Università di Padova

Facoltà di Scienze MM. FF. NN.
Dipartimento di Matematica Pura ed Applicata

Anno accademico 2006/2007

Laurea Magistrale in Informatica

Corso di Sistemi Concorrenti e Distribuiti

Progetto “Railway 2007”

Docente: Tullio Vardanega

Studente: Paolo Santi

27 Aprile 2007

Rielaborazione di un progetto di Alessandro Temil - 2006

Indice generale

Introduzione

Scopo del documento

Il documento presenta il lavoro “Railway”, un progetto didattico di programmazione di un sistema concorrente e distribuito.
Verranno esaminate le problematiche e le soluzioni adottate in riferimento ai Requisiti di Progetto illustrati nel documento Progetto.html presente nelle pagine del Corso di Sistemi Concorrenti e Distribuiti dell'anno accademico 2006/2007 tenuto dal prof. Tullio Vardanega.
Verranno date inoltre le informazioni necessarie alla compilazione ed esecuzione del software.

Scopo del prodotto

Il prodotto finale è un software rilasciato in formato sorgente ed ha lo scopo di sperimentare, applicare e valutare i concetti di programmazione concorrente e distribuita esposti nel corso di studi ad un contesto applicativo di esempio.
Il prodotto attuale si basa su uno analogo sviluppato lo scorso anno accademico nello stesso corso di Sistemi Concorrenti e Distribuiti dallo studente Alessandro Temil.

Presentazione del progetto

Linguaggio, strumenti di sviluppo, ambiente di esecuzione

Il prodotto è interamente sviluppato in linguaggio Ada utilizzando:

- il sistema di sviluppo GPS di AdaCore per l'ambiente di produzione ed esecuzione non distribuito,
- il compilatore gnatdist per la costruzione del programma in versione distribuita.
-

Oltre alla configurazione standard è stata utilizzata la libreria:

- gtkada per la gestione delle finestre Gtk+.

Il sistema di sviluppo e le librerie utilizzate sono gli stessi necessari per lo sviluppo del prodotto precedente.

Per la documentazione sono stati utilizzati:

- OpenOffice 2.0 per la scrittura dei testi;
- Umbrello UML designer per i diagrammi.

Il software è stato sviluppato e collaudato usando un PC con sistema operativo GNU-Linux; strumenti analoghi sono comunque disponibili anche per altre piattaforme Hardware e Software.

Progetto didattico (estratto dalle pagine del corso)

Per l'anno accademico 2006/7 il progetto didattico sarà sviluppato a partire dal risultato raggiunto dal prodotto miglior classificato tra quelli realizzati nel corso dell'anno accademico precedente, e dovrà soddisfare almeno uno dei tre requisiti sotto elencati.
Il prodotto iniziale realizza un sistema ferroviario complesso. [. . .]
Il progetto dovrà soddisfare almeno uno dei seguenti requisiti:

- **Obiettivo A** Riprodurre *esattamente* la semantica concorrente del prodotto di partenza e il suo paradigma di distribuzione utilizzando un linguaggio concorrente diverso da quello in esso impiegato.
- **Obiettivo B** Progettare un paradigma di distribuzione diverso e più ambizioso di quello utilizzato dal prodotto di partenza e realizzarlo, senza violare i requisiti funzionali del sistema iniziale, impiegando un linguaggio di programmazione a scelta.
- **Obiettivo C** Realizzare le funzionalità di un sistema *architetturalmente* di tipo A o B utilizzando però linguaggi di programmazione distinti ed eterogenei e le corrispondenti incarnazioni di CORBA.

Caratteristiche del prodotto di partenza

Il progetto di partenza, realizzato lo scorso anno accademico, implementa la simulazione di un sistema ferroviario complesso. Il sistema di sviluppo usato si basa su GPS di AdaCore e gnatdist, la libreria per programmazione distribuita Glade e GtkAda per la gestione dell'interfaccia grafica a finestre.
Per quanto riguarda la struttura progettuale e la realizzazione il lavoro di Temil è senza dubbio molto valido e interessante.
Dal punto di vista della distribuzione il sistema è diviso in tre partizioni:

- simulation,
- central_control (rci) – visualizzazione stato attuale passeggeri e treni,
- notice_boards (rci) – visualizzazione (storico) stato treni ad ogni stazione.

Volendosi riferire al noto paradigma MVC, di queste partizioni la prima implementa “Model”, la seconda e la terza “View”. Dal momento che l'utente non può interagire con il sistema la parte

“Controller” non è implementata.
Nel software analizzato lo stato di tutto il sistema è mantenuto per intero nella partizione main, mentre central_control e notice_boards lo visualizzano. Dallo stato ad ogni istante della partizione simulation determiniamo lo stato di ogni entità attiva e passiva del sistema. Da questo possiamo dire che la “quantità di ditribuzione” è molto bassa; una caratteristica che dovrebbe essere propria di un sistema distribuito è quella di mantenere lo stato del sistema suddiviso in più di una partizione.

