Relazione finale	11/10/09
Documento formale – esterno – v3.0	relazione_v3.pdf

Redazione:

Daniele Bonaldo,
Alberto Zatton

Revisione:

Daniele Bonaldo,
Alberto Zatton

Lista di distribuzione:

Prof. Vardanega Tullio,
Daniele Bonaldo,
Alberto Zatton

Registro delle modifiche:

Versione	Data	Descrizione delle modifiche
0.1	09/06/2009	Prima stesura
0.2	31/07/2009	Aggiunta del paragrafo “Modello”
0.3	07/08/2009	Aggiunta del paragrafo “Concorrenza”
0.4	18/08/2009	Aggiunta del paragrafo “Distribuzione”
1.0	25/08/2009	Preparazione per la consegna
1.1	27/08/2009	Aggiunta descrizione degli incidenti casuali e dello script di avvio
2.0	27/08/2009	Preparazione per la seconda consegna
2.1	05/10/2009	Aggiunto il paragrafo “La variabile tempo”
3.0	11/10/2009	Modificati i paragrafi “Box”, “Nodi individuati”

Indice

1 Introduzione

1.1 Scopo del documento

Questo documento è la relazione finale per il progetto di Sistemi Correnti e Distribuiti realizzato degli studenti Bonaldo Daniele e Zatton Alberto nell'anno accademico 2008/09.

Il documento ha la finalità di illustrare l'analisi svolta e le scelte fatte in termini di progettazione e realizzazione del progetto.

1.2 Struttura del documento

Il documento è diviso nelle seguenti parti

- **Introduzione,** in cui viene descritto sommariamente lo scopo e la struttura del documento e l'ambito del progetto.
- **Analisi**, in cui vengono riassunti i requisiti individuati e il lavoro di analisi svolto.
- **Modello**, in cui viene descritto il modello utilizzato per la simulazione della gara.
- **Concorrenza**, in cui vengono esposte le scelte fatte nella progettazione e realizzazione della concorrenza all'interno del progetto.
- **Distribuzione**, in cui vengono esposte le scelte fatte nella progettazione e realizzazione della distribuzione all'interno del progetto.
- **Implementazione software**, in cui vengono descritto il software realizzato per implementare il modello.
- **Compilazione ed esecuzione**, in cui viene descritta la fase di esecuzione del progetto.
- **Conclusioni**, in cui vengono trattate le conclusioni generali, includendo possibili sviluppi futuri del progetto e commentando eventuali scelte alternative a quelle effettuate.

1.3 Scopo del prodotto

Lo scopo del prodotto è quello di simulare una competizione sportiva assimilabile a quelle automobilistiche di Formula 1 utilizzando tecniche di concorrenza e distribuzione viste a lezione durante il corso di Sistemi Concorrenti e Distribuiti.

2 Analisi

In questa sezione viene riassunto il lavoro di analisi svolto.

L'analisi è stata svolta sulla descrizione del progetto e sulle clausole di partecipazione descritti alla pagina web

http://www.math.unipd.it/~tullio/SCD/2008/Progetto.html.

In seguito a tale analisi sono stati individuati i seguenti requisiti funzionali:

RF01	Presenza di un circuito
RF02	Il circuito deve essere dotato della pista e della corsia di rifornimento
RF03	La pista deve essere soggetta a regole realistiche di accesso
RF04	La pista deve essere soggetta a regole realistiche di condivisione
RF05	La pista deve essere soggetta a regole realistiche di tempo di percorrenza
RF06	La corsia di rifornimento deve essere soggetta a regole realistiche di accesso
RF07	La corsia di rifornimento deve essere soggetta a regole realistiche di condivisione
RF08	La corsia di rifornimento deve essere soggetta a regole realistiche di tempo di percorrenza
RF09	Presenza di un insieme configurabile di concorrenti
RF10	Ogni concorrente deve essere configurabile con caratteristiche specifiche di prestazioni, risorse e strategia di gara.
RF11	Deve essere presente un sistema di controllo
RF12	Il sistema di controllo deve poter visualizzare lo stato della competizione
RF13	Il sistema di controllo deve poter visualizzare lo stato dei vari concorrenti
RF14	Il sistema di controllo deve poter visualizzare le migliori prestazioni
RF15	Presenza di una competizione configurabile per quanto riguarda la durata
RF16	Controllo di terminazione dei concorrenti a fine gara
RF17	Deve essere presentata una relazione sul lavoro svolto
RF18	Il progetto deve presentare evidenti tracce di concorrenza
RF19	Il progetto deve presentare evidenti tracce di distribuzione

Sono stati inoltre individuati i seguenti requisiti opzionali:

RO01	Il circuito deve essere selezionabile in fase di configurazione
RO02	Deve essere presente una gestione del tempo meteorologico
RO03	I parametri della configurazione devono poter essere letti da file
RO04	Realizzare il progetto utilizzando almeno due linguaggi di programmazione

Come si potrà vedere anche in seguito, è stato scelto di svolgere il progetto attenendosi alle clausole di partecipazione di livello 3, sviluppandolo quindi utilizzando due linguaggi di programmazione.

3 Il Modello

Di seguito sarà descritto il modello utilizzato per simulare la competizione di Formula 1.

3.1 La Gara

La gara è definita da un circuito, i piloti partecipanti con relative vetture e il numero di giri che i piloti devono effettuare per terminare la competizione.

3.2 Il Circuito

Il circuito è diviso in:

- tracciato di gara;
- box.

Il primo è la parte del circuito nella quale avviene la gara vera e propria; il secondo serve ai piloti per accedere al proprio box ed effettuare il pit-stop con cambio gomme e rifornimento di carburante. Sono entrambi descritti nei paragrafi seguenti.

3.2.1 Tracciato di gara

Il circuito è descritto come una successione di segmenti: tali segmenti simulano un rettilineo od una curva. Ogni segmento ha quattro caratteristiche:

- velocità iniziale;
- velocità finale;
- numero di corsie;
- lunghezza;

La velocità iniziale rappresenta la velocità massima raggiungibile da una vettura nella prima parte del segmento; la velocità finale è la velocità massima che la vettura deve possedere alla fine del segmento. Chiameremo il punto di passaggio dalla prima velocità alla seconda *staccata*: essa varia da pilota a pilota a seconda delle caratteristiche personali. Approfondiremo ulteriormente l'argomento nel paragrafo 3.3 (I Piloti).

