Progetto DALI

Katiuscia Pellone - matricola 288642

Descrizione

L'obiettivo del progetto è di realizzare un sistema che simuli il processo di validazione dei progetti per un esame universitario.
In questo contesto ci sono due tipologie di agenti: studente (che può comportarsi in modo corretto o malevolo) e professore.

Lo studente, una volta sveglio, saluta e attende la pubblicazione delle specifiche del progetto. Ogni 10 secondi, fino a un massimo di tre volte, controlla se le specifiche sono disponibili. Se nel frattempo riceve una notifica di pubblicazione, le legge e sceglie (in modo random) la traccia prendendola da una lista di opzioni. Se non riceve la notifica manda un messaggio al professore per ricordargli di pubblicarle.
Dopo aver selezionato la traccia, invia la proposta. Se questa viene approvata, inizia a lavorare e può consegnare quando ha completato il progetto. In caso di rifiuto può inviare una richiesta al professore per fissare un ricevimento al termine del quale riceverà la traccia su cui lavorare.
Quando riceve il voto (un numero compreso tra 1 e 5), lo scrive.

Il professore, dopo essersi svegliato e aver salutato, lancia una moneta per decidere se pubblicare subito le specifiche del progetto. Se esce testa, le pubblica immediatamente, altrimenti lo farà solo quando riceverà la notifica da uno studente (in caso di più notifiche, le pubblicherà una volta sola).

Le specifiche vengono utilizzate anche per verificare la conformità delle tracce proposte. Nella conoscenza del professore sono presenti informazioni su quali tracce sono sincrone e/o deterministiche, e tali caratteristiche influenzano la l'approvazione o la bocciatura della proposta. Questi dati possono cambiare nel tempo (per esempio, una traccia inizialmente deterministica potrebbe non esserlo più a seguito di modifiche) e il professore riceve notifiche quando le informazioni devono essere aggiornate. Uno studente malevolo , in seguito a un rifiuto, potrebbe cercare di alterare queste caratteristiche per far sì che la propria traccia venga approvata. Per prevenire questi tentativi è stato implementato un filtro TOLD nel file *communication.con* che blocca queste notifiche se vengono inviate dagli studenti.

Se la traccia proposta soddisfa i requisiti, il professore la accetta. In caso contrario la rifiuta e può ricevere richieste di ricevimento, alle quali risponde indicando quando è disponibile per un colloquio. Per la simulazione la fine del ricevimento viene rappresentata con l'invio di un messaggio allo studente contenente la traccia su cui può lavorare.

Quando uno studente consegna il progetto, il professore lo aggiunge alla lista dei lavori da correggere. Quando la lista contiene almeno due progetti, inizia la correzione. Questo è stato fatto per non far correggere i lavori appena ricevuti, rendendo il comportamento dell'agente non puramente reattivo. Se ci sono esattamente due progetti li corregge entrambi, altrimenti dopo averne corretto uno decide se continuare o fermarsi per andare a fare altro (o per non fare nulla).

Tabella Event / Action - professore

Evento/Azione	Descrizione
sveglia!	Il professore si sveglia e, se non lo ha già fatto, saluta tutti.
decido_se_pubblicare!	Lancia una moneta per decidere se pubblicare subito oppure no le specifiche del progetto.
pubblicoSpecifiche!	Se è uscita testa, le pubblica e invoca *broadcast*.
broadcast	Invia a tutti gli studenti un messaggio per comunicare che ha pubblicato le specifiche.
sollecitoE	Se è uscita croce, riceverà prima o poi un messaggio da parte degli studenti che lo invitano a presentare le caratteristiche del progetto e, se non le ha già pubblicate, procede a farlo richiamando *pubblicoSpecifiche*.
propostaE(S, T)	Riceve da uno studente S la proposta di traccia T e procede con la valutazione.
valutaA(S, T)	Se la traccia è determinista o sincrona, la rifiuta e manda un messaggio *rifiutata* allo studente, altrimenti la accetta e manda *accettata.*
richiestaRicevimentoE(S)	Riceve da uno studente S la richiesta per un ricevimento e gli risponde dicendo quando è disponibile.
updateE(F, T)	Riceve la notifica che lo invita ad aggiornare le sue conoscenze aggiungendo un fatto di tipo F(T) dove F può essere deterministico/ non_deterministico/sincrono/asincrono e T è l'identificatore della traccia. Invoca *u(F, T)*.
updateE(OldF, OldT, NewF, NewT)	Riceve la notifica che lo invita a cancellare il fatto OldF(OldT) e ad aggiungere NewF(NewT). Invoca prima *d(OldF, OldT)* e poi *u(NewF, NewT)*.

