ALMA MATER STUDIORUM - UNIVERSITA' DI BOLOGNA
CAMPUS DI CESENA
SCUOLA DI SCIENZE

CORSO DI LAUREA IN SCIENZE E TECNOLOGIE INFORMATICHE

Analisi, progettazione e sviluppo di un modulo di gestione di messaggistica orientato alla sanità

Relazione finale in
Laboratorio di Basi di Dati

Relatore:
Chiar.mo Prof.
Matteo Golfarelli

Presentata da:
Silvia Sassi

Sessione II

Anno accademico 2012/2013

Indice

1
Introduzione

Le moderne organizzazioni sanitarie sono composte da molteplici sotto-sistemi autonomi, con numerose specializzazioni, sedi fisiche distribuite e principi di base spesso divergenti fra loro. Un sistema informativo è quindi composto da svariati componenti, più o meno indipendenti tra loro, sviluppati in tempi diversi per soddisfare bisogni diversi. I sistemi informativi sanitari trattano informazioni che spaziano dai risultati delle analisi di laboratorio all'amministrazione e, oltre a dover gestire informazioni al proprio interno, un sistema deve poter interagire con altri sistemi informativi distribuiti.

La gestione delle informazioni relative ai pazienti attraverso tali sistemi risulta essere un processo molto complesso, dovuto sia a differenze di tipo tecnologico, sia a una diversa rappresentazione dei dati dei pazienti, sia alla distribuzione in più strutture sanitarie dei dati di un singolo paziente. Risulta indispensabile, quindi, per un'azienda, disporre di un'applicativo che permetta di integrare i vecchi sistemi con quelli nuovi mantenendo stabile quanto già presente nell'azienda, ovvero che sia capace di far comunicare tra loro i diversi componenti di un sistema informativo salvaguardando il patrimonio informativo esistente.

Per poter far comunicare tra loro i diversi sistemi presenti e poter così integrare le loro funzionalità è necessario utilizzare un linguaggio comune per lo scambio di informazioni. Health Level 7 è un tentativo in corso dal 1987 teso a sviluppare un linguaggio comune per lo scambio di messaggi in ambito medico. Esso definisce un modello concettuale esplicito e condiviso per la creazione delle strutture dei messaggi in ambito sanitario.

L'obiettivo di questo lavoro di tesi è quello di realizzare una libreria che permetta ad applicazioni di diverso tipo e distribuite sulla rete di scambiarsi

messaggi con lo scopo di estendere le funzionalità di ciascuna applicazione con quelle fornite dalle altre applicazioni e di rendere virtuali le risorse aziendali, permettendo in questo modo alla logica di business del sistema sanitario di essere sviluppata e gestita indipendentemente dall'infrastruttura della rete e senza apportare modifiche alle applicazioni già sviluppate.

A tal fine verrà realizzato e definito un portale come strumento in grado di effettuare lo scambio di dati di interesse medico e l'integrazione di archivi provenienti da differenti enti locati sul territorio per creare un sistema unificato per lo scambio di informazioni.

Il progetto che si intende realizzare si muove all'interno dello standard HL7 e avrà come obiettivo la realizzazione di una libreria personalizzabile per la creazione e il parsing dei messaggi HL7. Integrando all'interno delle applicazione questa libreria sarà possibile tradurre e trasformare i messaggi in modo che le applicazioni vedano i dati trasmessi in un formato e in una struttura comprensibile, in questo modo verranno ridotti drasticamente i costi e il tempo necessari per apportare modifiche strutturali alle applicazioni già esistenti.

Data l'eterogeneità dei sistemi da integrare e delle tipologie di dati da scambiare risulta, quindi, indispensabile per la software house avere a disposizione un modulo altamente configurabile, affidabile, efficiente per la gestione di tali messaggi. Il modulo consente lo scambio bidirezionale di messaggi tra sistemi gestendo diversi formati e diverse modalità di invio/ricezione ed è integrabile con gli altri moduli dell'azienda.

2

Ricerca di informazioni con codifica in HL7

2.1 Situazione settore sanitario

Negli ultimi anni il settore sanitario sta vivendo profonde trasformazioni strutturali e organizzative [1]. Il Servizio sanitario nazionale (Ssn) è l'istituto attraverso il quale lo stato garantisce ai propri cittadini il diritto alla salute, a seguito di varie riforme, che hanno portato a profonde trasformazioni, oggi le funzioni e le risorse sanitarie sono state delegate alla Regione che le esercita attraverso le Aziende sanitarie locali. Il Ssn non è un ente pubblico autonomo ma una complessa articolazione di istituzioni, strutture, funzioni e risorse che interagiscono al fine di garantire le migliori condizioni di salute alla popolazione.

E' emerso un nuovo paradigma sull'Information and Communication Technology (ICT) in sanità, basato sulla condivisione di informazioni strutturate tra operatori sanitari e sull'integrazione di informazioni cliniche, organizzative e amministrative. L'evoluzione delle tecnologie dell'informazione si è fatta sempre più incalzante e pervasiva, passando dalle applicazioni isolate allo scambio di messaggi standard strutturati e poi a sistemi informativi integrati in organizzazioni virtuali.

Nel corso dell'ultimo decennio le regioni si sono dotate di strutture e hanno adottato modelli organizzativi assai diversi per governare i propri sistemi sanitari. Il problema centrale della sanità riguarda oggi l'integrazione tra i relativi sottosistemi, sia per la gestione dei dati clinici, gestionali ed economici che per il supporto decisionale nelle Aziende Sanitarie e Ospedaliere.

In quest'ottica assume un ruolo centrale l'automazione delle transazioni tra operatori e strutture sanitarie perché esse rappresentano il cuore gestionale del processo clinico. Il loro trattamento elettronico, oltre all'ovvia velocizzazione e all'aumento di efficienza, consente infatti di acquisire i dati del processo direttamente al momento della produzione dell'informazione. Tali dati vengono attualmente prodotti e gestiti da numerosi sottosistemi informativi amministrativi, gestionali e clinici, che ricadono sotto molteplici soggetti locali.

In questo scenario si è reso necessari realizzare un'infrastruttura tecnologica, una infrastruttura informativa e dei servizi di base che favoriscano l'armonizzazione dei sottosistemi locali e la loro integrazione. L'uso di una serie di standard sulle transazioni elettroniche tra applicazioni sanitarie è il punto di partenza per disporre di informazioni attendibili e tempestive e quindi per migliorare la qualità dell'assistenza, grazie ad una migliore gestione di informazioni e conoscenze. Lo standard più utilizzato nell'ambito sanitario è l'HL7, che andremo ora ad analizzare.

2.2 Health Level 7

Health Level Seven (HL7) [2] è uno degli standard funzionali di riferimento per la sanità. Lo standard è stato sviluppato inizialmente per il sistema sanitario degli Stati Uniti ed ha accumulato una notevole esperienza grazie all'utilizzo quotidiano nella maggior parte degli istituti ospedalieri.

HL7 sta coordinando gli sforzi con altri organismi internazionali di standardizzazione, coinvolgendo l'intera comunità internazionale. In parallelo con queste collaborazioni formali l'HL7 sta incoraggiando la formazione di sezioni nazionali affiliate in tutto il mondo.

Il progetto HL7 prende il via nel marzo 1987 con l'obiettivo di semplificare le interfacce fra i diversi applicativi sanitari e si incentra sulla standardizzazione dei formati per lo scambio di alcuni gruppi di dati considerati comuni ad ogni sistema di tipo sanitario. Dal 1994 l'HL7 è una organizzazione per lo sviluppo di standard accreditata presso l'ANSI (American National Standards Institute) e partecipa all'HISB (Health Information Standards Board).