Caratteristiche del prodotto realizzato

Il software sviluppato intende realizzare l'Obiettivo B sopra esposto cercando quindi di distribuire maggiormente lo stato e i compiti nell'intero sistema.
Partendo da quanto già disponibile, ho ritenuto utile mantenere in parte quanto già realizzato cercando di operare più che altro un lavoro di separazione ed interfaccia piuttosto che partire da zero e ricostruire un prodotto simile.
Analizzando le parti distinguibili all'interno del sistema preesistente ho ritenuto opportuno suddividerlo in più partizioni mantenendo totalmente intatta l'interfaccia grafica.
L'assetto definitivo è così definito:

- simulation – contiene solo main (normal),
- central_control - distrailway.central_control (rci) – visualizzazione stato attuale passeggeri e treni – invariato rispetto al prodotto precedente,
- notice_boards - distrailway.notice_boards (rci) – visualizzazione (storico) stato treni ad ogni stazione – invariato rispetto al prodotto precedente,
- net_map – distrailway.net_map (rci) – visualizzazione mappa e posizione dei treni,
- tracks – railway.tracks (rci) – gestione delle tratte di collegamento tra stazioni,
- stations – railway.stations (rci) – gestione delle stazioni (binari e biglietterie),
- trains – railway.trains (rci) – implementa i task relativi ai treni,
- travelers – railway.travelers (rci) – implementa i task relativi ai passeggeri.

Formalismi adottati

Come il prodotto di riferimento, anche in questo documento il sistema viene descritto utilizzando la metodologia HRT-UML.
I tipi di entità definiti sono i seguenti:

- entità attive le quali, con un proprio thread ed esterne al modello, il loro comportamento non fa parte e non interferisce con il modello;
- entità cicliche costituite come thread non terminanti che si ripetono nel tempo;
- entità sporadiche le quali sono dei thread non terminanti attivati da qualche evento come ad esempio un interrupt hardware;
- entità protette le quali modellano una risorsa di sistema con un protocollo con garanzia di mutua esclusione nell'accesso alla risorsa stessa;
- entità passive cioè risorse senza garanzia di mutua esclusione.

Descrizione del sistema

Il software prodotto simula, con opportune semplificazioni, una rete ferroviaria “complessa”.
L'ambiente generato è prima di tutto popolato dalle entità attive:

- treni;
- passeggeri.

Questi, per agire hanno bisogno di una infrastruttura di collegamento e servizi composta di entità passive:

- stazioni: biglietterie (Ticket Office) e marciapiedi di attesa ai binari (Platform);
- binari: tratte di collegamento (Segment) e binari (Track).

Treni

I treni circolano su un percorso di andata ritorno continuo. Quando arrivano al capolinea ritornano indietro seguendo un percorso analogo a ritroso A – B – C – D – C – B – A ecc...
Del percorso sono definiti la stazione e marciapiede di partenza e tratta percorsa per arrivare alla prossima destinazione. Il percorso comprende un anello, in senso logico, che può differenziarsi per i marciapiedi di fermata. In realtà nulla vieta che il percorso di andata e di ritorno siano effettivamente diversi purché l'ultimo stadio del percorso si colleghi con il primo.

> NOTA: per la simulazione sarebbe più realistico, e anche più semplice, che ogni treno percorresse una tratta dall'inizio alla fine e sempre e solo in un senso. Un altro treno dovrebbe percorrere il percorso inverso. Dal momento che i requisiti di progetto lo richiedono si è mantenuta la precedente struttura ciclica.

Le tipologie di treni sono due:

- Eurocity (EC) – treni con prenotazione obbligatoria e alta priorità;
- Interregionali (RT) – treni senza prenotazione e bassa priorità.

Se un passeggero vuole utilizzare un Eurocity deve prenotare il biglietto prima di salirvi. Per gli Interregionali deve acquistare un biglietto e portarsi al marciapiede di attesa.
Nella richiesta di risorse, Binari o Marciapiedi, gli Eurocity hanno la precedenza sugli Interregionali.
Il percorso di un treno (Route) è suddiviso in tappe (Stage).

- Uno **Stage** è formato dalla coppia Marciapiede di partenza e Segmento percorso per raggiungere la prossima destinazione (Platform, Track). La Platform di arrivo è quella di partenza del prossimo Stage.
- Il percorso, **Route**, è una lista ordinata di almeno due Stage che formano un anello chiuso. Dopo l'ultimo Stage del percorso il treno ricomincia dal primo.

I treni sono definiti come entità attive cicliche che nel linguaggio adottato sono implementati da **task**.
Nel sistema, come nella realtà, ogni treno è identificato da un codice numerico univoco.