Non esiste una distinzione netta tra rettilineo e curva: il circuito è interamente

simulato fornendo le velocità sopra descritte ai vari segmenti. Tipicamente un rettilineo avrà velocità iniziale maggiore o uguale alla velocità finale (per accedere ad un segmento curva, il pilota frena alla fine del rettilineo); una curva avrà invece velocità iniziale minore o uguale alla velocità finale (superata la corda della curva, il pilota accelera).

Il numero di corsie rappresenta quante vetture possono marciare in parallelo nella stessa sezione di segmento. Il valore è stato limitato al massimo a due. In un segmento con una corsia il sorpasso non sarà permesso; in un segmento con due corsie il sorpasso sarà permesso.

Due vetture non possono essere allo stesso momento nella stessa porzione di circuito: a questo scopo, ogni segmento è diviso in *microsegmenti*. Le proprietà di un microsegmento sono:

- numero di corsie: ereditato dal segmento di cui fa parte;
- in ogni corsia del microsegmento è ammessa al massimo una vettura.

I microsegmenti verranno ripresi in seguito nel par. 4 (“Concorrenza”).

3.2.2 Box

I box sono modellati esattamente come il tracciato principale. Sono costituiti da un unico segmento con velocità iniziale uguale alla velocità finale: come da regolamento FIA la velocità è imposta a 100 Km/h massimi. Il numero di corsie dei box è ristretto a 1. La lunghezza del segmento è pari a quella del primo segmento del tracciato principale.

3.3 I Piloti

I piloti sono le entità attive del modello. Essi gareggiano per finire nel minor tempo possibile la gara, rispettando le restrizioni imposte dal circuito. Di seguito sono definite le loro caratteristiche:

- nome / cognome;
- scuderia di appartenenza;

- numero della vettura guidata;
- coefficiente di accelerazione;
- coefficiente di decelerazione;
- velocità massima raggiungibile dalla vettura;
- strategia.

Il coefficiente di accelerazione unisce la caratteristica accelerazione della vettura con l'abilità di guida del pilota. Maggiore è il coefficiente, maggiore sarà l'accelerazione a disposizione. Il coefficiente di accelerazione influisce anche sulla velocità massima raggiungibile nella prima parte di un segmento prima della staccata: un pilota abile raggiunge velocità superiori.

Il coefficiente di decelerazione unisce la caratteristica decelerazione della vettura con l'abilità di guida del pilota. Maggiore è il coefficiente, maggiore sarà la capacità frenante della vettura. Questo si traduce in frenate (staccate) ritardate (quindi, nel caso di rettilineo con frenata finale, una maggiore velocità media nel percorrere il segmento). Il coefficiente di decelerazione influisce anche sulla velocità finale del segmento: maggiore è il coefficiente, più alta sarà la velocità permessa in uscita dal segmento.

La velocità massima contraddistingue la più alta velocità raggiungibile dalla vettura, a prescindere dalla velocità massima del segmento in cui si trova.

La strategia indica al pilota quando fermarsi ai box per effettuare i cambi gomme e il rifornimento di carburante. E' decisa ad inizio gara e non è più modificabile.

3.4 Svolgimento della gara

Viene ora illustrato lo svolgimento della gara con l'interazione di tutti gli elementi visti nei paragrafi precedenti.

Prima dell'inizio della gara, i concorrenti si dispongono sulla griglia di partenza e attendono il segnale di avvio. La direzione di gara da quindi il via alla competizione.

Ogni pilota procede ad affrontare il numero di giri previsto del circuito. Nella prima parte di un segmento del circuito, il pilota cerca di raggiungere la massima velocità

consentita (velocità iniziale con modificatori). Quando lo ritiene opportuno (staccata), decelera per raggiungere una velocità utile per affrontare il segmento successivo (velocità finale con modificatori). Se incontra una vettura nel suo percorso e le condizioni lo permettono (velocità superiore alla vettura antistante e corsia libera) il pilota tenta il sorpasso. Se invece non è possibile sorpassare, rallenta alla velocità della vettura antistante, riprovando il sorpasso non appena le condizioni tornano favorevoli. Al completamento di un giro, se previsto dalla strategia, il pilota si ferma ai box; il pit-stop viene eseguito effettuando un drive-through nella corsia dei box.

Il pilota avanza di un microsegmento alla volta. Incontrare una vettura nel percorso significa che tale vettura occupa una corsia del microsegmento successivo al proprio. Se il microsegmento successivo ha una corsia libera, il pilota può avanzare verso questa corsia (tentato sorpasso); altrimenti deve rallentare in attesa che si liberi il microsegmento.

Nel momento in cui tutti i piloti hanno effettuato i giri previsti, la gara termina.