Il termine "LEVEL 7" fa riferimento al livello più alto del modello OSI (Open System Interconnection), il livello applicativo. Ciò indica che HL7 definisce concettualmente una interfaccia di tipo paritario (applicazione-applicazione) posta al settimo livello del modello OSI. In altre parole HL7 fa riferimento ai dati scambiati, alla tempistica degli scambi e alla comunicazione di errori fra le applicazioni.

Lo standard definisce quindi un protocollo di scambio di messaggi e si prefigge lo scopo di regolare lo scambio di informazioni all'interno di una stessa struttura ospedaliera e di permettere la trasmissione di ogni tipo di documento dell'ambito sanitario (informazioni cliniche, organizzative, finanziarie, amministrative) riportando ogni informazione testuale o numerica relativa all'indagine diagnostica ad esclusione delle immagini, anche se lo standard non impedisce di scambiare informazioni relative ad immagini diagnostiche sotto forma di dati incapsulati quando tali informazioni possano essere ricondotte ad un unico messaggio. HL7 si prefigge, inoltre, di definire il formato dei messaggi e la formalizzazione di tutti gli eventi, detti anche trigger.

I principali obiettivi dell'HL7 sono:

- mettere a disposizione formati e protocolli per lo scambio di informazioni tra sistemi informatici sanitari;
- definizione dei formati dei messaggi scambiati per tutti gli ambienti legati alla medicina;
- migliorare l'efficienza della comunicazione;
- minimizzare il numero di interfacce riducendo al minimo le spese per la loro attuazione;
- possibilità di interfacciare sistemi proprietari diversi.

Nel corso degli anni ovviamente lo standard si è evoluto attraverso una serie di versioni, ed attualmente coesistono la versione 2, giunta alla nona revisione, e la versione 3, nata per superare le limitazioni della precedente versione.

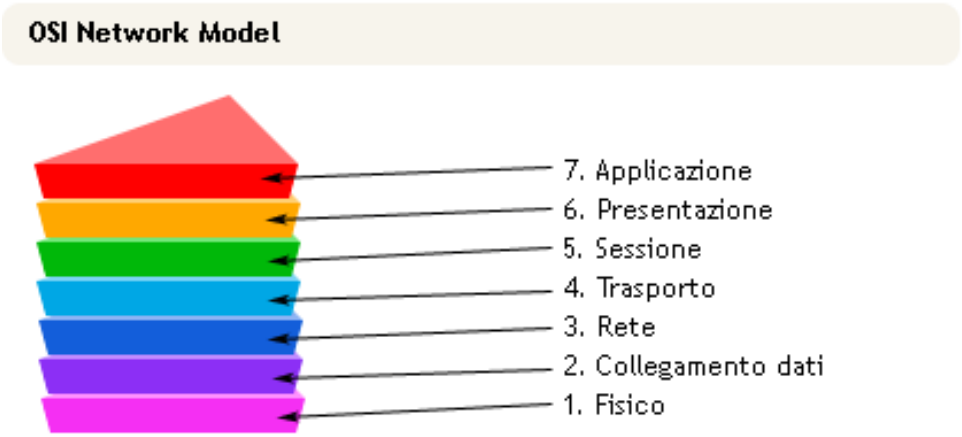

Figura 2.1. Schema illustrante la posizione dei protocolli HL7 nello stack ISO OSI

2.2.1 HL7 v.2

La versione 2 dello standard HL7, la cui approvazione presso l'ANSI risale alla fine degli anni 90, ha riscosso un ampio successo ed è attualmente implementato a livello capillare negli Stati Uniti e in un folto numero di organizzazioni del mondo dell'healthcare. Lo standard definisce le regole di codifica, la struttura e i tipi di dato ammissibili per la costruzione di messaggi per lo scambio di informazioni in formato elettronico nel contesto ospedaliero.

Questa versione prevede l'interscambio di informazioni attraverso l'utilizzo di un set predefinito di messaggi che devono essere codificati all'interno di testo ASCII delimitato, chiamato anche formato pipe [3]. Ciò rende i messaggi HL7 non molto differenti da insiemi di dati inseriti dentro un semplicissimo file CSV

(Comma Separated Value). Oggi una scelta del genere può sembrare anacronistica, ma bisogna tenere conto del contesto storico in cui operavano gli sviluppatori alla fine degli anni 80, dove il linguaggio XML (eXtensible Markup Language, quello che più si presta allo scopo preposto) non era ancora stato ideato. Dovendo quindi garantire l'interoperabilità fra applicativi eseguiti su sistemi molto eterogenei, la scelta adottata si dimostrò quanto di meglio era possibile ottenere con i mezzi dell'epoca.

Il paradigma centrale del protocollo è lo scambio di messaggi tra sistemi che necessitavo di scambiarsi informazioni. Un messaggio [4] è l'unità atomica di dati trasferiti tra applicazioni ed è testo ASCII delimitato da una serie di caratteri speciali e che deve rispettare una struttura molto precisa, per struttura del messaggio si intende le regole di formattazione utilizzate e la disposizione dei dati. Ogni messaggio è definito da un Message Type che identifica la struttura del messaggio. Ad esempio il tipo Messaggio ADT è utilizzato per trasmettere i dati anagrafici di un paziente da un sistema a un altro.

I messaggi in formato HL7 v2 sono usati per trasferite dati elettronici tra sistemi sanitari diversi. Un messaggio HL7 consiste di uno o più segmenti di lunghezza variabile posti in un preciso ordine. Ogni segmento è identificato da un codice univoco di tre caratteri conosciuto come l'identificativo di segmento, ad esempio: informazioni sui parenti (NK1), i dati demografici del paziente (PID) e le informazioni sulla visita (PV1). Tutti i messaggi hanno come primo segmento obbligatorio MSH che dà informazioni sull'applicazione chiamante, su quella ricevente, sull'evento scatenante, sulla versione del protocollo usato e altre informazioni.

I segmenti vengono trasmessi come sequenze di caratteri separati da un carattere speciale chiamato separatore di segmento. Nella costruzione di un messaggio vengono utilizzati alcuni caratteri speciali per delimitare le varie componenti del messaggio: il terminatore di segmento, il separatore di campo, il separatore di componente, il separatore di sottocomponente, il separatore di ripetizione e il carattere di chiusura.

Il terminatore di segmento è sempre un carriage return (in ASCII, 0D in esadecimale). Gli altri delimitatori sono generalmente definiti nel segmento MSH, con il delimitatore di campo nella posizione del 4° carattere. Gli altri delimitatori vengono indicati nel primo campo dopo l'identificativo del segmento e sono utilizzati nell'intero messaggio.

Delimitatore	Valore suggerito	Posizione del carattere	Uso
Terminatore di Segmento	<cr>	-	Termina un segmento. Questo valore non può esser e cambiato
Separatore di Campo	|	-	Separa due campi adiacenti in un segmento. Allo stesso modo separa l'identificativo del segmento dal primo campo dati in ogni segmento
Separatore di Componente	^	1	Separatore per quando sono presenti due componenti adiacenti in un campo data
Separatore di Subcomponente	&	4	Separa due subcomponenti adiacenti in un campo data quando sono presenti. Se non sono presenti subcomponenti in un campo questo carattere può essere omesso.
Separatore di Ripetizione	~	2	Separa occorrenze multiple di un campo
Carattere di Escape	\	3	Carattere di Escape usato in tutti i campi con data type ST, TX , FT, o con il quarto componente del tipo ED. Se non viene utilizzato in un messaggio questo carattere può essere omesso.