Evento/Azione	Descrizione
u(F, T)	Se F sta per “sincrono”, fa assert(sincrono(T)) Se F sta per “asincrono”, fa assert(asincrono(T)) e così via u(F, T) :- (F = sincrono -> assert(sincrono(T)); F = deterministico -> assert(deterministico(T)); F = asincrono -> assert(asincrono(T)); F = non_deterministico -> assert(non_deterministico(T))).
d(F, T)	Se F sta per “sincrono”, fa retract(sincrono(T)) e così via d(F, T) :- (F = sincrono -> retract(sincrono(T)); F = deterministico -> retract(deterministico(T)); F = asincrono -> retract(asincrono(T)); F = non_deterministico -> retract(non_deterministico(T))).
consegnaE(S, M)	Riceve da uno studente S il messaggio per la consegna del progetto e lo appende alla lista di progetti da correggere.
monitoroConsegneI	Tiene sotto controllo la lista di progetti da correggere e quando ce ne sono almeno 2, procede con le correzioni invocando *correggoA(L)*.
correggoA(L)	Inizia le correzioni e invoca *gestisciLista(L).*
gestisciLista([H | T])	Valuta un progetto alla volta. Il voto viene assegnato scegliendo randomicamente un voto da 1 a 5 e viene comunicato allo studente, dopodiché il progetto viene rimosso dalla lista. Se ci sono solo due progetti, vengono corretti entrambi. Se ce ne sono tre o più, dopo averne corretto uno lancia una moneta per decidere se continuare a correggere o andare a fare altro. Se esce testa passa al successivo altrimenti scrive che si ferma e va a gestire, eventualmente, altre cose.