3.5 Accelerazione/Decelerazione

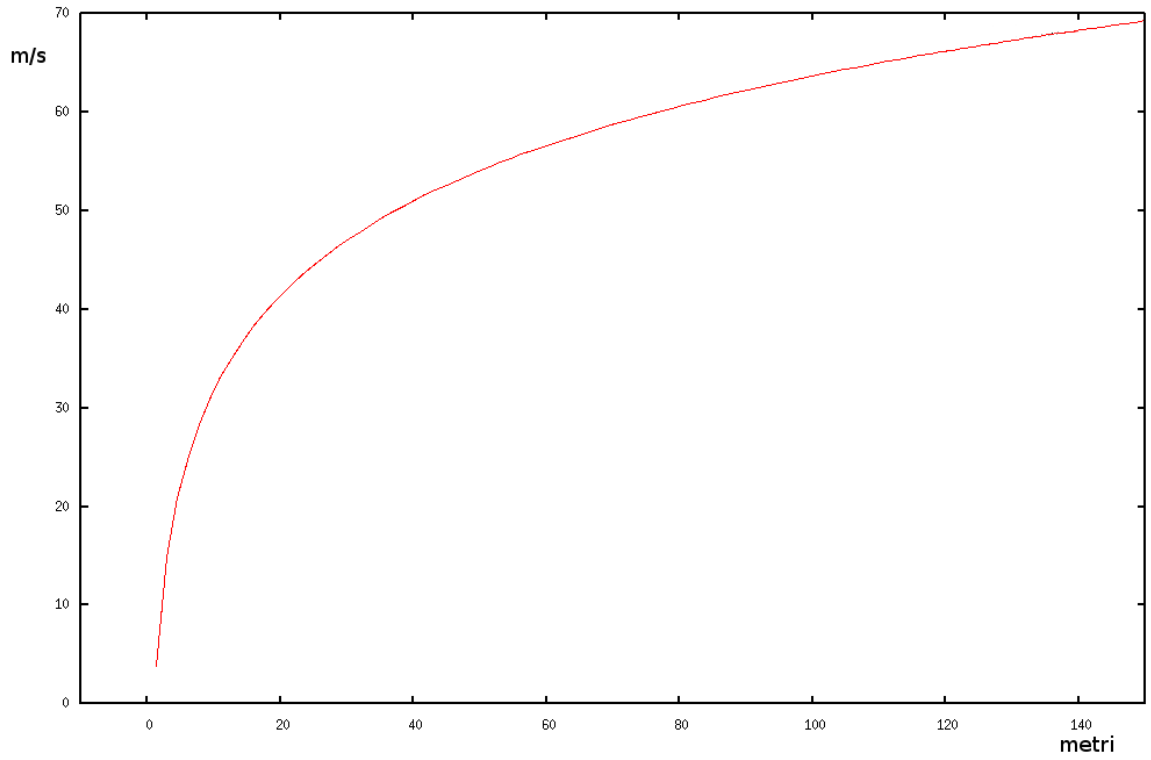
Una monoposto in accelerazione su rettilineo ha un'accelerazione media di 1,45 g (14,25 m/s^2), con un picco durante la fase di accelerazione intermedia e valori più blandi nella fase finale. I valori riportati in tabella si riferiscono ad una monoposto Renault R26 del 2006 (rif. Bibliografia, #1):

Secondi dalla partenza	Velocità raggiunta
1,7	100 Km/h
3,8	200 Km/h
8,6	300 Km/h

Data la complessità di una reale curva di accelerazione, si è scelto di approssimare tale curva con una funzione logaritmo. Inoltre si è scelto di far avanzare le vetture per segmenti: questo implica che la velocità non varia a seconda del tempo ma a seconda dello spazio percorso. La funzione risultante è stata perciò:

$$y = log_{1.075}\, x$$

dove *x* sono i metri percorsi dall'inizio dell'accelerazione mentre *y* è la velocità in m/s raggiunta in seguito a tale accelerazione. La funzione precedente è rappresentata graficamente come segue:



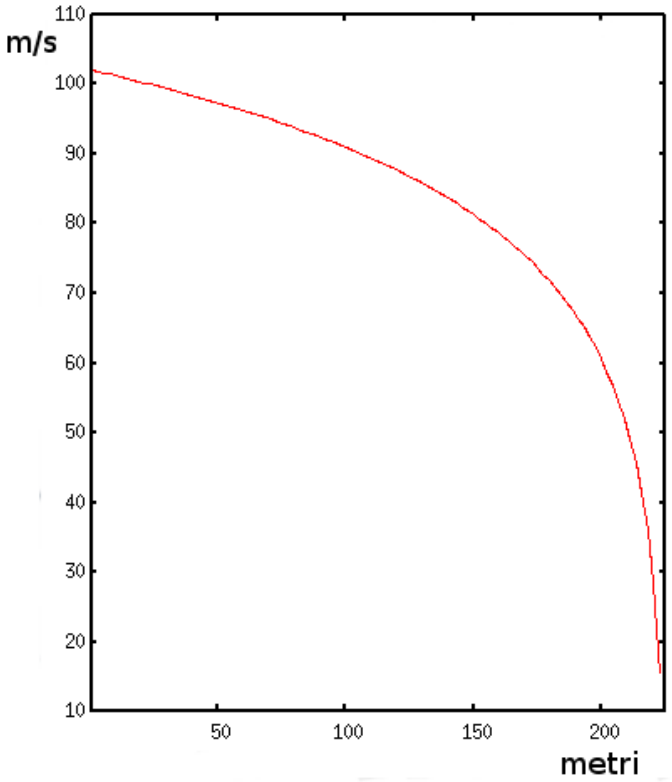
La decelerazione è stata modellata secondo gli stessi principi esposti per l'accelerazione. La decelerazione media in F1 è di 4 g (39 m/s^2) (rif. Bibliografia, #1).

Anche la funzione di decelerazione è stata approssimata con una funzione logaritmica ed espressa in m/s su metri di decelerazione:

$$y = log_{1.0545}\ (1.0545^{velocità\_max} - x)$$

dove *x* sono i metri percorsi dall'inizio della frenata, *y* è la velocità risultante da tale decelerazione e *velocità_max* è la velocità massima raggiungibile dalla vettura del

pilota. Una rappresentazione grafica di tale funzione è la seguente:



3.6 Eventi casuali

Nelle competizioni reali gli incidenti sono tutt'altro che rari e possono essere causati da vari fattori (errori dei piloti, cedimenti strutturali della vettura).

Durante la gara possono verificarsi perciò gli eventi elencati di seguito. Per ogni evento, è fissata la probabilità che esso avvenga:

- Rottura del motore (Probabilità: 0,001%);
- Scoppio di uno pneumatico (Probabilità: 0,001%);
- Problemi ai box (Probabilità: 0,001%);
- Scontro con un'altra auto (Probabilità: 0,1%);

Il verificarsi di uno qualsiasi di questi eventi comporta la fine della gara del pilota.