Figura 2.2. I delimitatori

All’interno dell’HL7 sono definiti dei segmenti ripetibili, ossia che possono essere presenti più volte all'interno di un messaggio. La ripetizione di segmenti si applica nei casi in cui sia necessario ripetere più campi ma che devono poter essere identificati in uno stesso raggruppamento; come ad esempio il segmento NK1 o AL1 che si riferiscono alle informazioni sulla parentela o alle allergie.

Ogni segmento consiste di uno o più compositi, noti anche come campi. Un segmento è composto essenzialmente da campi dato di lunghezza variabile e separati da un carattere separatore, “|”.

L'HL7 è order-sensitive, i campi dati sono individuati nel messaggio dalla posizione che hanno nel segmento associato.

Alcuni campi possono ripetersi all'interno di un messaggio o essere presenti solo una volta, e possono assumere significati diversi a seconda delle necessità, per esempio il caso di identificativi che si riferiscono ad una stessa anagrafica (ID dipartimentale, ID unico, ID fiscale, ID SSN).

Un campo è composto, a sua volta, da componenti, anche i componenti possono ripetersi all'interno del campo. Generalmente questo tipo di situazione viene negoziata con l'applicazione destinataria.

```
MSH|^~\&|SAGO|SAGO|G2|INSIEL|20131001011358||ADT^A31|a2gy3ua663dvhm8bdjyy|P|2.5
EVN|A31|20131001011357||||20131001011357
PID|||00576081^^^PK~^^^PK_CUP||ROCCA^ CESARE^^^^^L||19701003|M|||VIA VARESE N
7^^ROMA^^^^L^^058091~^^^^^^H~^^ROMA^^^^N^^058091||^^^^^^^^^^^064460025|||||RCCCSR70R
03H501N|RCCCSR70R03H501N|||||||100^ITALIA|||||N|PRO-CER
PV1||O
```

Figura 2.3. Esempio di messaggio HL7 v.2

Un aspetto importante dell'HL7 è il suo protocollo di riconoscimento. Questo protocollo si basa sul fatto che un'applicazione invia un messaggio di risposta all'applicazione client ogni volta che riceve un messaggio e ne consulta i dati. Lo stato del messaggio inviato non viene aggiornato fino a quando l'applicazione client non ha ricevuto un messaggio ACK di risposta. E' necessario sapere che l'applicazione ricevente ha ricevuto ed elaborato i dati ricevuti con successo, in altre parole non è sufficiente sapere che il sistema di trasmissione sottostante ha garantito la consegna del messaggio. Controllando lo stato del messaggio si evita la possibilità di perdere i dati durante la comunicazione.

Un messaggio ACK è costituito da due segmenti, il segmento MSH che definisce lo scopo, l'applicazione inviante, l'applicazione di destinazione e alcune particolarità della sintassi di un messaggio, e il segmento MSA contenente il codice della risposta e le informazioni per riconoscere il messaggio principale.

Il concetto chiave nel protocollo ACK è l'identificativo del messaggio. Questo è un identificativo univoco che ogni messaggio HL7 ha nel campo 10 del suo segmento MSH. Un HL7 ACK valido ripeterà questo identificativo nel secondo campo del suo segmento MSA.

L'applicazione client invia il messaggio principale e riceve il messaggio di riconoscimento, mentre il sistema ricevente riceve il messaggio principale e dopo averli processati invia il messaggio di risposta.

Le risposte possibili sono:

- il messaggio è stato processato con successo; viene generato un messaggio di risposta con il valore "AA";
- errore nel processare il messaggio; viene generato un messaggio che provvede a informare del tipo di errore il sistema includendo il valore "AE" nel campo MSA;
- errore nel processare il messaggio per ragioni non correlati al contenuto o al formato del messaggio, per momentanea indisponibilità del DataBase, errori interni al sistema, ecc. L'applicazione ricevente richiede alla applicazione mittente di rinviare il messaggio in tempi successivi con il valore "AR".

I limiti della versione

La versione 2 di HL7 è sicuramente la più diffusa e implementata, ma ciò non significa che non presenti limiti ne che sia immune da critiche dovute principalmente alla presenza di norme non particolarmente rigide per quanto riguarda modifiche o aggiunte al formato. Ciò significa che nel contesto produttivo i vari produttori usano un ibrido che mette insieme caratteristiche provenienti da varie versioni del protocollo (in particolare quelle dalla 2.1 alla 2.5). Inoltre spesso vengono inserite personalizzazioni ai messaggi tali da rendere spesso più difficoltoso il processo di parsing fra i diversi sistemi HL7.

Mentre l’opzionalità da un lato ha il vantaggio di dare più libertà ad un messaggio, rendendolo utilizzabile in più contesti, facilitandone la diffusione, dall’altro rende difficile stabilire le specifiche delle interfacce dello standard, causando incomprensioni tra fornitore e acquirente. Troppi campi opzionali e non formalizzati hanno portato a una libera interpretazione delle informazioni presenti nei messaggi dello standard. All’interno dello standard non sono presenti delle linee guida formalizzate per la costruzione dei messaggi, i dati sono spesso descritti in linguaggio naturale, e le relazioni tra di essi non sono chiare perché l’attenzione viene focalizzata sulla struttura sintattica del messaggio e non vengono esplicitate le relazioni semantiche tra le informazioni contenute.

Molti di questi difetti, come vedremo, sono stati risolti in HL7 v3 seppur a costo dell'assenza di retrocompatibilità con HL7 v2.

2.2.2 HL7 v.3

Alla fine degli anni 90 si iniziò a constatare che il protocollo v2, per quanto di successo, iniziasse a palesare una serie di debolezze difficilmente risolvibili con semplici modifiche. Per tanto venne pianificata la realizzazione della v3 che apportasse un radicale cambiamento nel processo di sviluppo delle specifiche, formalizzando una metodologia che nelle versioni precedenti era ambigua.

La prima grande differenza rispetto alla versione 2 riguarda il rapporto fra informazioni e dati: nella v2 le uniche regole riguardavano sequenze di segmenti e l'eventuale presenza dei valori all'interno dei composite, nella v3 è stato dato un modello alla struttura delle informazioni basato sul paradigma a oggetti basandosi sullo standard Unified Modeling Language (UML).

Questo cambiamento aumenta il dettaglio, la chiarezza e la precisione delle specifiche, favorendo sia gli attori che presiedono lo sviluppo dello standard, sia gli sviluppatori che debbano implementare lo standard stesso. Ogni informazione rappresenta un oggetto che possiede i propri attributi e che mantiene relazioni con altre classi (ossia altre tipologie d'informazione). Questo modello dei dati è noto in letteratura come HL7 Reference Information Model (di solito indicato dall'acronimo RIM).

Un altra grossa differenza riguarda il formato con cui i dati vengono trasportati, poiché si è passati dall'uso del testo delimitato nella versione 2 all'utilizzo dell'XML, in quanto linguaggio molto più adatto, consono e standardizzato per la condivisione e la trasmissione dei dati.

RIM

Il RIM [5] è il modello informativo tramite il quale gli standard della famiglia HL7 v3 devono trarre il modello relativo al contenuto delle informazioni.