Passeggeri

I passeggeri hanno un proprio percorso prefissato di andata e ritorno per cui sono stabiliti i treni da utilizzare e le stazioni di salita e discesa. Il viaggio può comprendere uno o più treni con cambio intermedio.
Alla partenza i passeggeri si presentano alla biglietteria, quella da cui inizia il viaggio, e acquistano i biglietti necessari per tutti i treni che intendono utilizzare.
Se i treni sono tutti interregionali all'acquisto del biglietto corrisponde un immediato accodamento al primo marciapiede di attesa.
Se nel percorso vi è una o più tratte da percorrere con treni Eurocity vengono prenotati i posti per tutti i segmenti necessari. Solo quando tutti i posti saranno disponibili il biglietto verrà rilasciato e il passeggero inizierà il suo viaggio.
Anche i passeggeri, entità attive cicliche, sono modellati come task.

Stazioni

Le stazioni sono composte logicamente da:

- biglietterie;
- marciapiedi di attesa.

Biglietterie: sono entità protette e i passeggeri vi accedono in mutua esclusione.
Queste accedono ad altre entità, i database dei biglietti, che forniscono le funzioni di prenotazione per i treni EC con prenotazione obbligatoria.

Marciapiedi: sono entità protette senza flusso di controllo con protocollo di accesso che dipende dal loro stato interno. E' una risorsa acceduta in mutua esclusione dai treni e dai passeggeri.
I passeggeri vengono accodati in attesa dell'arrivo del treno desiderato per discesa o salita. Il treno che arriva cambia lo stato del marciapiede facendo scendere e salire i passeggeri.

Problematiche relative alla prenotazione dei biglietti

La prenotazione dei biglietti può comportare problemi legati alla prenotazione simultanea da parte di più viaggiatori.
Supponiamo che A voglia acquistare un biglietto per le tratte x, y, z e B debba acquistare un biglietto per w, z, y. supponiamo (Murphy e Lupo Alberto lo giustificano) che vi sia un solo posto per le tratte z e y.
All'istante t0 A prenota x,y e B prenota w e z.
All'istante t1 A vorrebbe prenotare un posto per la tratta z ma l'ultimo posto l'ha avuto B, B desidera il posto nel tratto y ma è già in possesso di A.
In questo caso nessuno dei dei due può ottenere il posto per tutte le tratte interessate e quindi non partiranno mai attendendo all'infinito alle rispettive biglietterie di prenotazione.
Per evitare questa condizione di stallo si è scelto di eseguire le prenotazioni in un ordine predefinito:

- in ordine di treno (dal codice minore al maggiore);
- in ordine di tratta.

Tabelloni
Il sistema presenta la situazione istante per istante relativa a:

- stazioni – un tabellone per ogni stazione indicante i treni in transito nelle piattaforme;
- treni – un tabellone complessivo per tutti i treni che riporta posizione del treno e numero di passeggeri a bordo;
- passeggeri – un tabellone per tutti i passeggeri posizione attuale.