3.7 Restrizioni del modello conosciute

La simulazione descritta dal modello è restrittiva ed esclude molte caratteristiche di una gara di F1 vera e propria. Citiamo alcune tra le più importanti:

- Il set-up della vettura: ad ogni Gran Premio, alla vettura sono effettuate delle modifiche per meglio adattarsi alla gara (gomme, carico aerodinamico, assetto, …).
- Le condizioni meteorologiche: importanti in una vera competizione perché influenzano il comportamento della vettura. Inoltre sono soggette a cambiamenti durante la gara: di conseguenza si potrebbe rendere necessario un cambio di strategia.
- Il pit-stop: può determinare la vittoria o la sconfitta in una competizione. Può essere usato strategicamente per superare gli avversari senza dover lottare in gara.
- Comportamento della vettura (fisica): è influenzato da una moltitudine di fattori. Ad esempio nel modello sono state prese in considerazione solo l'accelerazione/decelerazione piana e non le forze centrifuga/centripeta che entrano in gioco in curva.

4 Concorrenza

Nel modello proposto per la simulazione di una gara di F1 è immediatamente riconoscibile la competizione tra i piloti per accedere ai microsegmenti del circuito (e quindi, avanzare).

Analizzando il problema, si riconoscono le seguenti entità:

- Entità attive: i piloti, i quali concorrono per accedere ai microsegmenti;
- Entità reattive: i microsegmenti.

E' necessario avere mutua esclusione nel possesso di ogni corsia del microsegmento. Inoltre è necessario garantire che le richieste di accesso alle corsie siano soddisfatte nell'ordine in cui esse sono state effettuate, in modo da evitare "sorpassi" casuali. Nel caso di microsegmento a corsia unica, queste sono le uniche richieste da soddisfare. Per quanto riguarda il primo requisito, quindi, un semplice semaforo per ogni microsegmento sarebbe sufficiente a garantire mutua esclusione sulla corsia, pur conoscendo i limiti di tale pratica (controllo della mutua esclusione troppo dipendente dalla disciplina del programmatore). D'altro canto, in caso di accodamento di chiamate in attesa, un semplice semaforo non garantisce una gestione della coda con logica FIFO.

Nel caso di microsegmento a doppia corsia inoltre è necessario gestire entrambe le corsie in modo da garantire l'accesso ad una o all'altra corsia se una sola è occupata.

Questo porta all'adozione di una struttura più complessa per la modellazione dei microsegmenti. Si è deciso di adottare le risorse protette in quanto soddisfano tutti i requisiti necessari ed offrono grande potenza espressiva. Nel paragrafo successivo è presentata la soluzione elaborata.

4.1 Implementazione dei microsegmenti

Ogni microsegmento viene implementato come risorsa protetta con le seguenti caratteristiche:

- due canali interni privati con guardia, ognuno corrispondente ad una corsia;
- una variabile che indica di quante corsie è composto il microsegmento;
- un canale pubblico sempre aperto che accoglie le richieste di entrata nel microsegmento;

- due variabili di stato corrispondenti alle due corsie (locked/unlocked);
- un canale utilizzato per uscire dal microsegmento;

La logica che adopera la risorsa protetta è la seguente:

- la richiesta di occupazione di corsia viene accolta dal canale pubblico; il canale rimane sempre aperto, in modo da soddisfare appena possibile la richiesta;
- la richiesta comprende l'identificatore (numero) della corsia che il chiamante desidera occupare. A questo punto possono verificarsi due casi:
  - il microsegmento ha una corsia. Un solo canale interno privato viene utilizzato dal risorsa protetta; la chiamata viene accodata in questo canale, a prescindere dalla corsia richiesta;
  - il microsegmento ha due corsie. Si controlla se la corsia richiesta è libera; in caso positivo, la chiamata viene accodata al canale privato corrispondente. In caso negativo si controlla l'altra corsia: se è libera la chiamata viene accodata nel canale privato corrispondente (si tenta il sorpasso). Se tutte le corsie sono occupate, la chiamata viene accodata nel canale privato con meno richieste (il pilota è costretto a rallentare).
- una volta avuto accesso alla corsia, essa cambia di stato (locked); il pilota attraversa il microsegmento, quindi utilizza il canale di uscita per lasciare il microsegmento (unlock).

La risorsa protetta garantisce la mutua esclusione nel percorrere le corsie, gestisce corsie multiple grazie all'utilizzo dei trasferimenti di coda ed infine garantisce il soddisfacimento delle chiamate nell'ordine corretto, in quanto tutte le code sui canali sono gestite con logica FIFO.

4.2 Stallo e starvation

Per avere una condizione di stallo, le seguenti 4 pre-condizioni devono essere verificate:

- Mutua esclusione;
- Cumulazione di risorse;
- Assenza di prerilascio;
- Attesa circolare.

Nell'implementazione proposta, la mutua esclusione è necessaria in quanto garantisce che, ad ogni istante, una corsia sia occupata al più da una vettura. La cumulazione di risorse non si verifica, in quanto si libera il microsegmento prima di accedere al successivo. Siamo in assenza di prerilascio in quanto il pilota libera volontariamente la corsia. Infine, l'attesa circolare non si verifica, ancora una volta perché si libera il microsegmento prima di accedere al successivo. In conclusione, in questa implementazione lo stallo non si può verificare.

Per quanto riguarda l'accodamento potenzialmente infinito, l'uso di code FIFO nella gestione delle code di chiamata ne scongiura il verificarsi.

5 Distribuzione

In questa parte verrà descritta la realizzazione della parte di distribuzione del sistema.

Verranno evidenziati i nodi in cui lo stesso è diviso, la modalità di comunicazione fra di loro, la gestione dei nomi e la verifica della correttezza del comportamento in caso di caduta di uno dei nodi.

5.1 Nodi individuati

Durante la realizzazione della distribuzione del progetto sono risultati subito evidenti i nodi in cui dividere il sistema. Nello specifico sono stati individuati i seguenti:

- **Startup**: è il nodo che contiene l'entità Startup e coordina la configurazione, l'inizio e la terminazione della gara.
- **Circuit**: è il nodo che contiene l'entità Circuit e riceve le richieste di accesso al circuito da parte dei Drivers.
- **Logger**: è il nodo che contiene l'entità Logger e riceve le notifiche dai Drivers di accesso ai vari segmenti che compongono il circuito.
- **Drivers**: per ogni entità Driver coinvolta nella gara ci sarà un nodo che la contiene.