Un modello informativo serve per esplicitare il significato dell’informazione che si vuole trattare. All’interno del RIM non viene definito in maniera concreta il formato delle informazioni, bensì un modello astratto delle stesse. Esprime classi e proprietà, quali attributi, vincoli, relazioni e stati, delle classi di informazione da rappresentare. Associato così un contenuto semantico ai dati

presenti all'interno di un messaggio inviato verso una applicazione. Ad esempio, una applicazione invia con un messaggio un dato corrispondente al concetto di uomo, inteso come persona (dunque l'unione dei concetti di uomo e donna), e l'applicazione che riceve il messaggio intende il concetto di uomo come distinto dal concetto di donna.

Il modello delle informazioni ha una struttura a oggetti, quindi esistono diagrammi UML (Unified Modeling Language) che lo schematizzano, la struttura fondamentale del RIM è composta da un insieme di sei classi, ovvero:

- Entity: definisce un oggetto fisico o una persona che prende parte al processo descritto;
- Act: sono tutte le possibili azioni che può svolgere un entities;
- Role: ruolo che ricoprono gli act nell'universo;
- Participation: definisce il contesto di una azione, nei termini di chi la esegue, chi la subisce ed il motivo per cui viene eseguita;
- ActRelationship: rappresenta l'insieme delle azioni minori nelle quali un'act può dover essere scomposto;
- RoleLink: indica se alcuni ruoli esistono nell'organizzazione in quanto collegati ad altri ruoli. Ad esempio un medico può esercitare nella clinica X solo se ne è un lavoratore dipendente.

Le relazioni fra questi sei componenti fondamentali sono illustrati dal seguente diagramma illustrante il meta-modello dei dati:

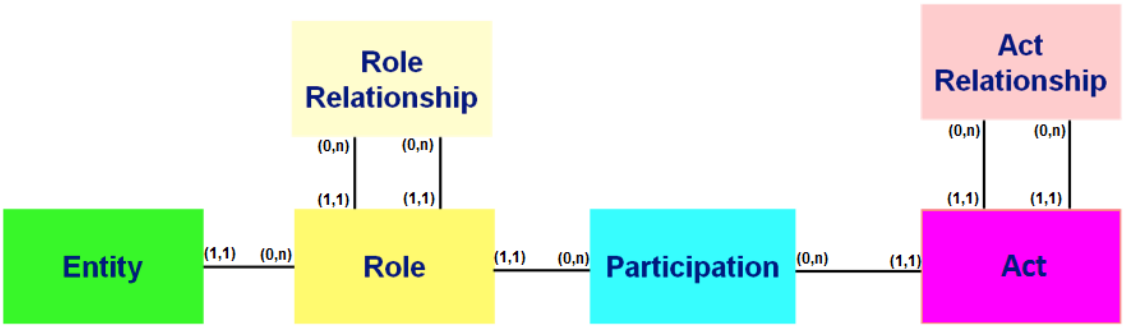

Figura 2.4. Meta-modello ridotto delle foundation classes in HL7 v3

Alcune di queste classi, nello specifico Act, Entity e Role, possono essere ulteriormente specializzate da una serie di sottoclassi, seguendo la regola che una sottoclasse può essere creata solo ed esclusivamente nel caso in cui ad essa si applichi un'associazione o un attributo che non si possa applicare anche alla classe padre.

In ambito HL7 v3, due esempi reali di modelli dati derivati dal RIM sono i D-MIM e il R-MIM:

- D-MIM (Domain Message Information Model): è un sottoinsieme del RIM che include una serie di classi e attributi utilizzabili per creare messaggi in un dominio specifico;
- R-MIM (Refined Message Information Model): è un sottoinsieme del D-MIM (quindi transitivamente del RIM) utilizzato per esprimere il contenuto informativo di un messaggio. Viene usato anche per rappresentare il contenuto informativo di strutture astratte dei messaggi chiamate Hierarchical Message Definition (HMD).

Messaggi HL7 v3

In HL7 v3 l'aspetto fondamentale del messaging, è dato dalle interazioni. Una iterazione è un'associazione univoca tra tipi di messaggi (trasferimento di informazioni), un particolare evento scatenante che inizia (la causa del trasferimento) e le responsabilità del ricevente. Ogni interazione è quindi legata ad un tipo di messaggio, un evento scatenante e delle responsabilità del ricevente.

Un'interazione da sola però non è sufficiente a innescare il trasferimento di informazioni: per fare ciò è necessario che avvenga un evento noto in letteratura come evento trigger. Questi eventi rappresentano situazioni reali come ad esempio quando un paziente è ammesso in ospedale e viene preso in carico da un applicativo di front-office (PS, Laboratorio di analisi, CUP) i suoi dati anagrafici (codice fiscale, nome, cognome, data di nascita etc.) devono automaticamente essere registrati anche nel sistema che amministra i pazienti (Anagrafica Centralizzata). L'evento "Ammissione paziente" da inizio allo scambio di dati tra il sistema di front-office che prende in carico il paziente e il sistema informativo ricevente (Anagrafica Centralizzata), il quale necessita della conoscenza dei dati anagrafici per poter completare lo scenario previsto.

In pratica l'HL7 implementa un elenco di eventi disponibili che rappresentano tutte le possibili richieste che un'applicazione può fare. Il protocollo prevede che ogni evento venga definito da un nome, un identificativo, la tipologia e una descrizione.

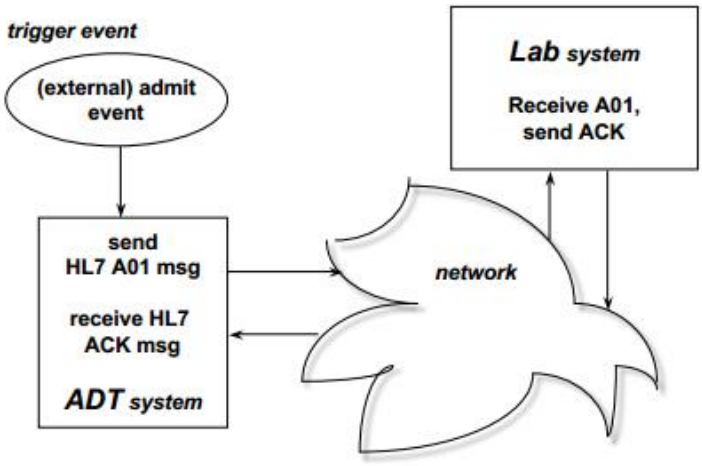

Figura 2.5. Funzionamento HL7 v.3

Confronto con la versione precedente

La differenza maggiore tra i due standard, ovvero il formato di comunicazione, è stata assottigliata molto in seguito all'aggiunta dell'XML. L'XML è stato aggiunto durante lo sviluppo dello standard HL7 senza modifiche alle metodologie di modellazione o ai contenuti, a dimostrazione della flessibilità dell'approccio con il quale lo standard viene realizzato; mentre la versione 2 è stata adattata all'XML la versione 3 supporta pienamente le capacità espressive dell'XML.

Nella versione 3 le informazioni vengono rappresentate come Schemi XML con le relazioni logiche tra le informazioni e i nomi degli elementi che si riferiscono direttamente ai modelli HL7 e quindi con i concetti che gli analisti ed i programmatori devono comprendere per relazionare i messaggi v3 con i propri sistemi informativi. HL7 v3 è molto più robusta e versatile rispetto alla precedente v2. Infatti HL7 v2 fornisce solo le regole per formare un messaggio nel modo giusto e per implementare il messaging, mentre HL7 v3 mette a disposizione anche schemi XML ossia un modello relativo al formato che le informazioni dovranno avere all'interno del processo di comunicazione in modo da consentire molta più estensibilità e flessibilità a un protocollo che nella sua v2 ha dimostrato di risentire delle aggiunte fatte da soggetti esterni.

