Capitolo 3

Introduzione ad HLA

3.1 Overview

Supponiamo di avere un insieme di simulazioni e che sia nostra intenzione riunirle in un unico sistema: ad esempio le simulazioni di agenti mobili dislocati su varie regioni. L'High Level Architecture (HLA) riveste il ruolo di *collante* al fine di ottenere un'unica simulazione composta, la sua definizione è molto prossima a quella di un *middleware* simulativo.

Introducendo la terminologia essenziale possiamo definire questa macro-simulazione come "federazione" e i singoli componenti come "federati" (fig. 3.1). Da subito facciamo una precisazione fondamentale: questi federati possono essere a loro volta scissi in altri sottolivelli architetturali. Su questo semplice concetto si basa tutta la nostra politica di adattività e dinamicità della simulazione rispetto alle risorse disponibili.

Nella pratica una federazione è composta da questi elementi essenziali:

1. un software che funga da supporto: il *Runtime Infrastructure* (RTI);
2. un modello per lo scambio di dati tra federati all'interno di una federazione: il *Federation Object Model* (FOM);
3. un certo numero di *federati*.

Approfondendo: il FOM rappresenta il dettaglio delle interazioni, per forma e tipologia, che vengono scambiate tra federati durante l'esecuzione di una simulazione.

All'interno di una federazione non vi è alcuna necessità di omogeneità tra federati, nel senso che ogni federato può svolgere un ruolo specifico, avendo diverse funzionalità. Ad esempio sono comuni i casi di federati che ricevono passivamente i dati della simulazione senza rivestire un ruolo attivo in essa e che si limitano a svolgere il ruolo di collettori o di visualizzatori di informazioni e statistiche (fig. 3.2).

Alcuni federati possono integrare nella simulazione input esterni, ad esempio l'interazione umana oppure offrire interfacce verso gli utenti: questa tecnologia è stata studiata anche con lo scopo di fare da supporto a simulatori usati per l'addestramento militare, si capisce bene come sia quindi necessario avere la possibilità di integrare azioni da parte del "giocatore" ed il ritorno di reazioni.

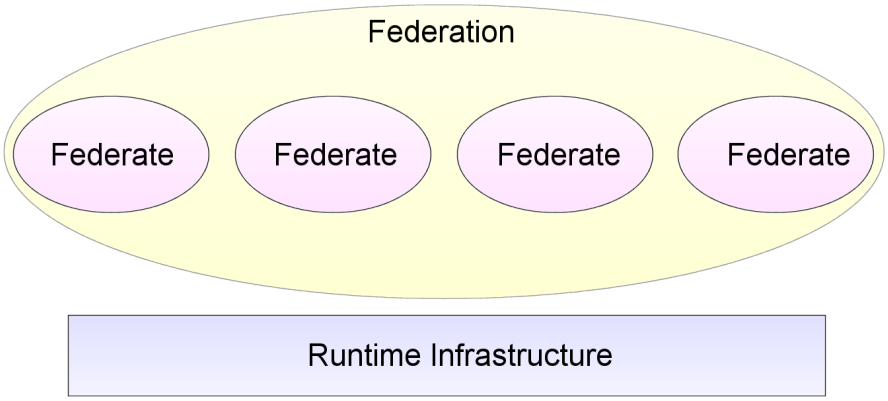


Figura 3.1: Struttura di una federazione

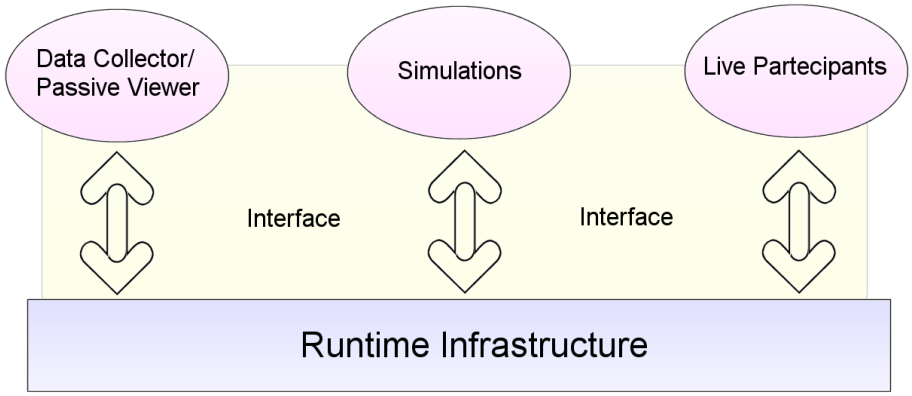


Figura 3.2: Federazione formata da federati eterogenei

3.2 Brevi cenni storici

Nel 1995 [DMSO95] appare evidente come il DoD abbia l'intezione di modificare la situazione esistente della simulazione con l'obiettivo di favorire il riutilizzo delle componenti e garantendo una maggiore interoperabilità. Questa generica intenzione si traduce solamente in seguito in un piano strategico che favorisca la definzione degli investimenti necessaria per la creazione di un framework generale. Sin dal principio venne fatta la scelta di non sviluppare una tecnologia di esclusivo interesse per il mondo militare e non diffusa all'esterno, essenzialmente per problemi di costi l'orientamento fu quello di coinvolgere contemporaneamente stato, industria e laboratori di ricerca. Il ruolo statale è quello di finanziatore iniziale, di guida e di controllore ma con l'obiettivo ben chiaro di defilarsi almeno parzialmente una volta creato uno sviluppo sufficiente.