E' stato scelta questa divisione sia per la distribuzione del carico computazionale, sia per aumentare la difficoltà realizzativa del progetto, sia per evidenziare la reale suddivisione delle entità coinvolte in una gara automobilistica.

5.2 Gestione dei nomi

Le varie entità utilizzano il Naming Service di CORBA, che svolge il ruolo di Name Server, per trovare ed ottenere i riferimenti alle interfacce remote.

Il modo tramite il quale le entità localizzano il Naming Service è molto semplice: il Naming Service si limita ad esporre su di un file testuale il riferimento portabile (IOR) alla sua interfaccia. La posizione ed il nome del file è noto a tutti i nodi, dunque l'acquisizione dello IOR da parte dei client del Naming Service consiste nella lettura della stringa presente nel suddetto file.

Una volta ottenuto il riferimento al Naming Service le altre entità possono ottenere riferimenti alle interfacce di cui sono client o pubblicare le proprie interfacce, se

fungono da server.

Una volta che un'entità server ha finito di svolgere la propria mansione (al termine della gara o dopo la configurazione nel caso dei driver), rimuove dal Name Server il proprio riferimento diventato inutile.

5.3 Comunicazione

Per far comunicare le parti dell'applicazione scritte in Ada con quelle scritte in Java è stato deciso di utilizzare un middleware esterno per poter supportare il modello CORBA, nello specifico PolyORB.

Ada necessitava la presenza di un middleware esterno, mentre Java non ha bisogno di strumenti esterni per fare ciò in quanto è già implementato nella piattaforma Java dalla versione 2.

I servizi condivisi vengono definiti usando l'Interface Description Language (IDL) di CORBA, da cui si ottiene tramite appositi compilatori, il codice sorgente nei linguaggi di programmazione in cui sono scritte le varie entità. Nello specifico “idlac” produce il codice sorgente per Ada, mentre “idlj” produce quello per Java. I vari scheleton e proxy vengono poi creati staticamente a tempo di compilazione.

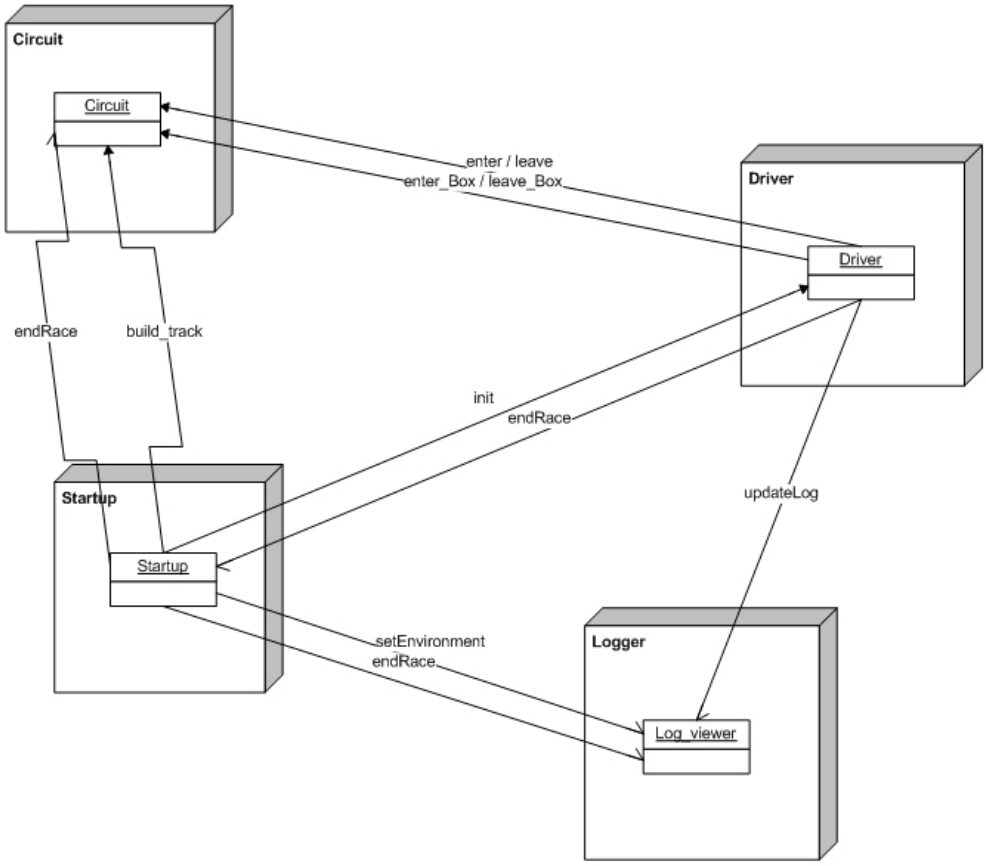
I nodi che fungono da server pubblicano, tramite un task apposito, un riferimento remoto nel Name Server e attendono le richieste da parte dei client.

E' compito degli ORB presenti nei vari nodi il marshalling e l'unmarshalling dei messaggi fra i nodi.

5.4 Oggetti e servizi condivisi

Gli oggetti e i servizi CORBA condivisi sono stati tutti raggruppati in un unico package denominato RI.

Nella seguente figura sono rappresentati graficamente i nodi del sistema, le entità ad esse afferenti e le loro relazioni di dipendenza.



5.5 Verifiche di correttezza

Di seguito si mostrerà come l'applicazione gestisce le eventuali anomalie che possono avvenire in fase di esecuzione.

5.5.1 Comportamento in caso di caduta di un nodo Driver

La comunicazione tra Driver e Logger è asincrona, perciò il logger non è a conoscenza di dove siano localizzati i client. Nel caso di caduta di un Driver il Logger non ne risentirebbe, ad esclusione del fatto che non verrebbero più aggiornate le statistiche e la posizione del Driver in questione.