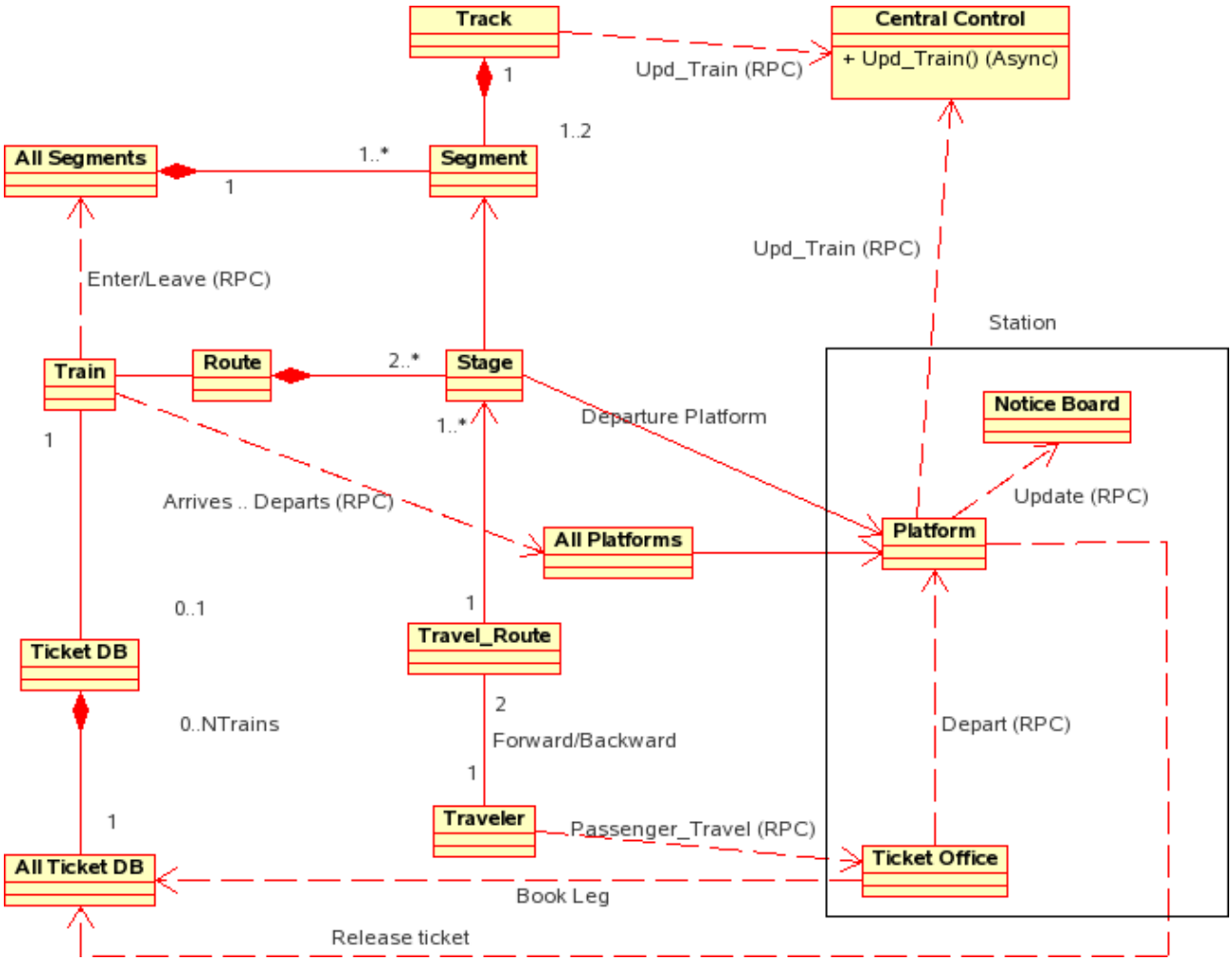
Tratte e Binari

I binari di collegamento tra le stazioni sono modellati come risorse protette condivise dai treni. Vi sono collegamenti a singolo (bidirezionale) e doppio binario (unidirezionale).
Se il binario è singolo vi può transitare solo un treno in una direzione o dall'altra.
Ad esempio, se il treno Ta deve percorrere la tratta x-y e il treno Tb la tratta y-x solo uno dei due ne può usufruire in un dato istante. La tratta x-y (o y-x che è lo stesso) viene assegnata al primo treno che lo richiede. Se le richieste sono simultanee un treno Eurocity ha priorità su un Interregionale.
Se il binario è doppio i due ipotetici treni possono percorrere la tratta (in senso inverso) simultaneamente; Ta otterrà il possesso della tratta x-y, Tb di quella y-x che sono di fatto distinte.

Architettura del sistema

Entità coinvolte e loro relazioni

Nella figura seguente è riprodotta la struttura a blocchi in notazione UML degli elementi componenti il software realizzato.