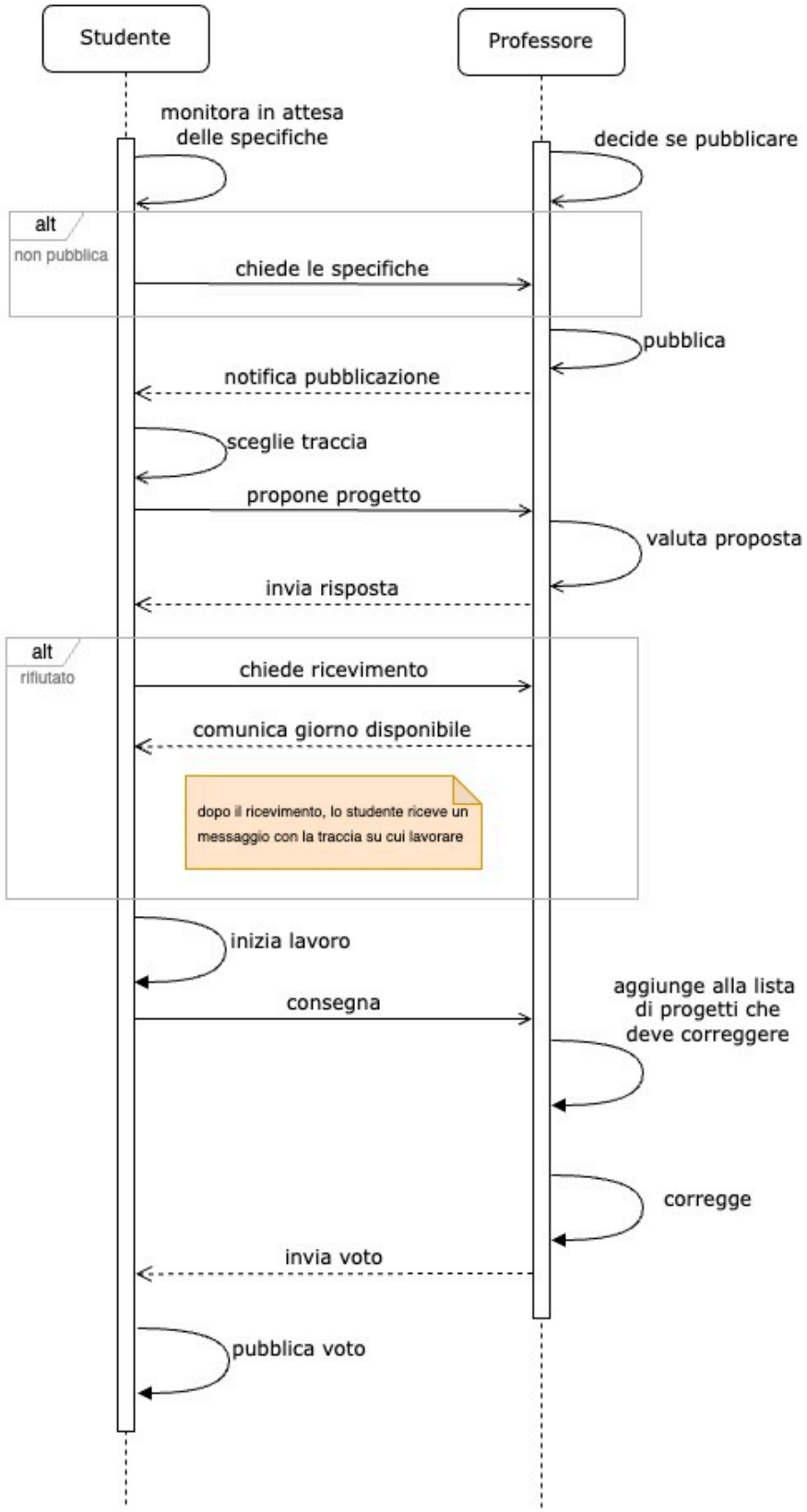
Tabella Event / Action - studente

Evento/Azione	Descrizione
sveglia!	Lo studente si sveglia e, se non ha salutato, scrive un messaggio per farlo.
attendoSpecifiche!	Dopo essersi svegliato e aver salutato, si mette in attesa delle specifiche. Ogni 10 secondi controlla se sono state pubblicate, ma lo fa per massimo tre volte.
pubblicazioneE	Quando le specifiche vengono pubblicate, gli studenti ricevono una notifica e sono invitati a leggerle.
noPubblicazione!	Se, dopo aver controllato tre volte, le specifiche non sono state ancora presentate, lo studente invia un messaggio al professore per richiederle.
scelgoTraccia!	Dopo aver letto le specifiche, se lo studente non ha già scelto la traccia, ne prende una randomicamente da una lista.
contattoProf!	Dopo aver scelto la traccia, se non ha già contattato il professore, lo studente invia la proposta di progetto.

