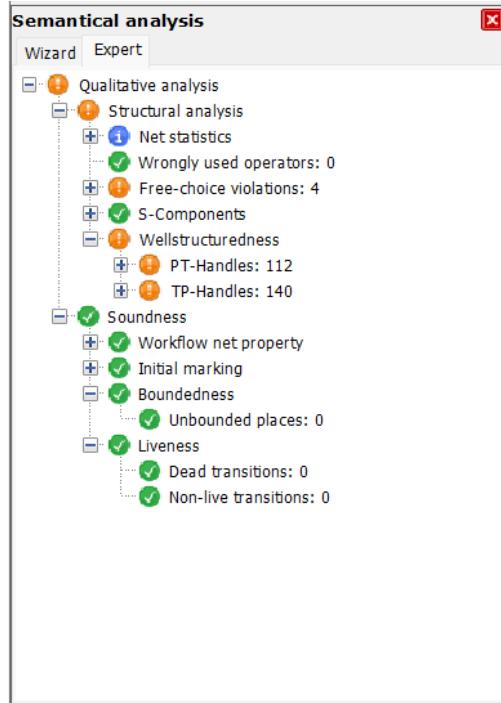


Figure 2.7: Analisi semantica WFNet completa con Woped

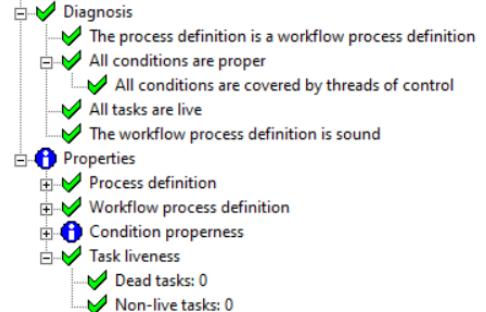


Figure 2.8: Analisi semantica WFNet completa con Woflan

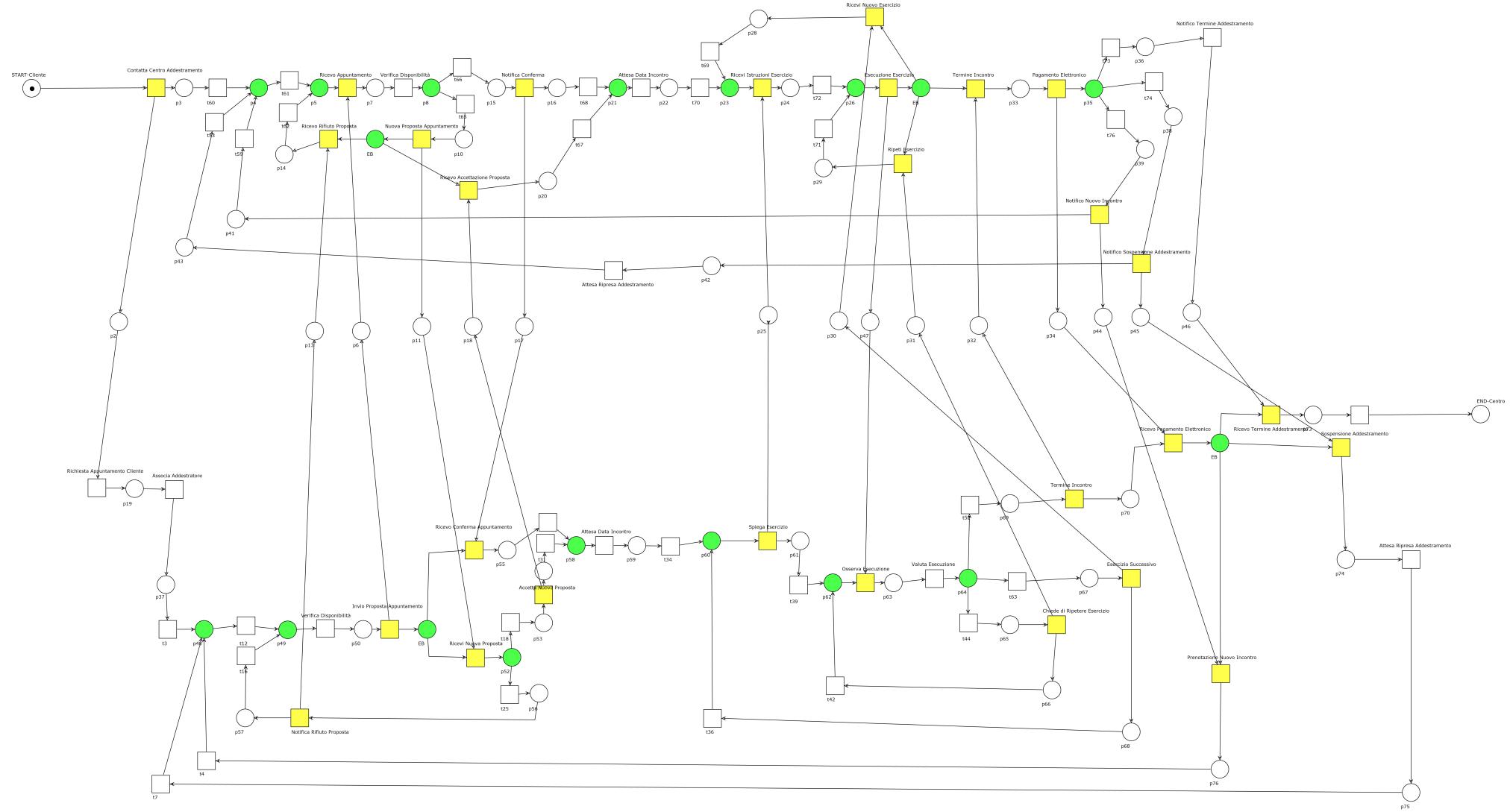


Figure 2.9: wfnet completa

Come ultimo step abbiamo costruito il reachability graph della workflow net completa, illustrato nell'immagine 2.10, dal quale emerge il problema della **state explosion**. Il grafo conferma le proprietà della rete: la **liveness** viene garantita dal fatto che da ogni nodo del RG possiamo sempre raggiungere un altro