Da parte di Circuit, la caduta di un Driver non ha conseguenze, a parte nel caso in

cui il Driver avesse occupato una corsia di un segmento. In quel caso la caduta del Driver può essere considerata come un'incidente con la macchina incidentata che occupa quella parte del circuito. Per il resto la gara procede normalmente.

5.5.2 Comportamento in caso di caduta del nodo Startup

La caduta del nodo Startup può avvenire in due momenti diversi, con diverse conseguenze:

- **Prima dell'inizio della gara**: in questo caso non vengono inizializzati i Driver con le loro configurazioni iniziali e non viene creato il tracciato. La caduta in questo caso non produce alcuna eccezione negli altri nodi, che restano in attesa di inizializzazione da parte di una nuova istanza di Startup.
- **Durante la gara o al termine della stessa**: la caduta viene rilevata lato client dai Driver, nel tentativo di invocare il metodo remoto asincrono “endRace”. In questo caso è compito del primo Driver notificare la fine della gara a Logger ed a Circuit. Questi termineranno normalmente, mentre gli altri Driver termineranno notificando una prematura caduta del circuito.

5.5.3 Comportamento in caso di caduta del nodo Circuit

La caduta del nodo Circuit può avvenire in tre momenti diversi, con diverse conseguenze:

- **Prima dell'inizio della gara**: Startup non può invocare il metodo remoto “build_track” e quindi creare il circuito. In questo caso la gara viene annullata e l'applicazione termina notificando l'impossibilità di generare il circuito.
- **Durante la gara**: in questo caso sono i Driver a notare la mancanza di comunicazione con Circuit, nel tentativo di accedere ad un segmento del circuito o di lasciarlo. L'eccezione viene gestita lato client che comunica a Startup la caduta di Circuit. La gara non può continuare senza circuito e quindi termina notificando il motivo della terminazione prematura.
- **A gara finita ma prima di ricevere l'”endRace” da Startup**: in questo caso la gara è terminata e i Driver non necessitano più di accedere al circuito. Il metodo “endRace” è asincrono e quindi Startup gestisce lato client l'eccezione scaturita dalla chiamata a metodo remoto, mentre l'applicazione termina normalmente.

5.5.4 Comportamento in caso di caduta del nodo Logger

Tutte le comunicazioni verso il Logger sono asincrone, perciò il logger non è a conoscenza di dove siano localizzati i client. In caso di caduta del Logger sono state dunque gestite lato client le eventuali eccezioni scaturite da una chiamata a metodo remoto, l'applicazione si limita a notificare tramite un messaggio sulle shell dei client la caduta del Logger e la gara continua senza la visualizzazione delle statistiche.

6 Implementazione software

Un esempio di realizzazione del modello proposto è stato elaborato per mezzo di due linguaggi di programmazione: Ada e Java. In questo capitolo illustreremo il software così creato.

6.1 Entità coinvolte

I componenti del software sono stati ricavati di pari passo alle entità elencate nel modello. Per quanto riguarda Ada, tali entità sono:

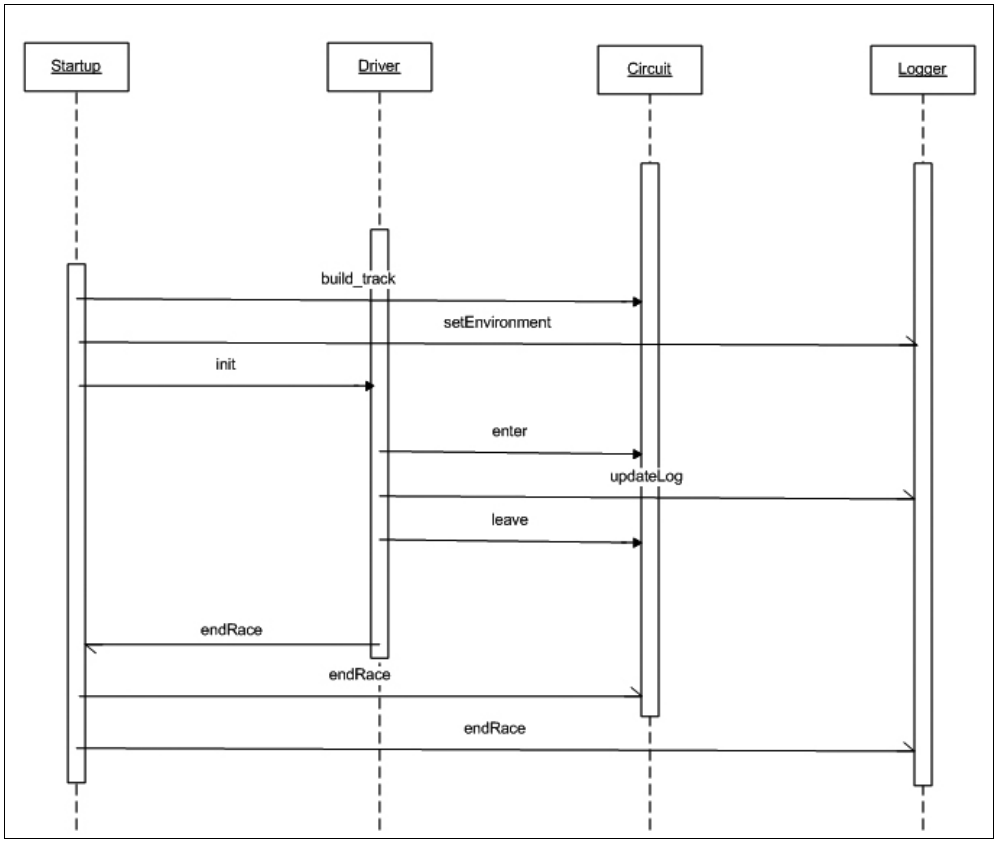
- Gara: l'entità gara è stata implementata nel *package* “Race”. Al suo interno, come da modello, sono implementate tutte le altre entità. In “Race” sono definiti tutti i tipi comuni (segmenti, microsegmenti, …) e soprattutto la procedura “Startup”, che carica tutte le impostazioni, configura gara circuito e piloti, fa partire la gara e attende la sua conclusione.
- Circuito: l'entità circuito è stata implementata nel *package* “Circuit”. Al suo interno è presente il tracciato di gara e i box modellati come sequenze di microsegmenti (risorse protette).
- Pilota: l'entità pilota è stata implementata nel *package* “Driver”. Al suo interno è presente il *task* “Driver”, dalla quale saranno istanziate le entità attive che gareggeranno nel circuito.