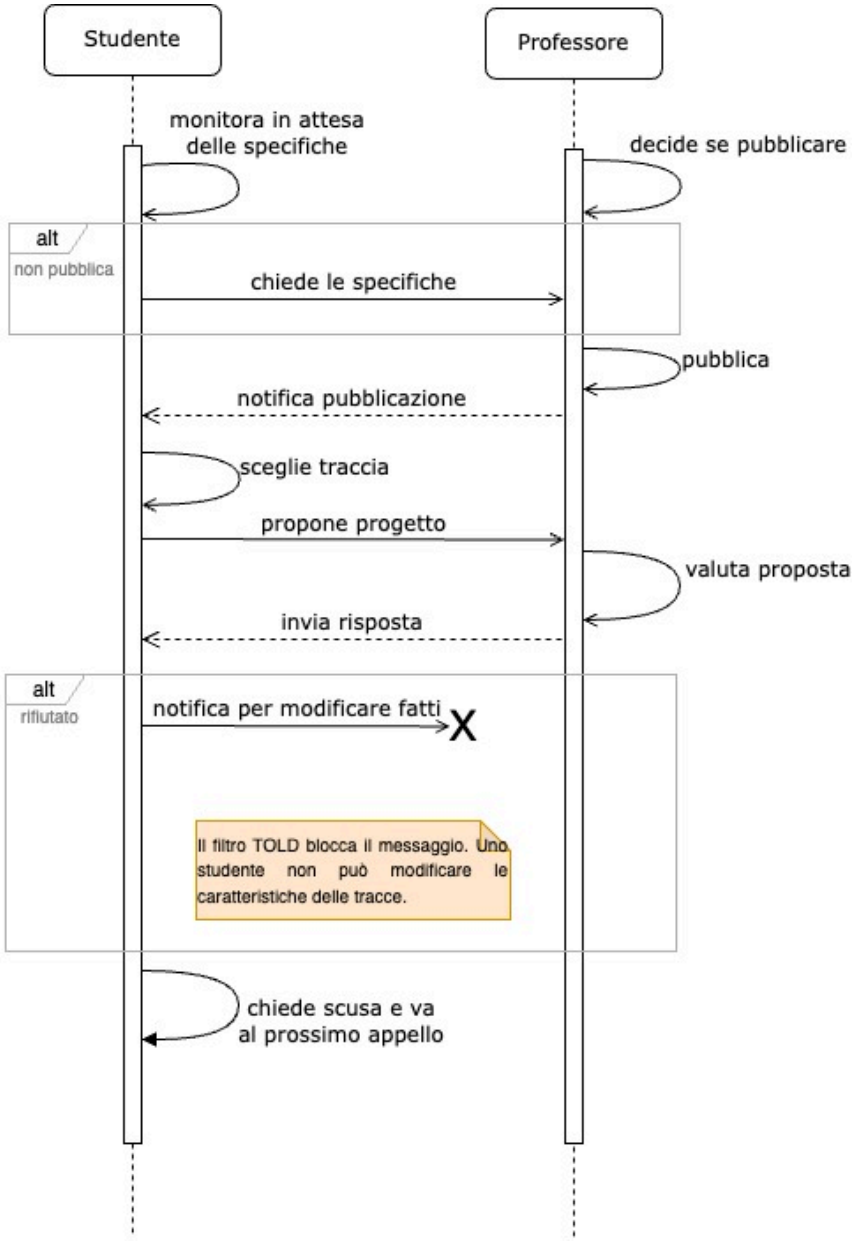
Evento/Azione	Descrizione
accettataE	Quando la proposta viene accettata, lo studente ha il progetto su cui poter lavorare.
rifiutataE	Quando la proposta viene rifiutata, invia una richiesta di ricevimento al professore. Solo lo studente malevolo prova a inviare notifica per barare aggiornando i fatti.
ricevimentoE(Quando)	Il professore risponde comunicando quando è disponibile e lo studente lo scrive in agenda (nei fatti).
ricevimentoFattoE(T)	Dopo aver fatto il ricevimento, lo studente riceve un messaggio con la traccia T su cui può iniziare a lavorare.
lavoroI	Se ha il progetto su cui lavorare e non lo ha finito, lavora.
consegnoI	Se ha finito il progetto, invia un messaggio al professore per consegnarlo.
votoE(V)	Quando riceve la valutazione, la scrive e saluta tutti.

Sequence Diagrams

Nel primo sequence diagram è rappresentato lo scenario normale tra uno studente che si comporta in modo corretto e il professore.



Nel secondo sequence diagram invece è rappresentato il comportamento di uno studente malevolo che, quando si vede rifiutare la traccia, prova a inviare un messaggio per modificare le conoscenze del professore e far sì che la sua traccia venga accettata. Il filtro TOLD non permette la ricezione del messaggio da parte del docente e stampa un messaggio “Eliminated message” per indicare che è stato scartato. Lo studente poi si scusa pubblicamente.



Simulazione

In questo primo esempio di esecuzione del progetto è possibile notare che il professore pubblica immediatamente le specifiche del progetto, tutti gli studenti le ricevono e procedono con la scelta della traccia e l'invio della proposta. Il professore approva la traccia scelta dallo studente 1 mentre boccia quelle degli studenti 2 e 3, spiegando il motivo (una traccia è sincrona e l'altra è deterministica).
In figura (a) si può osservare lo studente 3 che prova a modificare le informazioni del professore sostenendo che la traccia 7 adesso è non deterministica. Questo messaggio viene correttamente bloccato dal filtro e viene stampato un messaggio "Eliminated message: conditions not verified for […]".
In figura (b) si può vedere invece che lo studente 2 manda una richiesta per fissare un ricevimento e segna in agenda la data ricevuta dal professore. La fine del ricevimento è simulata con l'invio di un messaggio allo studente contenente la traccia su cui può lavorare. Dopo averlo letto, inizia a lavorare.
Quando ci sono almeno due progetti, il professore inizia a correggere. In questo caso ce n'erano esattamente due, per cui li ha valutati entrambi.