Materialmente in tempi brevi venne commissionata un'indagine conoscitiva sulla possibilità di raggiungere gli obiettivi prefissati; al termine di quest'indagine risultò chiaro come la definizione dello standard dovesse in qualche modo essere direttamente controllata da coloro che in seguito avrebbero dovuto usare la tecnologia. A questo scopo venne formato un gruppo denominato *Architecture Management Group* (AMG) composto dai maggiori utilizzatori di tecnologia simulativa all'interno del settore "difesa". Ovviamente si trattava di una comunità piuttosto eterogenea per caratteristiche di utilizzo, questo ha complicato notevolmente la progettazione ma ha anche dato la possibilità di analizzare e predisporre soluzioni per un ampio spettro di problematiche.

Il meccanismo con cui materialmente si realizzarono le esperienze fu l'uso di componenti software definite protofederazioni; queste non implementavano altro che le maggiori problematiche nelle distinte aree di simulazione (Platform Prototype Federation, Joint Training Protofederation, Analysis Protofedration, Engineering Prototype Federation), analizzando gli aspetti particolari ed i problemi di interoperabilità. Ovviamente al fine di ottenere un risultato coerente ogni federazione condivideva architettura, specifiche di interfaccia, modello di interazione tra gli oggetti e implementazione del runtime (RTI). Nel corso del 1996 [DOD96] proprio da queste esperienze nasce la versione iniziale (1.0) di HLA.

La chiave di volta di tutta l'implementazione e aspetto più critico per il suo successo risiede nel runtime (RTI); la sua esistenza era fondamentale perché lo sviluppo di tutta la struttura potesse procedere. Proprio per questo motivo si indicò immediatamente che la precedenza era orientata verso le funzionalità, l'aspetto prestazionale andava tenuto in considerazione ma risultava di secondo piano rispetto al problema di rendere utilizzabile e "di successo" questa tecnologia: le ottimizzazioni non erano prioritarie[1].

Un prototipo iniziale di RTI venne realizzato in modo congiunto da un team formato dalla MITRE Corporation, MIT Lincoln Laboratory, SAIC, Virtual Technology Corporation. Passando attraverso una serie di prototipi (la serie RTI 0.X, RTI 1.0 e RTI-S) si è giunti alla RTI 1.3: la versione che con qualche revisione è ancora disponibile e che risulta essere la prima a fornire tutti i requisiti previsti dalla definizione architetturale.

Con l'adozione della tecnologia HLA ad IEEE standard (1516) sono state

[1] La situazione anche attualmente non è migliorata di molto, il runtime continua ad essere piuttosto esigente in termini di risorse: implementazioni dal basso impatto di memoria potrebbero essere di interesse in molti ambiti.

apportate alcune modifiche che rendono l'implementazione dell'IEEE standard ed i vecchi prototipi parzialmente inconsistenti ed incapaci di interagire: per questo motivo ci si attende una doppia implementazione del RTI a seconda che il produttore decida di seguire lo standard IEEE oppure rimanga ancorato alla vecchia definizione DMSO.

Assieme al RTI si è rilevato come vi fossero delle altre necessità cruciali per il successo della tecnologia. Lo sviluppo di federazioni sempre più complesse e l'applicazione della tecnologia a casi pratici hanno mostrato che l'assenza di una serie di strumenti di supporto avrebbe provocato grossi ritardi di utilizzo. Per questo motivo oltre al RTI il DMSO ha provveduto a finanziare lo sviluppo di tutta una serie di software, orientati alla definizione automatica delle interfacce di comunicazione tra federati, al monitoraggio dell'esecuzione delle simulazione e all'analisi dei risultati. Questi strumenti, dotati di un'interfaccia visuale hanno lo scopo di facilitare le configurazioni e le personalizzazioni e sono stati rilasciati gratuitamente sempre nell'ottica di favorire un'adozione quanto più vasta possibile della tecnologia, creare una grossa base di utenti quindi mercato. In realtà proprio per questo settore di prodotti l'iniziativa privata è stata più intraprendente che in altri e ora sono disponibili ulteriori prodotti sviluppati da terze parti.