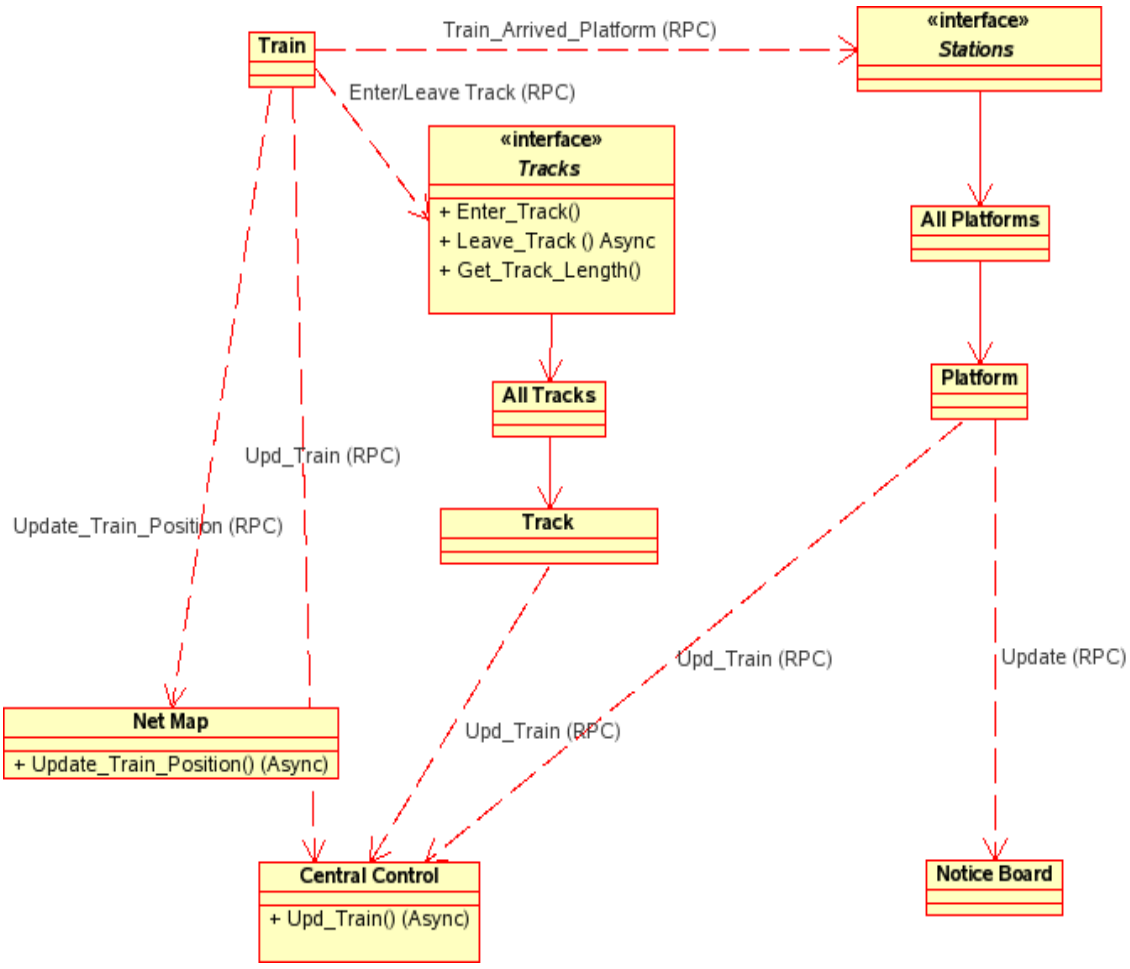


Sono rappresentate tutte le entità attive e passive fin qui citate più altre che rappresentano elementi di strutturazione o di gestione degli elementi a livello inferiore.
In particolare notiamo:

- All Segments – insieme di tutti i segmenti di collegamento
- All Ticket DB – insieme di tutti i database di booking (solo per treni Eurocity)
- Route – percorso di un treno ovvero l'insieme dei segmenti che lo formano
- Travel Route – percorso di un viaggiatore