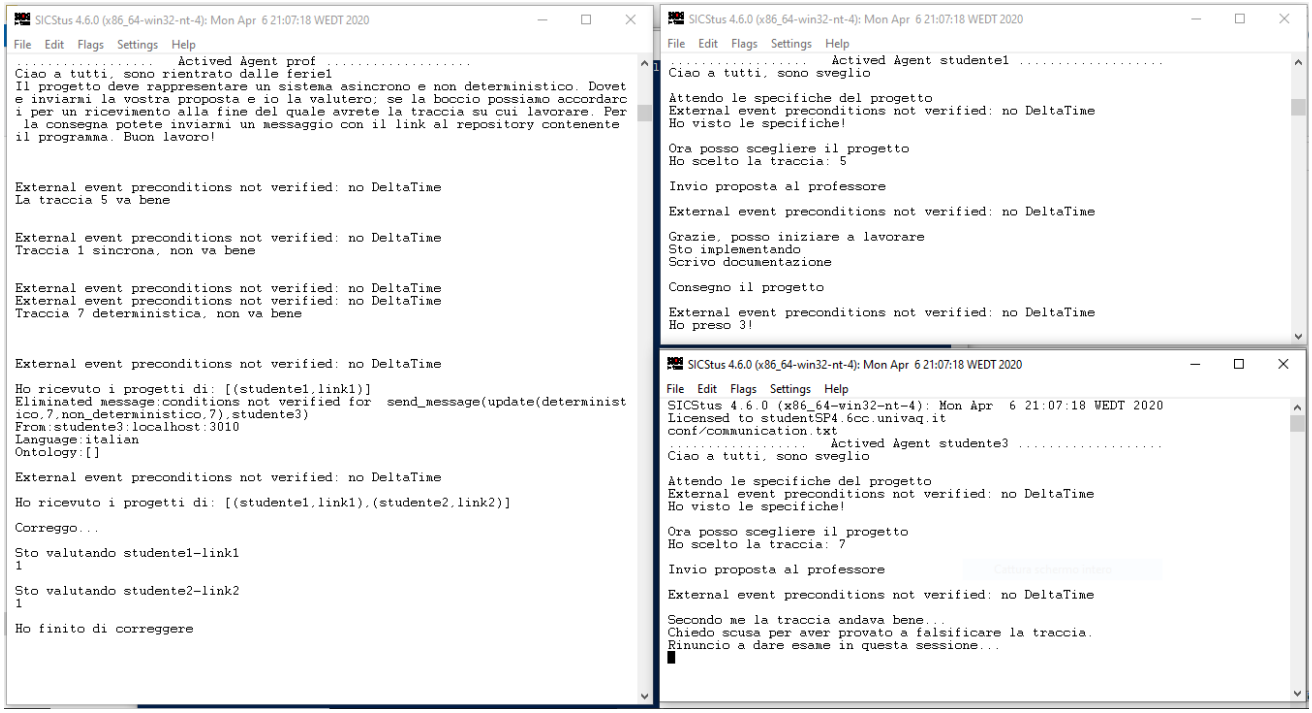


Figura (a)