Una volta superata la fase di progettazione, implementazione e verifica della tecnologia HLA [DMSO98a, DMSO98b, DMSO98c], il DMSO a partire dall'ottobre 2000 ha previsto che tutte le simulazioni di sua competenza realizzate dovessero essere conformi alla tecnologia, creando quindi uno standard di fatto e obbligando la migrazione della base esistente.

3.3 Descrizione dello standard

HLA è rappresentato da un'architettura software e non da una particolare implementazione, di conseguenza questo standard è definito da una serie di documenti e non da implementazioni di riferimento[2]. Ultimamente HLA ha raggiunto il livello di standard dell'Institute of Electrical and Electronics Engineers (IEEE), rispettivamente come P1516 (HLA Rules), P1516.1 (Interface Specification) e P1516.2 (OMT).

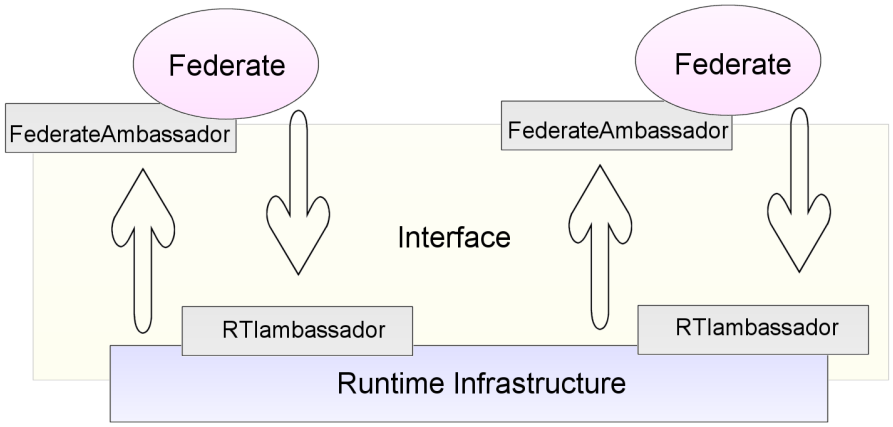
In dettaglio lo standard risulta formato da tre componenti:

1. *HLA Rules*;
2. *Object Model Template* (OMT);
3. *Interface Specification.*

Le prime regolano i principi e le metodiche attraverso le quali i singoli federati che fanno parte di una federazione possono interagire durante l'esecuzione.

[2]Questo non significa che il DMSO non abbia finanziato un'implementazione di riferimento. Lo scopo di questa implementazione realizzata sotto controllo "diretto" era quello di fornire nel minor tempo possibile una versione funzionante dell'infrastruttura a tutti gli interessati alla tecnologia. Si è trattato di un modo per sobbarcarsi i costi iniziali dell'implementazione incoraggiando comunque il settore privato ad affiancare il pubblico o eventualmente prenderne il posto una volta superati i primi momenti.

Figura 3.3: Meccanismo di interazione tra Runtime Infrastructure e Federati



Nel caso dell'OMT, si torna a fare riferimento al Federation Object Model (FOM), l'OMT non è altro che la definizione di struttura ammessa per la realizzazione del FOM, un meta-modello che i FOM devono seguire per risultare ben formati.

Le specifiche di interfaccia invece definiscono come avvengono le interazioni tra i vari federati e RTI: queste interazioni sono standardizzate, soprattutto facendo riferimento alla possibilità di avere implementazioni del RTI diverse tra di loro e non completamente omogenee (fig. 3.3).

3.4 Descrizione di HLA

Come indicato precedentemente HLA rappresenta un'architettura per simulazione basata sui "componenti", in questa visione i componenti sono rappresentati dai federati, ovvero **singole simulazioni che hanno la possibilità di interagire tra loro**. I componenti sono simulazioni, quindi insiemi di entità e non necessariamente singole entità.

Vediamo alcuni degli obiettivi che sono alla base di questa architettura software:

- la possibilità di decomporre simulazioni di grosse dimensioni in problemi molto più piccoli e facilmente gestibili. E viceversa: la possibilità di ricomporre elementi di simulazione in un'unica struttura capace di interagire ed evolversi in modo coerente;

- la possibilità di combinare queste simulazioni anche con nuove entità, magari ancora inesistenti al momento della loro creazione. Così da generare una struttura che non richieda modifiche delle parti esistenti al momento di aggiungere nuove funzionalità, in pieno accordo con i principi di riusabilità;

- un livello quando più profondo possibile di isolamento tra le simulazioni e la tecnologia utilizzata nell'infrastruttura simulativa: così da proteggere ulteriormente le simulazioni dai progressi tecnologici, preservando il funzionamento del sistema e rendendolo in grado di evolversi continuamente, riducendo i vincoli tecnologici di riferimento.

A questo punto il fatto che l'architettura supporti la costruzione di simulazioni in ambiente distribuito, tra computer appartenenti ad una singola Local Area Network (LAN) (fig. 3.5), su Internet o sullo stesso elaboratore (fig. 3.4), diventa un effetto del supporto dei componenti[3]. Nella definizione architetturale non c'è nulla che presume o richiede che l'implementazione del sistema avvenga in ambiente distribuito, l'approccio è assolutamente trasparente.