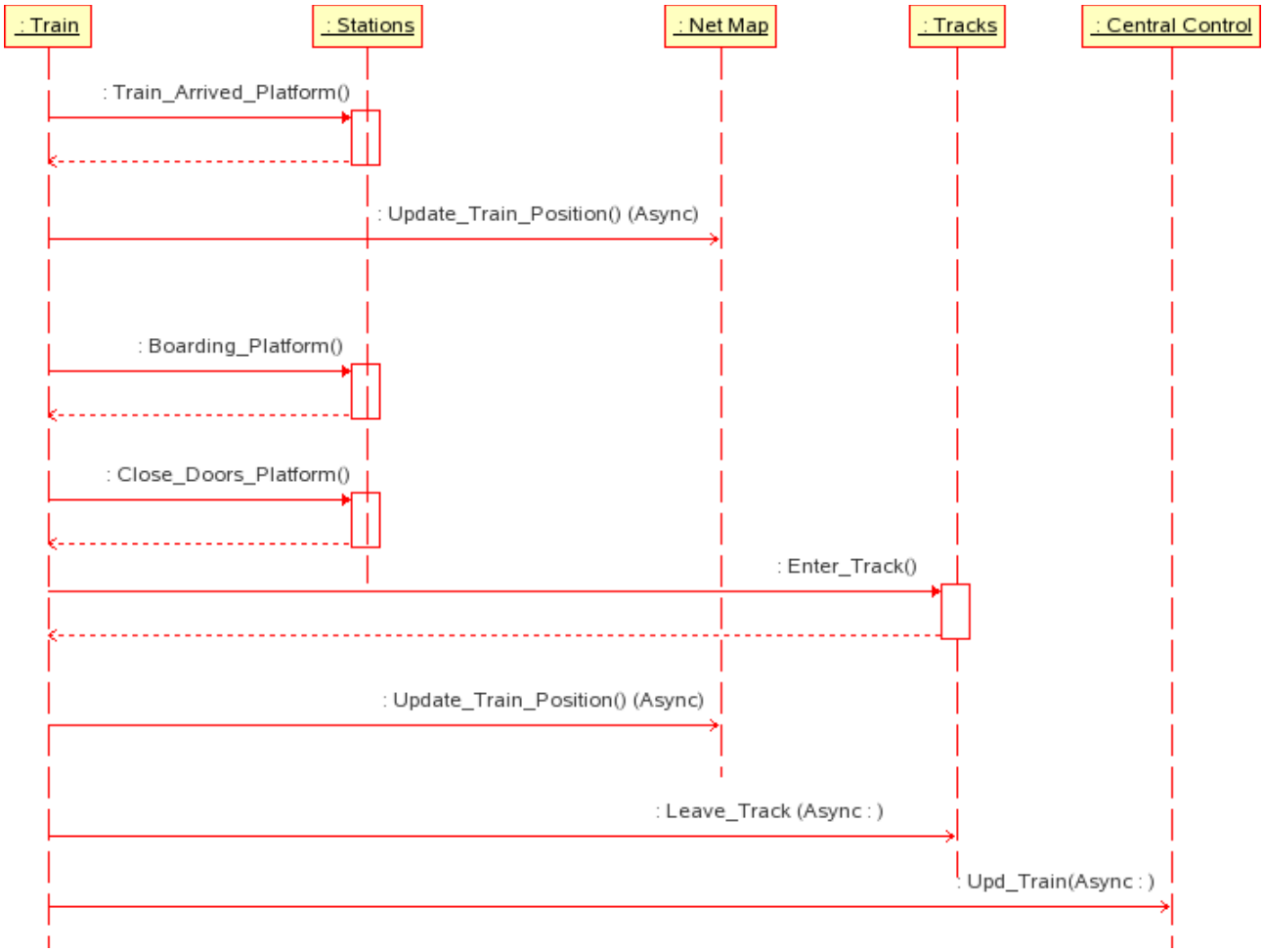
Treni e relazioni con il sistema

Il diagramma mostra le relazioni di dipendenza e uso tra le entità treni e quelle da essei utilizzate.



E' importante notare che l'entità Train non interagisce direttamente con Track né con Platform. Per distribuire su partizioni distinte i treni e le entità utilizzate si sono introdotte delle procedure di interfaccia per accedere via Remote Procedure Call (RPC) qui rappresentate da Stations e Tracks. Nel progetto attuale tutti i task train sono definiti in un'unica partizione Trains. Nulla toglie, comunque di poter suddividere i treni in più partizioni, fino anche ad avere un solo task train per ogni partizione.

Diagramma di sequenza relativo all'esecuzione del task train.



Passeggeri e entità coinvolte

Anche i passeggeri, come i treni, interagiscono con il sistema permezzo delle procedure di Stations.

In particolare ad ogni viaggio di andata e ritorno, i passeggeri richiamano la funzione Passenger_Travel() per effettuare il viaggio di andata o ritorno rispettivamente.