Da non dimenticare, nonostante i grandi vantaggi portati, che HL7 v3 è intrinsecamente incompatibile con la precedente v2, dato che le differenze architetturali fra i due protocolli sono pressoché incolmabili. Questa considerazione è il grosso motivo per cui oggi il mercato non recepisce ancora in maniera massiccia i benefici di HL7 v3, poiché, la grandissima maggioranza

degli HIS, Hospital Information System, attualmente presenti che adottano in maniera massiccia sistemi basati su HL7 v2 dovrebbero essere riconvertiti. Questo processo, oltre a richiedere costi molto alti e ammortizzabili solo in tempi molto lunghi, comporterebbe anche tempi molto lunghi e difficilmente accettabili. In ogni caso i due sistemi, pur non essendo interoperabili possono tranquillamente coesistere poiché il protocollo usato per la trasmissione dei dati è una cosa molto differente dal formato con cui essi vengono successivamente memorizzati all'intero dei database degli HIS.

2.3 Casi d’uso

Per lo studio dell'applicazione sono stati analizzati una serie di casi clinici reali ai quali corrisponde l'utilizzo del protocollo HL7 allo scopo di fornire un esempio reale dell'uso degli stessi:

- ammissione, dimissione, trasferimento: i messaggi del tipo ADT (Admission, Discharge, Transfer - Ammissione, Dimissione, Trasferimento) permettono lo scambio dei dati anagrafici di un paziente e dei dati relativi ad un accesso alla struttura, che può essere per una visita ambulatoriale o per un ricovero. Attraverso la definizione di una serie di eventi e dei relativi messaggi è possibile per il sistema ricevente trattare in modo adeguato tutti i casi possibili;
- trasmissione dati sanitari: l'oggetto di questa parte dello standard è la trasmissione di dati di natura clinica, un uso frequente di queste transazioni è lo scambio di dati fra sistemi dove i dati vengono prodotti (sistemi informativi diagnostici) e sistemi di altro tipo (ad esempio sistemi di gestione della cartella clinica di reparto o di analisi). L'unità minima trasmissibile è una registrazione che può a sua volta comprendere o un singolo dato clinico o l'intera storia clinica del paziente: le informazioni sono comunicate in maniera strutturata il che significa che le singole registrazioni vengono mantenute come entità separate all'interno di un unico contenitore che è la registrazione che le raggruppa;
- scambio delle referenze di un paziente tra diverse organizzazioni sanitarie: ovvero definizione delle modalità per lo scambio di informazioni fra diverse organizzazioni sanitarie in merito ai pazienti. Tali organizzazioni possono spaziare dal medico di base, agli specialisti, agli ospedali, ecc.. In questo particolare contesto occorre tener conto che le due controparti conoscono ben poco riguardo alle informazioni necessarie per identificare il paziente e occorre pertanto presupporre che il ricevente necessiti di un intero insieme di informazioni anagrafiche. Nel caso il ricevente abbia già in archivio il paziente occorre garantire comunque nello scambio un insieme sufficiente di informazioni per permettere al ricevente di associare le informazioni già presente in archivio e le informazioni ricevute;

- trasmissione ordini: un ordine è genericamente una richiesta di materiale o di un servizio, solitamente riferito o riferibile ad un paziente: ad esempio oggetto di una richiesta possono essere farmaci, test di laboratorio, pasti, indagini diagnostiche in genere, biancheria. Sebbene la maggior parte degli ordini siano riferiti ad un paziente, lo standard prevede anche l'effettuazione di ordini che non hanno tale riferimento, come ad esempio gli ordini di reparto.

  Nell'ordine devono essere specificati il richiedente e il fornitore del materiale. Il messaggio principale (General Order Message - ORM) serve per trasmettere le informazioni relative all'ordine e fornisce le informazioni per creare nuovi ordini, cancellare ordini esistenti, bloccare l'esecuzione di un ordine o terminare l'esecuzione di un ordine già in esecuzione. Oltre ai messaggi per l'effettuazione dell'ordine e per la conferma della ricezione dell'ordine sono presenti anche i messaggi per l'interrogazione del sistema circa lo stato di un ordine e il messaggio per la relativa risposta.

  Lo standard permette inoltre di specificare quando il servizio deve essere fornito e in quali quantità: è possibile specificare in maniera assai precisa la temporizzazione che può essere funzione anche delle modalità organizzative dell'ente, ad esempio è possibile specificare che la somministrazione di un farmaco deve essere fatta due volte al giorno alle ore canoniche che l'organizzazione stabilisce per la somministrazione dei farmaci ad un paziente.

3
Le tecnologie

Per la realizzazione dell'applicativo di messaggistica sono state sfruttate diverse tecnologie a seconda delle esigenze. Per lo scambio di messaggi l'applicazione può fruttare le socket per poter condividere e ricevere informazioni attraverso la rete, oppure utilizzare i web service per supportare l'interoperabilità con i diversi sistemi distribuiti sulla rete, l'XML per la creazione dei messaggi e un database MySQL per il salvataggio dei dati da inserire all'interno dei messaggi.

Passiamo ora a descrivere brevemente quali sono le tecnologie usate.

3.1 Socket

Il paradigma socket [6] viene presentato per la prima volta nella Berkeley Software Distribution (BSD) della University of California a Berkeley.

Le Socket sono un meccanismo che permette la comunicazione in rete di due programmi, indipendente dal dispositivo fisico tramite il quale viene effettuata la connessione. Una socket è come una porta di comunicazione e non è molto diversa da una presa elettrica, da cui prende appunto il nome: tutti i dispositivi in grado di comunicare tramite il protocollo standard TCP/IP possono effettuare una connessione e comunicare tramite le socket, allo stesso modo in cui un qualsiasi apparecchio che funzioni a corrente può collegarsi a una presa elettrica e sfruttare la tensione messa a disposizione. Nel caso delle socket, invece dell'elettricità, nella rete viaggiano pacchetti TCP/IP. Tale protocollo e le socket permettono quindi di far comunicare dispositivi diversi che utilizzano lo stesso protocollo rendendoli indipendenti dalla piattaforma su cui sono sviluppati.

Per l'utilizzo delle socket Java mette a disposizione alcune classi. Usando queste classi un client può stabilire un canale di comunicazione con un host remoto. Si potrà comunicare attraverso questo canale utilizzando stream particolari, specializzati per le socket.

Un client, per comunicare con un host remoto usando il protocollo TCP/IP, dovrà innanzitutto creare una Socket specifica per comunicare con tale host, per farlo dovrà specificare l'indirizzo IP dell'host e il numero di porta sulla quale sarà presente un host remoto con un server in ascolto.

Un server, che rimane in attesa di connessioni su tale porta, ogni volta che un client effettua una connessione crea una socket dedicata tramite la quale il server potrà accettare connessioni dal client e scambiare informazioni attraverso la rete. Dopo aver creato un oggetto Socket per ogni connessione il server potrà poi comunicare, come fa un client, estraendo gli stream di input ed output dalla socket.

I passi tipici di un server sono:

- creare un oggetto della classe ServerSocket specificando un numero di porta locale, cioè la porta in cui il server rimarrà in ascolto di richieste di connessioni;
- attendere connessioni dai client;
- usare la socket ottenuta ad ogni connessione, per comunicare col client.