In dettaglio un federato rappresenta una simulazione o un tool (ad esempio visualizzazione, logging dai dati ecc.) riutilizzabile anche all'interno di federazioni diverse da quella originaria dove era stato pensato, implementato e collocato. Tutto questo grazie alle appropriate interfacce di funzionamento: il federato rappresenta nella visione generale di HLA l'**unità minima di riutilizzo del codice**.

Un altro risultato fondamentale di questa visione consiste nella possibilità di far interagire simulazioni diverse senza la necessità di combinarle in un unico software residente in una sola locazione. Attraverso questa tecnologia i vari componenti software possono continuare a risiedere sul supporto elaborativo del proprietario e comunque interagire con altri componenti, risulta solamente necessaria l'installazione del software di supporto alla simulazione e una specifica dettagliata delle interfacce di interazione. Ecco quindi molto evidente il vantaggio in termini di garanzia della proprietà intellettuale ed il risparmio in termini di gestione e competenza.

3.5 Tipologie di comunicazione tra federati

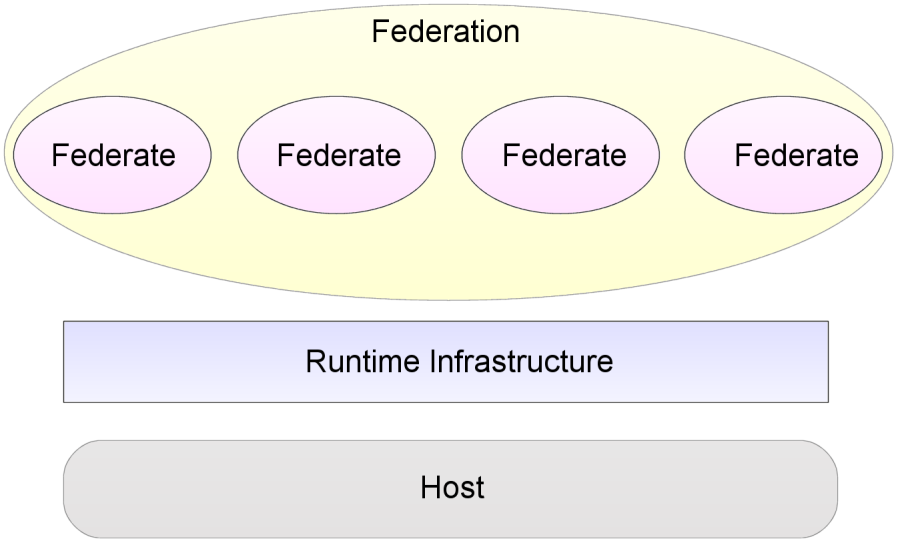
Vediamo ora come sia possibile la comunicazione tra federati. L'*Object Model Template* (OMT) definisce la struttura del *Federation Object Model* (FOM) creando un "vocabolario" locale. In pratica il FOM assegna dei nomi ai singoli oggetti e alle occorrenze, non si occupa di tutte le entità presenti ma solamente di quelle che assumono un significato nell'interazione con gli altri federati: non avrebbe senso e costituirebbe un lavoro inutile marcare anche le entità interne all'esecuzione del federato, naturalmente si persegue l'obiettivo di ridurre al minimo le comunicazioni tra federati. In questo modo il FOM assume il significato di vocabolario per le interazioni che avvengono attraverso l'RTI[4].

Vista la sua importanza è chiaro come il FOM funga da parametro iniziale all'esecuzione del federato nel RTI: nella pratica assume la forma di un file di configurazione. Inoltre questa struttura assume anche il ruolo di ulteriore livello di astrazione: cambiamenti nella struttura delle entità del federato non

[3] E' un grosso problema nella progettazione e nell'implementazione del RTI che deve fungere da supporto a tutta l'esecuzione, indipendentemente dallo scenario di utilizzo.

[4] Implementativamente nulla vieta che due federati comunichino "scavalcando" l'RTI, ovviamente se questa pratica viene attuata per comunicare dati definiti nel FOM risulta in violazione alle regole e comporta una perdita di consistenza di tutta la simulazione: l'RTI non è in grado di dare alcuna assicurazione di correttezza su queste comunicazioni di cui non ha traccia.

Figura 3.4: Federazione implementata su un singolo Host



provocano alcuna modifica al RTI che rimane perfettamente indipendente, stessa cosa avviene con alcune limitazioni (come vedremo meglio in seguito) per gli altri federati che interagiscono.

Veniamo ai metodi veri e propri di comunicazione che sono disponibili: **oggetti** ed **interazioni**. I primi hanno utilità principalmente per rappresentare delle entità durature nel tempo e che subiscano una serie di cambiamenti di stato, le interazioni sono invece estremamente utili per rappresentare singoli cambiamenti dell'ambiente. Per inciso, tutta l'architettura del framework implementato per la realizzazione di questa tesi è basata su interazioni.