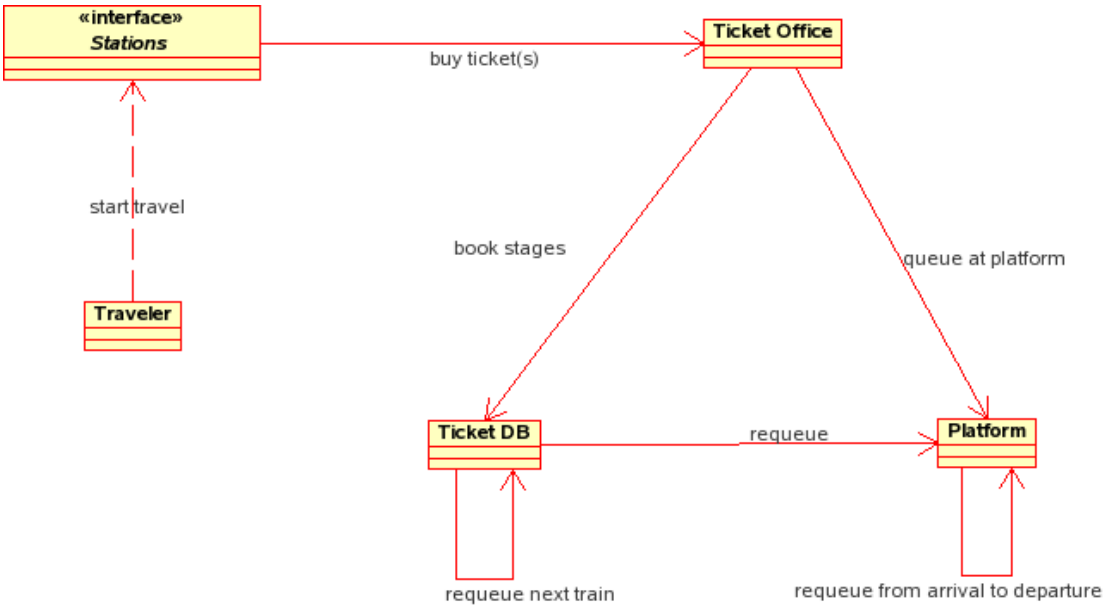
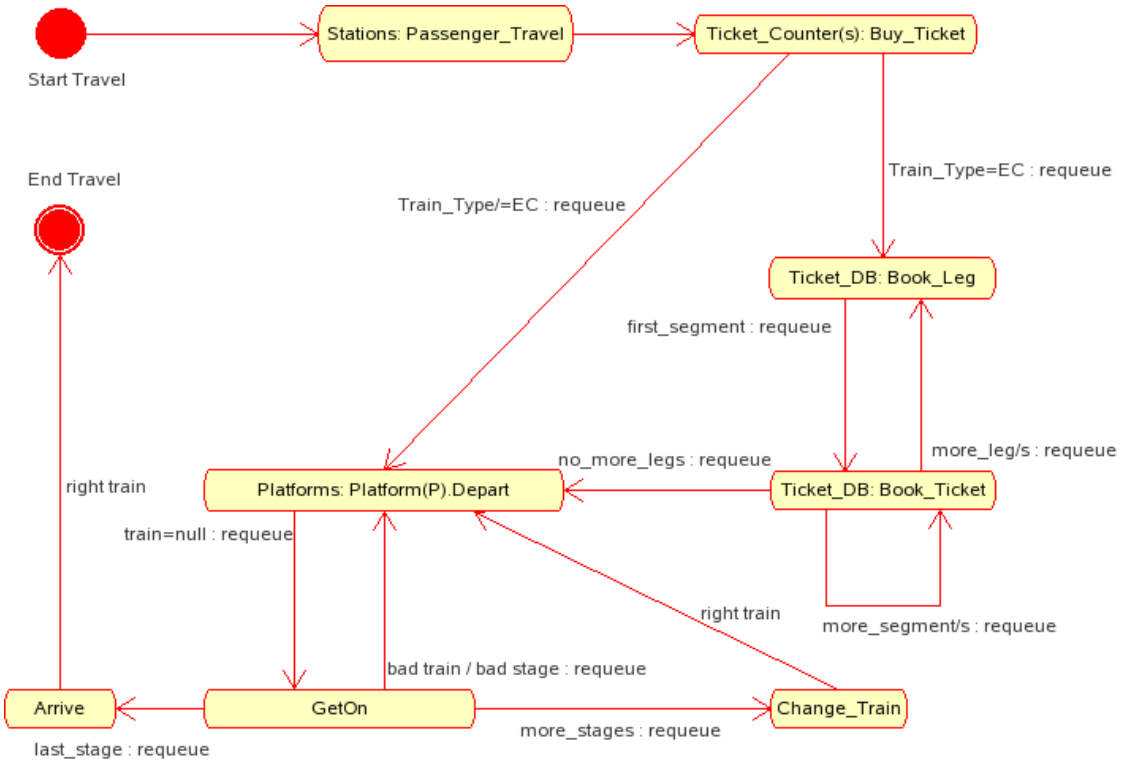


Diagramma degli stati di traveler durante un viaggio.