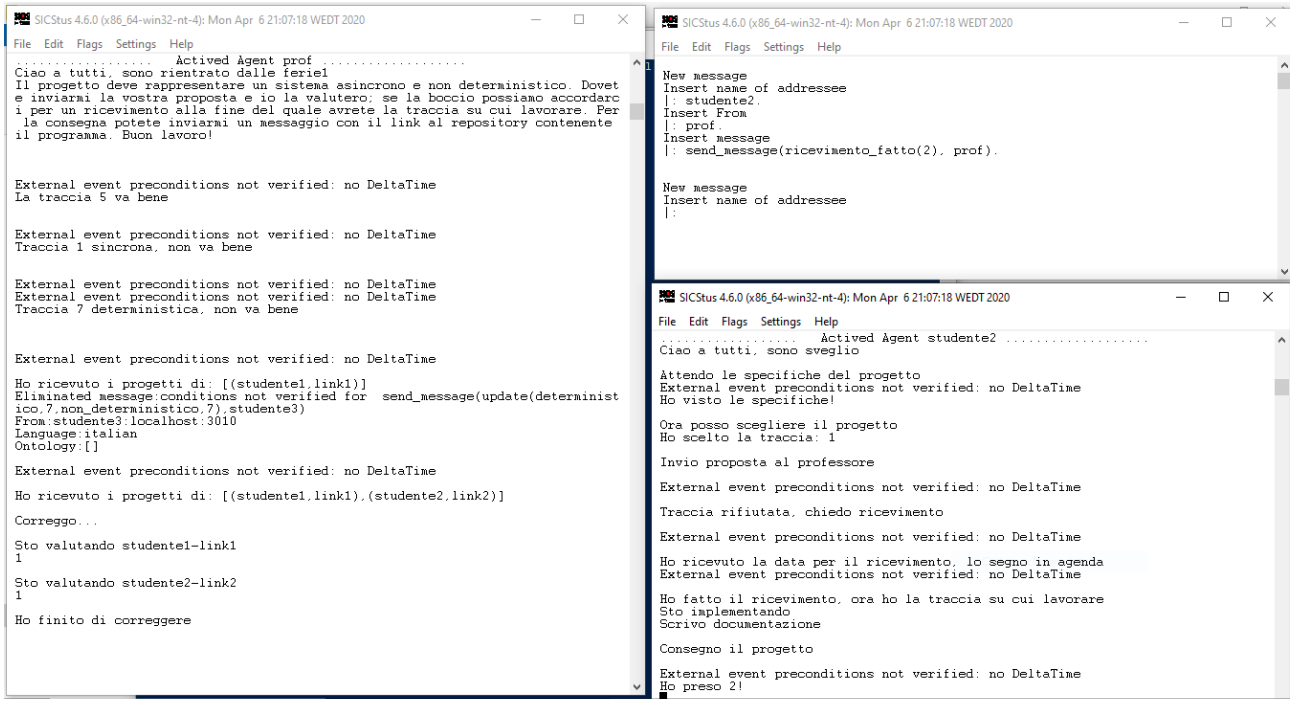


Figura (b)

Nell'esempio di esecuzione in Figura (c) è stato evidenziato quello che accade quando il professore sceglie di non pubblicare le specifiche appena sveglio. Tutti gli studenti restano in attesa e ogni 10 secondi controllano se sono state rese disponibili. Lo studente 1 (in alto a destra), che si è svegliato per primo, invia un messaggio per richiedere la pubblicazione delle specifiche e il professore procede subito. Gli altri studenti stavano ancora aspettando e ricevono la notifica prima dello scadere del tempo, per cui non inviano nulla.

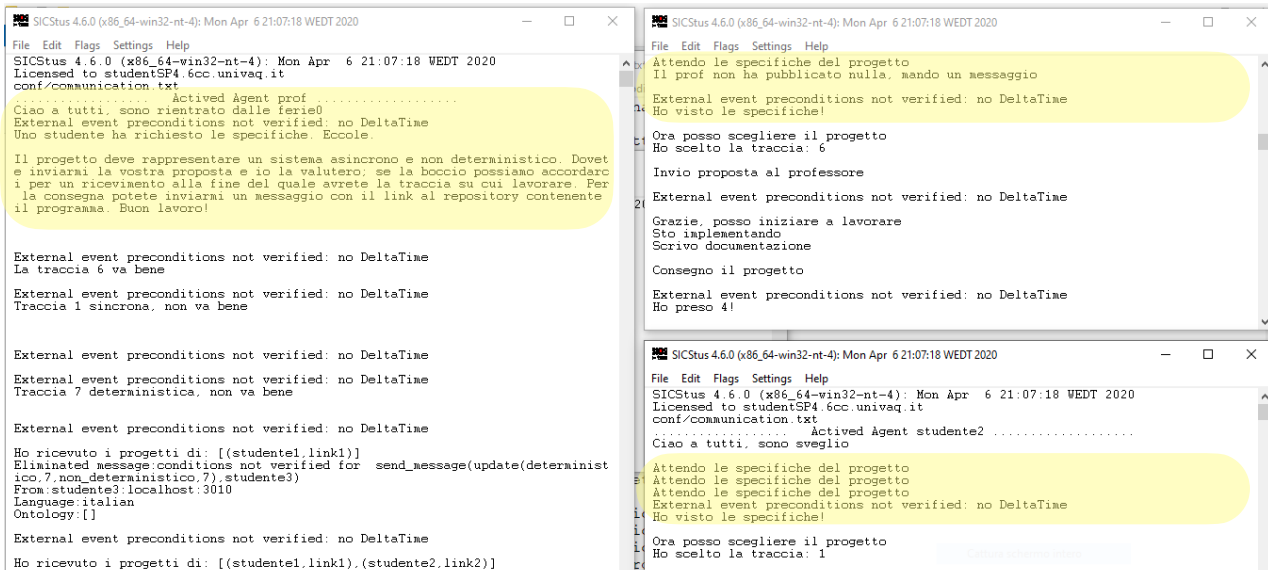


Figura (c)