3.2 Web service

Un Web service [7] è una applicazione modulare descritta, pubblicata, individuata ed utilizzata attraverso il web. Un Web service è, essenzialmente, un'interfaccia che descrive un insieme di funzioni invocabili da remoto attraverso un sistema di messaggistica. Le funzioni rese disponibili possono spaziare da una semplice richiesta di informazioni ad un complicato processo commerciale, possono essere sviluppati internamente o pubblicati all'esterno, chiamate da altre applicazioni o da altri Web services, a livello locale oppure da qualche parte su Internet e possono essere sfruttate contemporaneamente per più operazioni essendo condiviso in rete e disponibile a tutti.

La tecnologia dei Web service ha permesso di definire un sistema indipendente dalle piattaforma e dai linguaggi di programmazione utilizzati, fornendo una via estremamente versatile e semplice per far comunicare componenti software attraverso la rete. E' proprio grazie alla descrizione astratta dei loro servizi e all'indipendenza da architetture software/hardware particolari che è possibile far comunicare client e server basati su tecnologie e linguaggi differenti. I Web service forniscono i mezzi standard di interoperabilità tra diverse applicazioni software in esecuzione su una varietà di piattaforme. Una volta sviluppati sono in grado di eseguire qualsiasi funzionalità e possono, potenzialmente, unire sistemi eterogenei di ogni tipo.

Per analizzare l'architettura e gli attori coinvolti nella tecnologia dei Web service, individuiamo i tre ruoli principali: il Service Provider, il Service Consumer, il Service Registry.

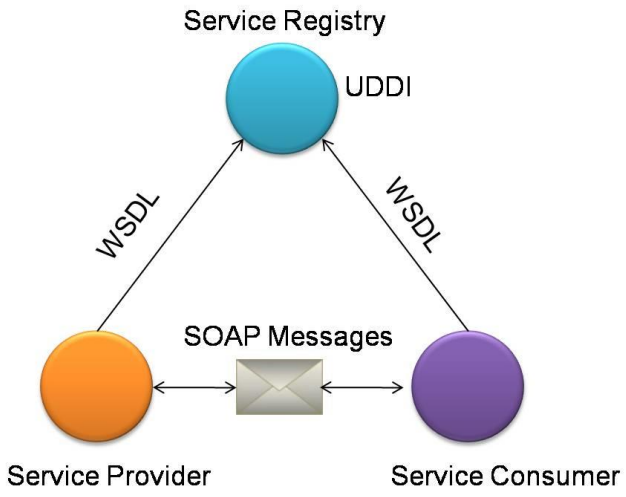

Figura 3.1. Gli attori principali dei Web Services

Il Service Provider realizza e mette a disposizione un Web service e il suo file descrittore (WSDL), inoltre si occupa di pubblicare, e quindi rende visibile sulla rete, la descrizione dei servizi disponibili che, in questo modo, possono essere trovati da altre entità che vogliono utilizzarli. Una volta pubblicato, un potenziale Service Consumer può trovare il servizio interrogando a sua volta il Service Registry ed utilizzando l'interfaccia UDDI.

UDDI, ovvero Universal Description, Discovery and Integration, definisce un meccanismo comune per la scoperta e l'accesso ai Web services, mantiene le informazioni relative ai servizi come l'URL e le modalità di accesso. Lo standard UDDI definisce i protocolli per poter accedere ad un registry di Web service, fornisce i metodi per controllare l'accesso a questo registro e un meccanismo per distribuire o delegare i dati, presenti in esso, verso altri registri. In pratica un registro UDDI fornisce un approccio standardizzato per consentire la localizzazione di un servizio software per invocare le operazioni rese disponibili e per gestirne i metadati associati.

Un Web service mette a disposizione la descrizione delle sue funzionalità pubbliche che sono esplicitate mediante un altro standard, basato su XML, che viene chiamato il descrittore del servizio. Il Service Registry, se in grado, soddisfa la richiesta del Service Consumer fornendogli il descrittore espresso in formato WSDL e un URL al quale contattare il Web service vero e proprio. Il descrittore fornisce tutti i dettagli necessari per l'interazione con il servizio: il formato dei messaggi, i tipi di dato utilizzati, protocolli di trasporto, l'URL al quale è possibile contattare il servizio e il protocollo di trasporto da utilizzare.

Il descrittore è espresso mediante il formato WSDL (Web Services Description Language). Esso descrive l'interfaccia esterna di un Web service affinché uno sviluppatore possa creare un client capace di accedervi. E' un linguaggio basato su XML usato per descrivere, in modo completo, un Web service. Più precisamente un documento WSDL fornisce informazioni riguardanti l'interfaccia del Web service in termini di URL ad essi associato, servizi offerti dal WS, modi per invocarlo, argomenti accettai in ingresso e modalità con cui debbano essere passati, formato dei risultati restituiti e formato dei messaggi. In altre parole si può dire che un file WSDL fornisce la descrizione relativa ad un Web service in termini di cosa fa, come comunica e dove si trova.

Il Service Consumer rappresenta un potenziale utente che interagisce con il Service Registry per ottenere il servizio più adatto ai propri obbiettivi. Esso ottiene il file descrittore e l'URL al quale contattare il Web service, tramite i quali

è in grado di effettuare la connessione e iniziare ad invocare le operazioni remote messe a disposizione utilizzando le giuste convenzioni (nome metodo e parametri) come descritte nel suo descrittore WSDL. A questo punto le operazioni di richiesta-risposta tra Service Consumer e Web service avvengono mediante lo scambio di messaggi espressi secondo le specifiche definite da SOAP.

Simple Object Access Protocol, SOAP, è un protocollo specifico per lo scambio di informazioni strutturate in formato XML all'interno di Web service grazie a una varietà di sottostanti protocolli di comunicazione. E' un protocollo di alto livello ed è completamente indipendente dal protocollo di trasmissione sottostante, che può essere indifferentemente HTTP, JMS, SMTP, MIME o altri, tra questi il più usato è HTTP. E' stato progettato per rimanere indipendente da un particolare modello di programmazione o da altre specifiche semantiche d'implementazione

SOAP mette a disposizione un meccanismo semplice, ma allo stesso tempo solido, che permette a una applicazione di mandare un messaggio ad un'altra applicazione.

3.3 XML

L'Extensible Markup Language (XML) [8] è un metalinguaggio che permette di creare dei linguaggi personalizzati di markup; è un formato di testo semplice, molto flessibile che offre un formato completamente estendibile, di facile apprendimento e molto ricco, per la strutturazione di dati e documenti che possono essere scambiati sul Web.

Il formato XML, in quanto standard per la descrizione di dati aperto, ha reso possibile la creazione di una rete Internet programmabile. Così come il formato HTML fornisce un linguaggio standardizzato per visualizzare informazioni su un'ampia varietà di piattaforme e il formato TCP/IP fornisce connettività universale per Internet, il formato XML fornisce un linguaggio standardizzato per lo scambio di dati per l'utilizzo automatizzato grazie alla sua capacità di rappresentare i dati in un formato ampiamente accettato che consente alle applicazioni di inviare e ricevere dati in un stile prevedibile, abilitando una programmabilità che si estende oltre i sistemi chiusi e controllati. XML contiene tutta la potenza, ma non tutte le complesse funzioni raramente utilizzate del SGML (Standard Generalized Markup Language), lo standard internazionale da cui è derivato l'XML. Grazie alla sua semplicità ed estensibilità l'XML consente di definire praticamente qualsiasi cosa, lasciando spazio per l'espansione.