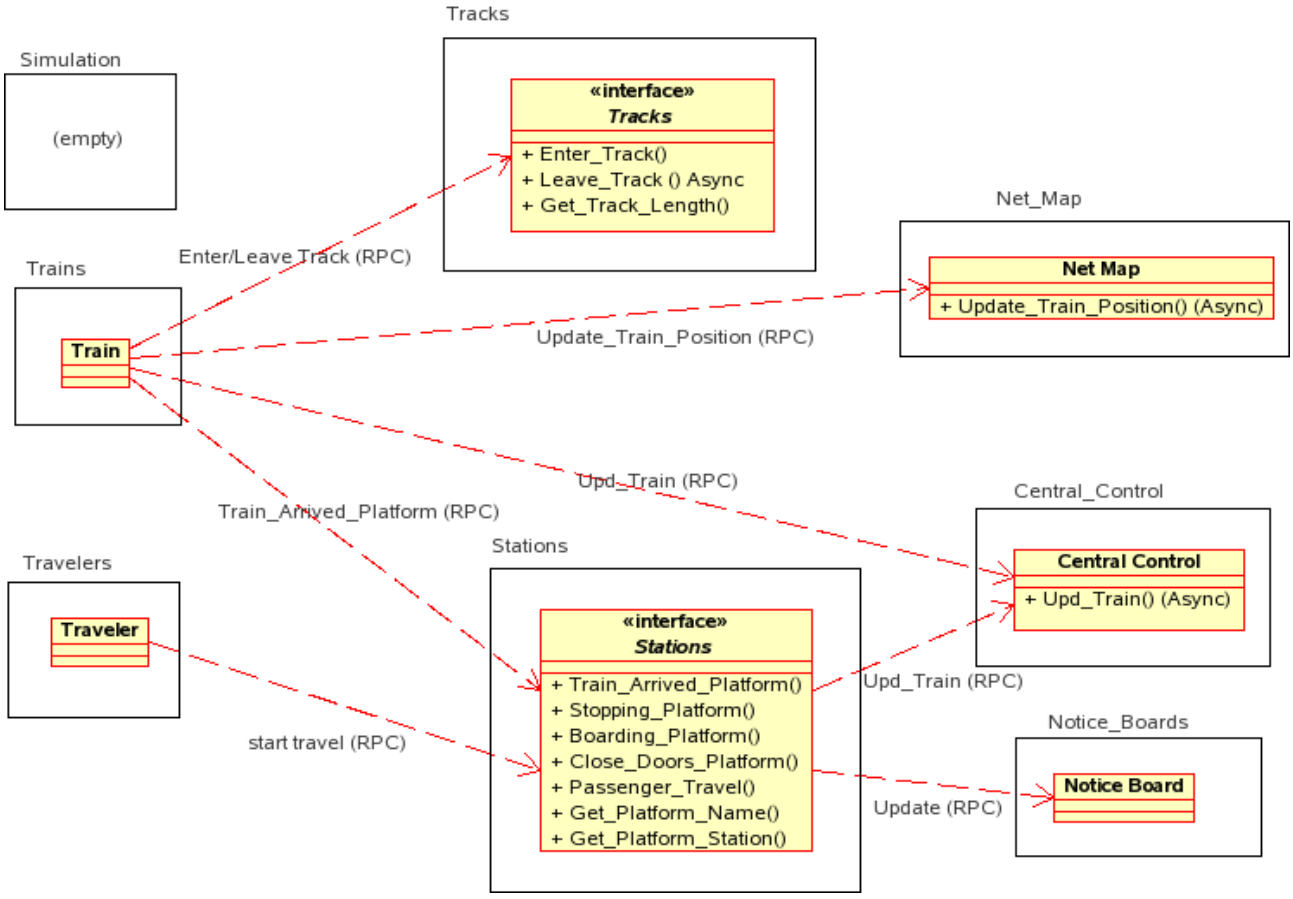


Architettura di distribuzione del sistema

Il sistema distribuito è suddiviso in otto partizioni distinte.

- Simulation
- Track
- Stations
- Central_Control
- Notice_Boards
- Net_Map
- Trains
- Travelers

Nella seguente figura sono rappresentate graficamente le partizioni del sistema, le principali entità ad esse afferenti e le loro relazioni di dipendenza.



Per illustrarne l'impostazione riportiamo il file di configurazione del sistema distribuito distributed_railway_2007.cfg.

```
configuration Distributed_Railway_2007 is
   pragma Starter (none);
   Simulation      : Partition := ();
   Tracks          : Partition := (Railway.Tracks);
   Stations        : Partition := (Railway.Stations);
   Central_Control : Partition := (Distrailway.Central_Control);
   Notice_Boards   : Partition := (Distrailway.Notice_Boards);
   Net_Map         : Partition := (Distrailway.Net_Map);
   Trains          : Partition := (Railway.Trains);
   Travelers       : Partition := (Railway.Travelers);
```

```
   for Partition'Directory use "bin";
   procedure Main is in Simulation;
end Distributed_Railway_2007;
```

La partizione Simulation è vuota; contiene solamente la procedura main che ha corpo null. Questa paradossale situazione è stata evidenziata per mostrare l'effettiva indipendenza del modello dalla procedura principale che qui non fa niente se non esistere.

Installazione e compilazione del software

E' possibile utilizzare il programma con l'ambiente di sviluppo GPS - GNAT o con il compilatore per programmi distribuiti gnatdist.

- Per utilizzare il prodotto con GPS (NON distribuito) è necessario installare:
  - ambiente integrato GPS
  - libreria gtkada
- Per utilizzare il prodotto con gnatdist (distribuito) è necessario installare, oltre ai prodotti precedenti, anche:
  - gnatdist

- Per utilizzarlo con GPS (NON distribuito) aprire il file progetto Railway2007.gpr
  - E' probabile che il file progetto debba essere modificato a mano per fornire le corrette impostazioni al compilatore “for Default_Switches ("ada") use . . .”
  - Per ottenere le opzioni del sistema in uso lanciare ilprogramma gtkada-config da una finestra di shell.
  - Volendo ottenere le impostazioni desiderate racchiuse tra virgolette e separate con la virgola è possibile utilizzare il comando:
    `gtkada-config | awk '{ ORS=" "; for(i=1;i<=NF;i++) print "\""$i"\"," };' - > tmp.txt`
    Questo inserisce nel file tmp.txt le opzioni da copiare e incollare nei Default_Switches del file di configurazione (togliere l'ultima virgola).
  - Nel file di configurazione sono presenti già altre impostazioni commentate.
  - Compilare con il comando
    Build/Make/main.adb
  - Eseguire con
    Build/Run/Main

- Per compilare il programma distribuito con gnatdist
  - Aprire una finestra di shell nella cartella obj.
  - Per compilare dare il comando:
    make
  - Lanciare il programma in forma distribuita con:
    ./demo.sh <nomehost> <porta>
    Esempio
    ./demo.sh localhost 5000
    Premere invio nella finestra shell per terminare.