Per quanto riguarda Java è presente l'entità:

- Logger: l'entità logger è stata implementata nel *package* “log”. Al suo interno è presente il *task* “Log_viewer”, incaricato di visualizzare le statistiche della gara durante il suo svolgimento, ottenendo informazioni sulla stessa dai Driver.

6.2 Sequenza di esecuzione

Di seguito vengono elencate le operazioni effettuate da ogni entità durante l'esecuzione, illustrate con l'aiuto del seguente diagramma di sequenza.



6.2.1 Race (Startup)

Di seguito sono riportate le attività di lettura e configurazione dei parametri, nonché avvio e termine gara, svolte dall'entità “Race”:

- legge i file di configurazione per piloti, team e circuito;
- costruisce una struttura dati contenente le proprietà di ogni segmento (chiameremo questa struttura LR, Lista Proprietà);
- chiama la funzione di creazione del circuito (LR) presente nell'entità “Circuit” fornendo le caratteristiche del circuito (LP);
- occupa il primo microsegmento del circuito (eventualmente, entrambe le corsie) per creare la linea di partenza;
- inizializza le entità “Driver” secondo l'ordine di comparsa nel file di configurazione relativo. I piloti si accodano in ordine sul primo microsegmento;
- fa partire la gara liberando il primo microsegmento;
- crea la propria interfaccia remota e ne pubblica il riferimento sul Naming Service;

- resta attiva in attesa del segnale di terminazione della gara da tutti i piloti;
- comunica la fine della gara alle entità “Logger” e “Circuit”;
- termina l'esecuzione.

6.2.2 Circuit

Di seguito sono riportate le attività necessarie per inizializzare il circuito e renderlo disponibile per la gara effettuate dall'entità “Circuit”:

- crea la propria interfaccia remota e ne pubblica il riferimento nel Naming Service;
- aspetta di essere configurato da “Race”;
- ottenute le caratteristiche del circuito da “Race”, lo costruisce popolando un'apposita struttura dati contenente l'implementazione dei microsegmenti (tracciato di gara e box). Chiameremo questa struttura LR, Lista Risorse;
- rende accessibile il circuito ai “Driver”;
- attende il segnale di fine gara da “Race”;
- termina l'esecuzione.

6.2.3 Driver

Di seguito sono elencate le attività che compie il pilota dalla sua inizializzazione fino alla fine della gara:

- crea la propria interfaccia remota e ne pubblica il riferimento sul Naming Service;
- recupera il riferimento remoto del logger;
- recupera il riferimento remoto del circuito;
- attende di essere configurato da “Race”;
- si configura con i parametri passati da “Race” (nome del pilota, scuderia, coefficienti di accelerazione/decelerazione, velocità massima, strategia, tracciato di gara “LP”, numero di giri da effettuare);
- si posiziona in coda al primo microsegmento e attende l'avvio della gara;
- fino alla fine della gara, per ogni microsegmento del circuito, ripete le seguenti operazioni:
  - chiede l'accesso al microsegmento a “Circuit”;
  - ottenuto l'accesso, a seconda della velocità attuale, degli avversari incontrati sul percorso e della porzione di segmento in cui si trova accelera, decelera o mantiene la velocità;
  - calcola il tempo necessario per attraversare il microsegmento, tenendo conto dell'accelerazione/decelerazione appena effettuata;
  - comunica al logger occupazione segmento e tutti i dati necessari (velocità, id pilota, numero segmento)

- si sospende per attraversare il microsegmento;
- libera il microsegmento;
- passa al microsegmento successivo.

- al termine di ogni giro, controlla se la strategia prevede la fermata ai box ed in caso positivo effettua il drive-through nella corsia dei box;
- riferisce a “Race” quando ha ultimato il numero di giri previsti;
- in caso di incidente riferisce al logger la natura dell'incidente e il fatto di aver terminato in anticipo la gara;
- termina l'esecuzione.

6.2.4 Logger

- legge dal file “ior.txt” nella cartella “txt” lo IOR del Naming Service di CORBA e ne recupera il riferimento remoto
- crea la propria interfaccia remota e ne pubblica il il riferimento nel Naming Service
- attende di essere configurato da Startup
- crea una struttura dati con le statistiche di ogni pilota (intermedi, tempi sul giro, velocità...)
- mette a disposizione una procedura (“updateLog”) chiamata dai Driver ad ogni ingresso di segmento
- ad ogni chiamata di “updateLog” aggiorna lo stato di quel pilota e la classifica che viene visualizzata costantemente coerente con lo stato della gara
- al termine della gara notifica la fine e rimane attivo per permettere la visualizzazione delle statistiche.

6.3 La variabile tempo

Il tempo gioca un ruolo importante in presenza di threads multipli e di sincronizzazione con risorse protette. Per queste ragioni riportiamo in questo paragrafo le scelte effettuate nell'utilizzo del tempo per preservare il corretto svolgimento della gara.

In questo paragrafo, con il termine “tempo di gara” si intende il tempo trascorso dal pilota nell'affrontare la gara ed è composto dalla somma dei tempi di percorrenza di ogni singolo segmento; con “tempo di sistema” si intende il tempo istantaneo ricavato dal sistema sottostante il programma.

Ogni driver, alla partenza, salva il valore istantaneo del tempo di sistema. Con “partenza” non si intende l'avvio della gara dato dal modulo Race (Startup), ma il momento in cui il driver accede per la prima volta al primo segmento del circuito.