Questo linguaggio nasce dall'esigenza di permettere agli utenti del World Wide Web di condividere le informazioni su sistemi differenti. La famiglia di tecnologie XML è indipendente dal mezzo fisico o dalla piattaforma usata, ideale per la distribuzione di dati a una schiera infinita di agenti utenti, fra cui applicazioni, browser, dispositivi wireless abilitati al Web, e anche a strumenti che ancora non sono stati concepiti.

L'XML è un meta-linguaggio di markup, ovvero serve per definire altri linguaggi. Di per sé non è altro che un insieme standard di regole sintattiche per modellare la struttura dei documenti e dei dati, non ha tag predefiniti e non serve per definire pagine Web né per programmare. Si caratterizza per la semplicità con cui è possibile scrivere documenti, condividerli e trasmetterli nel Web. Le specifiche, ovvero l'insieme di regole, definiscono le modalità secondo cui è possibile crearsi un proprio linguaggio di markup, in questo modo gruppi di persone o organizzazioni possono creare il proprio linguaggio di markup specifico per il tipo di informazione che trattano; per molte applicazioni e per

diversi settori gli esperti hanno già creato linguaggi di markup specifici come ad esempio l'Health Level Seven, il Channel Definition Format, il Mathematical Markup Language ed altri.

Grazie all'XML viene riportata la normale separazione tra la struttura e la rappresentazione dei dati nella realizzazione di documenti per il Web che con il tempo, nella programmazione HTML, si erano confusi. La struttura dei dati separata dal contenuto della presentazione permette di visualizzare in vari modi lo stesso sorgente. Le informazioni, grazie all'XML, sono sempre disponibili essendo compatibili con i vari sistemi rendendo così disponibili le informazioni ad ogni apparato in grado di leggere XML, è possibile in questo modo accedere a documenti XML anche senza pagine HTML. Lo stesso sorgente scritto una volta sola può essere visualizzato attraverso audio da un cellulare come essere scritto su di un monitor. E' possibile, in questo modo, veicolato attraverso device diversi, non necessariamente presi in considerazione all'atto della sua stesura, un documento scritto secondo queste specifiche. L'XML quindi, pur essendo nato propriamente per il mondo Web, ha senso anche fuori da questo, comunque e dovunque qualcuno voglia produrre un documento, a prescindere dal mezzo trasmissivo.

L'XML è utilizzato in moltissime applicazioni Web proprio per il fatto che trasportando e stoccando dati ne semplifica la condivisione ed il mantenimento. Sta svolgendo un ruolo sempre più importante sul Web nello scambio di una grande varietà di dati e altrove ed è stato introdotto anche all'interno dello standard HL7 durante lo sviluppo della versione 3, che lo supporta pienamente mentre la versione 2 dello standard è stata adattata a questo formato.

La scelta di usare XML [9] per la strutturazione dei dati è dovuta da alcune sue importanti qualità:

- sintassi semplice: caratterizzata da poche e semplici regole;
- interoperabile: supportato da una grande varietà di framework e linguaggi di programmazione, garantisce un livello di interoperabilità indispensabile per l'implementazione di servizi web;
- estendibile: gli elementi già sintatticamente definiti possono essere estesi e adattati ad altri utilizzi;
- auto descrittivo: non richiede file esterni che ne definiscano la semantica perché quest'informazione è già contenuta al suo interno;
- permette agli sviluppatori di creare facilmente strutture ad-hoc per contenere informazione strutturata;

- parser: sono diffusi su tutte le piattaforme e sono free. Permettono di decodificare e validare le strutture XML, limitandosi a gestire solo l'informazione contenuta usando API specifiche (DOM, SAX);
- text-based: è facilmente editabile anche a mano ed è leggibile anche dagli esseri umani. Supporta UNICODE, quindi è adatto a ogni tipo di scrittura.

L'XML però presenta anche alcuni svantaggi, tendono ad essere molto più ingombranti di quelli in formato binario, quindi la loro trasmissione sulla rete non è ottimale, a causa delle struttura testuale e dei tag.

3.4 MySQL

MySQL [10] è un Database management system (DBMS) composto da un client con interfaccia a caratteri e da un server, entrambi i componenti sono disponibili sia per sistemi operativi UNIX che per il sistema operativo Windows. MySQL svolge il lavoro di DBMS sulle piattaforme LAMP e WAMP, una della più usate e installate in internet per lo sviluppo di siti e di applicazioni web dinamiche.

Dalla fine degli anni 90 MySQL supporta la maggior parte della sintassi SQL (Structured Query Language) e si prevede che in futuro supporti completamente lo standard ANSI. Possiede delle interfacce, dette driver, per diversi linguaggi di programmazione quali ad esempio Java e per gli ambienti di sviluppo quali Mono, che opera sotto UNIX, e .NET, che opera sotto Windows. Il codice di MySQL viene sviluppato fin dal 1979 dalla ditta TcX ataconsult, che ora si chiama MySQL AB, e nasce dall'esigenza di avere un database relazionale che potesse gestire grandi quantità di dati, ma che fosse allo stesso tempo molto veloce e sicuro. Solo dal 1996 viene distribuita una versione che supporta il linguaggio SQL, sfruttando del codice proveniente da un'altra applicazione: mSQL.

Le caratteristiche principali di MySQL [11] sono:

- la velocità: il sistema MySQL nasce, per volontà degli sviluppatori, per essere quanto più leggero e veloce possibile. L'obiettivo è stato certamente raggiunto, ma il prezzo da pagare è la limitazione a un ridotto insieme di funzioni rispetto ai prodotti concorrenti, anche se le funzioni che mancano sono quelle che solo pochi utilizzatori sono interessati ad usare.

  Le nuove funzioni sono sempre introdotte a ritmo di una o due per versione del programma, e sono sviluppate, ottimizzate e provate per mesi prima dell'effettiva introduzione, in modo da poter garantire la massima stabilità e la massima velocità di funzionamento.
- I costi: altro obiettivo di MySQL è quello di vendere un prodotto competitivo dal punto di vista economico: i costi, infatti, sono molto contenuti rispetto a quelli della concorrenza.

  Sono previsti due tipi di licenze per coloro che utilizzano il prodotto: una sotto licenza GNU-GPL, quindi è codice open source e free, per coloro

che non copiano, modificano o distribuiscono l'applicazione, ovvero se non ridistribuiscono MySQL in alcun modo si è liberi di usare il prodotto per far funzionare l'applicazione; e una licenza commerciale per coloro che utilizzano MySQL per creare applicazioni commerciali, in questo caso è previsto il pagamento della licenza, questa è proposta ad un prezzo più vantaggioso rispetto ai prodotti concorrenti.

Una buona parte del codice che compone il client è licenziato sotto GNU-GPL e quindi può essere usato per applicazioni commerciali.

- La portabilità: MySQL è completamente scritto e sviluppato tramite linguaggio C e C++ e verificato su vari compilatori in molteplici piattaforme. Questo sistema risulta attualmente essere l'unico DBMS disponibile per una così vasta scelta di piattaforme: allo stato attuale esistono versioni per Windows, MacOS, Solaris, FreeBSD, Novell NetWare, Linux ed altri sistemi ancora.

  Lo sviluppo in C/C++ e la libera distribuzione del codice sorgente ne rendono possibile la conversione per qualsiasi sistema esistente.
- Open Source: un'ultima caratteristica, che rende unico il sistema MySQL, è il sistema di sviluppo Open Source. Il codice che compone il programma è liberamente distribuito, ogni utente, purché esperto, può leggere e modificare il programma a proprio piacimento. Le modifiche possono essere inviate ai programmatori di MySQL AB, i quali le valutano e possono decidere di inserirle o meno in una successiva edizione del programma.