marking che ha un arco in uscita etichettato con la live transition (e questo è vero per ogni transizione della rete). Inoltre, esiste sempre un arco etichettato con una transizione della rete, quindi non abbiamo **dead task**. Il grafo è finito in quanto **bounded** (non abbiamo omega-nodes) e ovviamente è **deadlock-free** visto che non esistono nodi privi di archi uscenti. Infine, il sink place (p48) viene sempre marcato e non abbiamo token rimanenti nella rete, confermando le proprietà della **soundness**.

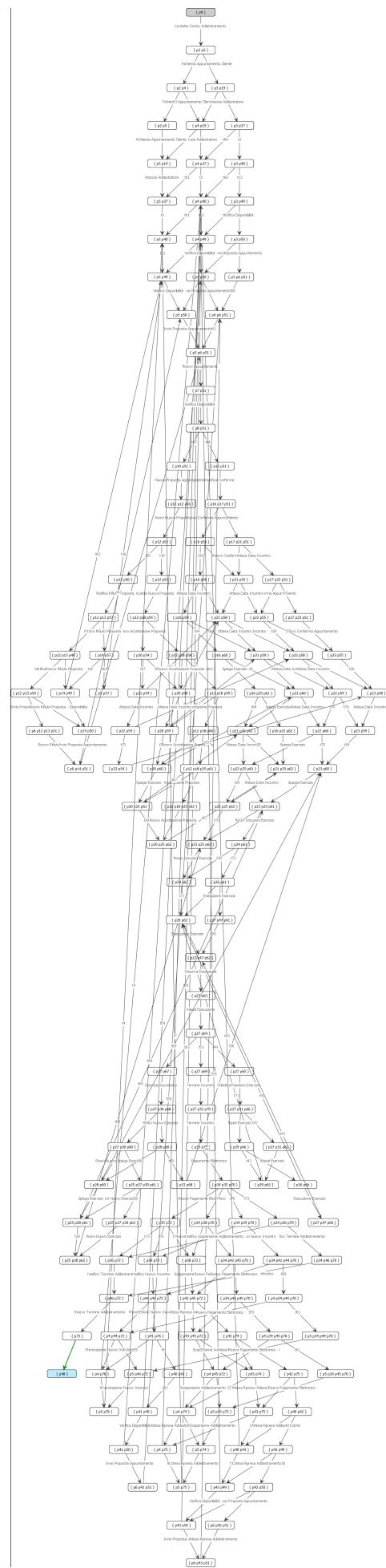


Figure 2.10: Reachability graph