3.5.1 Oggetti

Un oggetto è un'entità simulata di interesse a più di un federato e che di conseguenza viene presa in considerazione dal RTI, come detto è una rappresentazione che risulta particolarmente adatta per entità durature nel tempo, soggette ad aggiornamenti. L'OMT non definisce singoli oggetti ma *classi* di oggetti: ogni classe a sua volta è costituita da un insieme (eventualmente anche vuoto) di attributi (ulteriori entità a cui viene attribuito un ben preciso *label*).

La struttura delle classi di oggetti non è lineare ma risulta raggruppata in una struttura ad albero, ogni classe è direttamente figlia di una super-classe, l'eredità multipla risulta vietata. La radice di tutta questa struttura viene definita *ObjectRoot*, per la denominazione della classe si usa il tipico procedimento di concatenare tutte le classi incontrate nell'attraversamento (a partire dal'ObjectRoot sino alla classe stessa) separando le varie classi con un punto, in questo

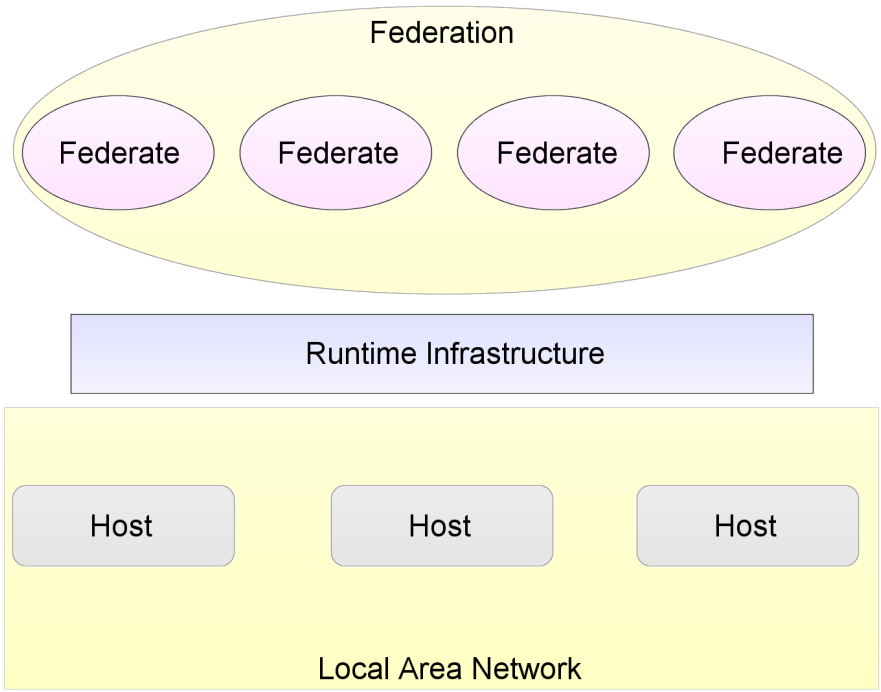


Figura 3.5: Federazione implementata su una Local Area Network

modo otteniamo il cosiddetto *fully qualified name*. Vista l'impossibilità di avere eredità multiple e la necessità di definire in modo univoco le classi, ogni fully qualified name risulta univoco all'interno dello stesso FOM.

Come facilmente intuibile ogni classe figlia eredita gli attributi appartenenti alla progenitrice, gli attributi ereditati vengono normalmente definiti *available attributes*. La classe ObjectRoot possiede un attributo obbligatorio definito *privilegeToDeleteObject*, di conseguenza ogni classe appartenente al FOM possiede almeno un attributo definito.

Gli attributi, come li abbiamo visti finora, assumono un doppio significato, possono indicare entità appartenenti in generale alla classe oppure possono appartenere ad una singola istanza dell'oggetto che rappresenta la classe. Nel primo caso assumono la definizione di *class attributes*, nel secondo invece *instance attributes*. L'utilità di questo meccanismo è evidente: un federato che sia interessato ad essere aggiornato su tutte le variazioni di stato di una qualsiasi istanza di un certo oggetto chiederà l'aggiornamento di un class attribute, nel caso invece sia interessato solamente alle variazioni di un particolare oggetto la sottoscrizione avverrà solamente per un determinato instance attribute.

3.5.2 Interazioni

E' estremamente facile immaginare le interazioni come messaggi che vengono scambiati tra federati. A differenza degli oggetti il loro utilizzo è quindi opportuno per modellare i cambiamenti di stato e le comunicazioni che non sono destinate a ripetersi in modo costante ed immodificato nel tempo.