Il tempo di gara è formato sommando tutti i tempi di percorrenza dei segmenti attraversati. Quando un driver ha calcolato il tempo di percorrenza del segmento attuale, lo somma al tempo di gara. La somma tra tempo di sistema salvato all'inizio e il tempo di gara da il tempo di sistema al quale il driver deve passare al segmento successivo. Il thread del driver si sospende, se necessario, fino a tale tempo per simulare il passaggio del segmento.

In questo modo il tempo di gara è preservato da fattori inquinanti quali:

- prerilascio;
- tempo di calcolo: il tempo necessario al programma stesso per eseguire le operazioni di calcolo;
- tempo di attesa in coda su chiamata alla risorsa protetta (blocco su guardia chiusa).

7 Compilazione ed esecuzione

7.1 Requisiti

Per la compilazione e l'esecuzione del progetto sono necessari i seguenti requisiti:

- Java SDK 1.6 o successivo
- Polyorb 2.4.0 o successivo
- Compilatore gnat 2008 o successivo

7.2 Compilazione

Viene fornito uno script automatizzato per la compilazione dell'intero progetto che produrrà gli eseguibili relativi alle varie parti in cui è suddiviso il progetto.

Per eseguire lo script è necessario digitare da terminale, nella cartella iniziale del progetto:

- `sh make.sh`

7.3 Files di configurazione

I files di configurazione contenuti nella cartella “txt” sono i seguenti:

- circuit.txt, che contene i parametri del circuito di gara. Per ogni macro segmento del circuito sono indicati:
  - lunghezza (in decine di metri)
  - velocità massima all'inizio del macro segmento (in km/h)
  - velocità massima alla fine del macro segmento (in km/h)
  - dopo quanti metri passare dalla prima velocità alla seconda
  - numero di corsie del macro segmento
- race.txt, che contiene i parametri della gara, nello specifico i giri da effettuare.
- drivers.txt, che contiene i dati di ogni pilota. Per ognuno di essi vengono indicati:
  - nome
  - scuderia

- numero vettura (ID del pilota)
- coefficiente di accelerazione [1..10]
- coefficiente di decelerazione [1..10]
- velocità massima della vettura (in km/h)
- strategia di gara, rappresentata dai giri in cui il pilota si fermerà ai box

7.4 Esecuzione

7.4.1 Avvio del progetto

Per l'avvio del sistema è fornito uno script che avvia automaticamente le entità del sistema. Tale script va lanciato con il seguente:

- `sh run_all.sh`

Sono forniti degli script per eseguire le singole entità, da lanciare con i seguenti comandi.

Tali script vanno lanciati nel seguente ordine:

- `sh run_naming.sh`

  avvia il servizio di naming di Polyorb.

- `sh run_logger.sh`

  avvia il visualizzatore di statistiche con il tabellone della gara.

- `sh run_circuit.sh`

  avvia il circuito

- `sh run_driver.sh <ID driver>`

  avvia un singolo driver. <ID driver> deve essere un numero intero a partire da 1. Ogni nuovo driver deve avere come ID il numero successivo all'ultimo driver creato.

- `sh run_startup.sh`

  avvia la gara.

8 Conclusioni

8.1 Linguaggi di programmazione utilizzati

Per la realizzazione del progetto, in conformità con la decisione di svolgere il livello 3 della specifica, sono stati utilizzati due linguaggi di programmazione:

- **Ada**: un linguaggio che possiede molti costrutti utili per implementare correttamente la concorrenza e garantire un corretto acceso a risorse condivise;
- **Java**: è un linguaggio dotato di grande facilità d'uso ed integrazione, soprattutto per quanto riguarda la realizzazione di interfacce grafiche e per questo motivo è stato scelto per la realizzazione del logger.

8.2 Scelte alternative a quelle effettuate

Una possibile scelta alternativa a quelle effettuate è quella di utilizzare Distributed Ada al posto di CORBA.

Per quanto riguarda la comunicazione fra parti del sistema scritte in linguaggi di programmazione differenti, la scelta di utilizzare un middleware come CORBA è stata una scelta obbligata; d'altro canto per la comunicazione fra nodi scritti in Ada sarebbe stato possibile utilizzare il **Distributed Ada** descritto nell'Annex E dell'Ada Reference Manual.

Mediante tale sistema si sarebbe ottenuta una maggior trasparenza e pulizia del codice, a scapito però di un inferiore controllo sulle modalità di comunicazione.

E' stato scelto di utilizzare solo CORBA per la comunicazione fra nodi per avere un unico middleware e un unico servizio di naming presente nel sistema.

8.3 Possibili sviluppi futuri

Verranno qui descritte alcuni possibili sviluppi ed ampliamenti del progetto.

8.3.1 Ampliamento del modello

Per rendere la simulazione più realistica, si potrebbe procedere all'inclusione nel modello di alcune caratteristiche delle competizioni di Formula 1 escluse dalla nostra realizzazione. Esempi di tali caratteristiche si possono trovare nel paragrafo 3.6 (Restrizioni del modello conosciute).

8.3.2 Visualizzazione grafica del tracciato

Una visualizzazione grafica del tracciato potrebbe rendere più semplice seguire lo svolgersi della gara, specialmente per quanto riguarda le posizioni dei vari piloti.

Una simile visualizzazione permetterebbe inoltre di constatare con più evidenza il corretto accesso alle risorse condivise rappresentanti i vari segmenti del tracciato.

8.3.3 Caricamento dei dati da xml

Un'altra funzionalità non obbligatoria che sarebbe possibile implementare in futuro riguarda i file contenenti le configurazioni iniziali della simulazione.

Invece di utilizzare dei semplici file di testo, le configurazioni potrebbero essere salvate in file xml letti poi utilizzando dei parser appositi.

9 Bibliografia

1. http://en.wikipedia.org/wiki/Formula_One_car (Accelerazione e decelerazione tipiche di una monoposto di F1)
2. Slides del corso “Sistemi concorrenti e distribuiti” tenuto dal Prof. Vardanega Tullio nell'a.a. 2008/2009 (http://www.math.unipd.it/~tullio/SCD/2008/calendario.html)
3. Ada Reference Manual (ISO/IEC 8652:1995/Amd 1:2007) consultabile al sito http://www.adaic.com/standards/ada05.html