Il messaggio vero e proprio che viene trasportato da un'interazione è rappresentato da un insieme di dati: l'RTI non si cura minimamente del significato di questi, essi assumono un senso semantico in base all'interpretazione data dal mittente e dal destinatario che quindi devono concordare su un comune protocollo di interpretazione; l'unico lavoro svolto dal sottosistema simulativo è l'assicurazione che non vengano violate le regole di HLA.

Ogni interazione, ovviamente, è caratterizzata da un mittente e da un destinatario, il destinatario non deve necessariamente essere un singolo federato, attraverso vari meccanismi si possono creare delle strutture di interazione broadcast o eventualmente multicast.

I dati di volta in volta trasportati vengono definiti "parametri" e risultano etichettati con un nome: anche in questo caso la struttura di gestione delle interazioni è prettamente gerarchica, la radice dell'albero è costituita dalla *InteractionRoot*. L'eredità dei parametri in questa struttura ad albero ricalca esattamente quanto descritto precedentemente per gli oggetti.

3.5.3 Vantaggi della struttura gerarchica per oggetti e interazioni

Come abbiamo visto sia gli oggetti che le interazioni sono "raggruppati" in una struttura ad albero, la scelta non è solamente di comodità di utilizzo e di referenziazione ma nasconde l'intento di proteggere i federati dai cambiamenti strutturali che possono avvenire negli altri federati con cui interagisce.

La specializzazione di un oggetto in sotto-oggetti maggiormente specializzati, attraverso questo "stratagemma" non produce nessun problema nei federati che erano stati implementati seguendo la prima definizione di federato. Si è esteso il

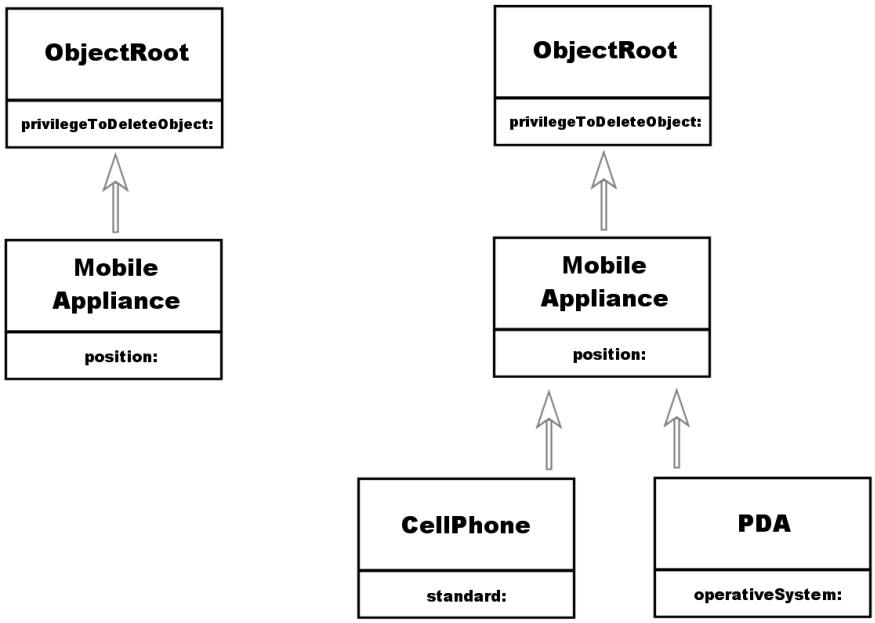


Figura 3.6: Modello di rappresentazione dispositivi mobili

sistema, si sono offerte nuove possibilità di interazione mantenendo compatibilità con l'esistente.

Vediamo (fig. 3.6) un esempio pratico: immaginiamo di voler rappresentare un insieme di dispositivi mobili dislocati su un certo territorio, in figura vediamo una possibile rappresentazione, ovviamente ognuno di questi dispositivi possiede un attributo posizione. La specializzazione di questo modello porta alla modifica della struttura, ora abbiamo due ulteriori oggetti che rappresentano i telefoni cellulari e palmari wireless, il modello precedente è stato esteso. I federati progettati per funzionare con la versione iniziale degli oggetti non hanno nessun problema nel proseguire a funzionare correttamente, dal loro punto di vista non c'è stato nessun cambiamento ma allo stesso tempo possiamo creare nuove simulazioni che sfruttino la specializzazione del modello.

3.6 Regole

Una parte dello standard HLA è formato da regole, queste sono diverse a seconda che si riferiscano al comportamento che deve essere tenuto dalla federazione nel suo complesso o dai singoli federati. Vediamole in rassegna in quanto, anche se particolarmente schematiche, offrono un veloce riassunto ed una buona immagine di come funzioni il sistema.

3.6.1 Federazione

1. *Tutte le federazioni devono avere un FOM, questo deve essere realizzato seguendo le direttive previste dall'OMT.*
   Il FOM rappresenta il vocabolario utilizzato da una federazione e di conseguenza descrive gli oggetti e le interazioni che esso espone verso il possibile utilizzo da parte di altri federati: in altri termini si tratta della definizione della sua interfaccia di utilizzo. Risulta quindi ovvia la sua fondamentale importanza.

2. *All'interno di una federazione, tutte le istanze di oggetti che abbiano a che fare con la simulazione devono essere contenute all'interno delle federazioni e non possono quindi risiedere nel RTI.*
   Il compito del RTI è quello di fungere da supporto: se fosse stato progettato con lo scopo di avere un ruolo attivo nelle interazioni sarebbe necessariamente stato meno generico. Questa regola rafforza il concetto per il quale l'RTI non ha nessun ruolo di database degli oggetti e delle interazioni di supporto alla simulazione, si limita a "non avere stato" ma a fornire il supporto per il cambio di stato dei federati.

3. *Durante l'esecuzione di una federazione, ogni scambio di dati (definiti nel FOM) tra federati deve necessariamente avvenire attraverso l'RTI.*
   Non è possibile in alcun modo[5] "scavalcare" la presenza del RTI per tutte quelle comunicazioni che riguardino dati precedentemente definiti all'interno dell'interfaccia del federato. Qualora questo avvenga non potrebbe più esservi alcuna garanzia sull'esecuzione (ad esempio l'ordinamento temporale degli eventi) e la riusabilità di tutta la struttura sarebbe minata.

4. *Durante l'esecuzione di una federazione, i federati possono interagire con l'RTI seguendo quanto definito nell'HLA Interface Specification.*
   Non solo i singoli federati espongono delle proprie interfacce, anche l'RTI espone una propria interfaccia d'interazione nei confronti dei federati e ovviamente i federati sono tenuti a seguire queste regole. L'obiettivo della norma è quello di impedire che le peculiarità di ogni singola implementazione del RTI possano creare problemi all'uso delle federazioni su implementazioni diverse da quelle originarie di progettazione.

5. *Durante l'esecuzione di una federazione, ogni istanza di attributo può contemporaneamente essere posseduta da al più un federato per volta.*
   Non è accettabile che un'istanza di oggetto sia posseduta contemporaneamente da più di un'entità alla volta; questa norma nasce dal vincolo che il proprietario dell'istanza assume la responsabilità di mantenerne aggiornato il valore e propagare le eventuali modifiche. Qualora vi fossero proprietà comuni questa porterebbe necessariamente a conflitti d'aggiornamento. Può invece accadere che un federato rassegni di propria iniziativa la proprietà di un attributo che a questo punto diviene acquisibile da un altro federato.

[5] Non si fa riferimento ad una possibilità implementativa ma al rispetto della norma.

3.6.2 Federati

1. *Ogni federato deve avere un HLA Simulation Object Model (SOM), realizzato secondo quanto definito dall'OMT.*
   Le interfacce esportate da ogni federato sono definite in base al proprio SOM, l'OMT definisce le modalità di definizione del SOM e un insieme minimo di informazioni che sono necessarie perché il federato possa essere considerato parte di una federazione.

2. *I federati devono essere in grado di aggiornare e riflettere gli aggiornamenti degli oggetti, spedire o ricevere interazioni; in base a quanto specificato all'interno del loro SOM.*
   A seconda di quanto definito nel SOM l'RTI deve fornire ai federati la possibilità di svolgere le interazioni definite, seguendo quanto prescritto dal SOM, queste prevedono la possibilità di aggiornare il valore degli attributi in proprio possesso, ricevere gli aggiornamenti degli attributi considerati d'intereresse, comunicare attraverso la spedizione e la ricezione di interazioni.

3. *I federati devono essere in grado di trasferire e/o ricevere la proprietà di attributi in maniera dinamica rispetto all'esecuzione della simulazione, in ottemperanza con quanto definito nel SOM.*
   La struttura definita da HLA in questo caso non è statica ma permette che i federati modifichino in maniera del tutto dinamica la proprietà di attributi, cedendola o acquisendola di volta in volta. Rimane il vincolo che queste possibilità siano precedentemente indicate nel singolo SOM di ogni federato.

4. *I federati devono avere la possibilità di modificare le condizioni per le quali sono tenuti a fornire aggiornamenti dei propri attributi, secondo specifica SOM.*
   Le operazioni di aggiornamento possono avvenire in base ad una serie di eventi (passaggio di tempo, superamento di una soglia ecc.), anche in questo caso l'aggiornamento degli attributi può avvenire secondo caratteristiche dinamiche purchè questo sia stato definito nel SOM.

5. *I federati devono essere in grado di gestire il proprio tempo locale di simulazione in modo che sia possibile il coordinamento dello scambio di dati tra membri di una federazione.*
   E' fatto obbligo al RTI di fornire ai federati tutto un insieme di funzioni attraverso le quali ogni federato possa gestire il trascorrere del tempo ed indicare i suoi rapporti temporali con il resto della federazione. Ovviamente è auspicabile che siano disponibili diverse metodologie per la gestione del tempo e per il coordinamento, e che ogni federato sia in grado di definire esattamente a quale tipologia intende aderire.

3.7 Management Object Model (MOM)

Come abbiamo visto le federazioni sono solitamente dei sistemi distribuiti, non è un vincolo ma senza la necessità di distribuzione non avrebbe molto senso utilizzare la tecnologia HLA. La gestione di un sistema distribuito come sempre

pone delle problematiche, il Management Object Model (MOM) è la risposta HLA all'esigenza dei singoli federati di avere una visione comune della federazione e delle entità che ne sono coinvolte. Vista la sua importanza per la vita delle federazioni questa struttura è stata standardizzata, la sua implementazione e gestione è compito del RTI che provvede alla creazione di una serie di classi liberamente accedibili ed eventualmente estendibili dai vari federati. L'accesso alle classi istanziate dal RTI (ad esempio Manager.Federate) avviene con gli usuali meccanismi di sottoscrizione, così come per tutti gli altri dati contenuti negli usuali FOM.

3.8 Servizi offerti da HLA divisi per aree

HLA in veste di architettura distribuita fornisce ai singoli federati una serie di servizi differenziati a seconda delle aree e che hanno lo scopo di permettere l'attività di simulazione distribuita.

Passiamo in rassegna ognuna delle sei aree individuando ogni tipologia di servizio.

3.8.1 Federation Management

Si tratta dell'insieme di servizi che permettono la creazione di una federazione, l'ingresso o l'uscita da parte di un federato e la fine esecuzione. Ovviamente sono di importanza fondamentale e indispensabili per ogni simulazione. Oltre a questo scopo "primario" di gestione della vita di una federazione sono presenti altri servizi che hanno la caratteristica di riguardare la federazione nel suo complesso: punti di salvataggio e recupero di stato, gestione degli eventuali punti di sincronizzazione.

3.8.2 Declaration Management

Lo scambio di informazioni tra federati (sia quando avviene sotto forma di oggetti che di interazioni) non è diretto come si potrebbe immaginare, la struttura segue sempre le fasi di pubblicazione di informazioni e di sottoscrizione[6]. I servizi di declaration management danno la possibilità di pubblicare e sottoscrivere informazioni: attraverso questi servizi l'RTI è in grado di individuare le forme migliori per la loro comunicazione tra federati[7], per la trasformazione (ad esempio eliminando le parti non essenziali) e per individuare le aree di interesse.

3.8.3 Object Management

Lo scambio vero e proprio dei dati avviene attraverso questi servizi, ad esempio per spedire o ricevere interazioni, registrare nuove istanze di un oggetto o

[6] Anche le interazioni seguono questo procedimento, risulta chiaro analizzando la struttura gerarchica che le caratterizza. Solo utilizzando meccanismi di divisione in regioni è possibile interferire sul routing delle interazioni ma questo non modifica l'approccio pubblicazione/sottoscrizione.

[7] In modo quanto più efficiente possibile in base alla sua implementazione, lo standard non entra in questi particolari. Questo aspetto può condurre a grosse differenze prestazionali a seconda dell'implementazione utilizzata.

propagare le modifiche che sono state effettuate a degli attributi sotto la propria responsabilità; ovviamente sono coperti sia gli aspetti di generazione che di ricezione degli aggiornamenti.

3.8.4 Ownership Management

Come abbiamo già visto in precedenza gli attributi di un'entità non sono staticamente di proprietà di un federato, la responsabilità di mantenerne aggiornati gli stati è del proprietario. Le entità non devono necessariamente risiedere sotto il controllo di un unico federato, è possibile che la loro simulazione avvenga in modo congiunto: ogni federato si occupa di un insieme di attributi. Allo stesso modo durante un'esecuzione della federazione è comune che lo stesso attributo passi di proprietà più volte tra federati diversi. Questi aspetti di cambio di proprietà e responsabilità di simulazione sono del tutto dipendenti dall'architettura scelta per il caso specifico di simulazione: nulla vieta che si sia deciso di utilizzare un'architettura che non ha nessuna necessità di scambiare proprietà di attributi o creare entità simulate in modo congiunto da più federazioni.

3.8.5 Time Management

L'ordinamento temporale dei messaggi all'interno di un sistema distribuito è fondamentale al fine di creare simulazioni aventi un significato. Una parte fondamentale di servizi si occupa della gestione del tempo all'interno dei federati in modo che sia coerente rispetto al resto della federazione. La gestione temporale degli eventi è uno degli aspetti più delicati di tutta l'architettura HLA, il capitolo 4 ha lo scopo di chiarirne i concetti e mostrare la flessibilità con cui è stato definito.

3.8.6 Data Distribution Management

Si tratta dei servizi che permettono di interagire e modificare il meccanismo di pubblicazione e sottoscrizione delle entità da parte dei federati. Eventualmente creando "routing virtuali" particolari a seconda delle esigenze richieste dalla simulazione.