  Grazie a questo lo sviluppo dell'applicazione procede in modo rapido, consentendo una costante evoluzione e una rapida correzione degli eventuali errori di programmazione introdotti lungo il percorso.

  Questo sistema consente anche di mantenere un team ufficiale di programmatori interni molto ristretto, poco oneroso per la società, e in grado di prendere decisioni in modo rapido ed indipendente.

4

Architettura del sistema e implementazione della libreria

Nel corso degli anni nel settore sanitario si è assistito a una massiccia informatizzazione dei processi [12]. La loro implementazione è avvenuta senza alcuna logica di integrazione con i sistemi preesistenti, per cui le informazioni digitali non sono divenute, e non lo sono tuttora, patrimonio comune dell'intera organizzazione sanitaria, ma sono rimaste confinate nei sistemi informatici che le hanno generate.

La finalità di tali sistemi purtroppo si è fermata a soddisfare le specifiche esigenze operative e gestionali di alcune realtà (servizi organizzativi, pronto soccorso, servizi di amministrativi, servizi di analisi).

Oggi l'automazione dei singoli processi isolati non è più sufficiente per una corretta gestione del sistema ospedaliero. E' necessario realizzare strutture di gestione completamente compatibili che garantiscano interoperabilità e integrazione tra gli applicativi esistenti. Diviene fondamentale la condivisione di informazioni tra strutture che forniscono prestazioni di diagnosi e cura, al fine di garantire al cittadino un servizio altamente qualificato.

Le azioni possibili per una struttura sanitaria sono fondamentalmente due:

- l'adozione di un nuovo sistema di gestione con le caratteristiche sopra descritte, questa soluzione comporta però degli svantaggi:

- un aggravio economico dovuto all'acquisto di un nuovo software;
- costi aggiuntivi dati dalle personalizzazioni del sistema, dalla formazione del personale e dal recupero dello storico;

- da non trascurare inoltre i disagi creati all'utenza derivati dal periodo di start-up del sistema.

- l'utilizzo di tecnologie che abilitino gli applicativi a comunicare tra loro adottando un protocollo standard di comunicazione.

Nel momento in cui occorre acquistare dispositivi medici o sistemi informatizzati, oppure sviluppare del software o progettare un nuovo sistema, la scelta di soluzioni standardizzate, soprattutto se applicate in modo sistematico può portare numerosi vantaggi, eliminando quasi tutti i problemi derivanti da sistemi proprietari. L'utilizzo di standard consentono:

- di mantenere i sistemi in uso e introdurre nel sistema informativo aziendale una tecnologia che abiliti le applicazioni esistenti all'interoperabilità;
- di ridurre i costi: il costo di acquisto è generalmente una frazione rispetto al costo di una nuova procedura integrata; vengono abbattuti i costi derivanti dalle personalizzazioni del sistema e quelli necessari per la formazione del personale;
- da considerare inoltre che non modificando la procedura originale viene ridotto l'indice di rischio di errore dato dall'utilizzo di un nuovo sistema.

Dato che sempre più frequentemente vengono introdotti dispositivi medici complessi, si cerca dunque di introdurre standard per ridurre la complessità e i costi e per semplificare la comunicazione fra dispositivi di produttori differenti. Una maggiore integrazione facilita la distribuzione dell'informazione e rende più economica la gestione complessiva del sistema, a vantaggio dell'attività clinica e della salute del paziente. Con soluzioni standard il trasferimento automatico dell'informazione diventa più semplice e riduce la necessità di interventi diretti da parte degli operatori, aumentando la loro disponibilità e diminuendo la probabilità di errori dovuti all'intervento umano.

La scelta di adottare questa seconda soluzione per rendere il sistema interoperabile è dovuta anche al fatto che nell'adottare uno standard come protocollo di comunicazione una nuova applicazione potrà essere aggiunta in qualsiasi momento senza la necessità di modificare il sistema di integrazione, chiunque può interfacciare il proprio sistema indipendentemente dall'applicazione o dalla piattaforma usata.

Questo scenario è permesso utilizzando una nuova generazione di linguaggi e tecnologie software che permettono di condividere informazioni sulla rete con l'utilizzo di standard condivisi, quali ad esempio HL7 e DICOM.

L'utilità di utilizzare un formato standard come l'HL7 diventa evidente quando i sistemi che devono scambiarsi informazioni sono diversi, in questo caso si fa riferimento ad un unico documento che definisce regole comuni per tutti.

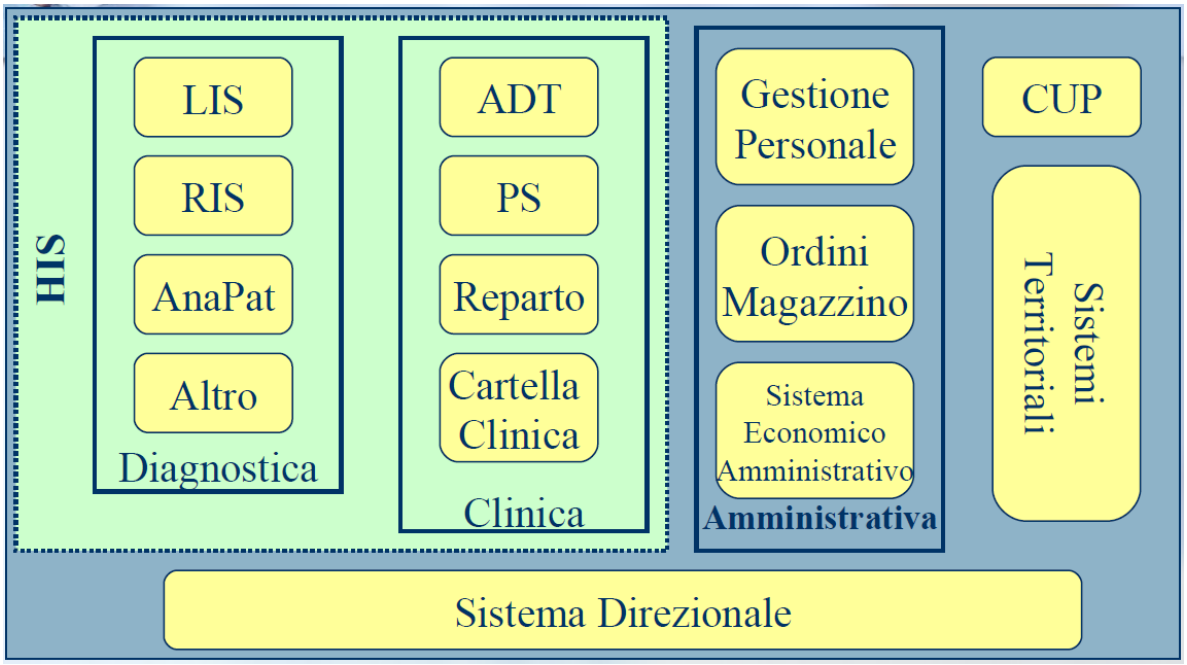

Figura 4.1. Esempio di un sistema informativo sanitario

4.1 Una libreria per le analisi HL7

Il progetto realizzato è una potente e completa piattaforma per lo scambio di messaggi con il più utilizzato protocollo di messaggistica in ambiente sanitario, l'HL7.

Attualmente le differenze che esistono nei vari processi amministrativi tra strutture sanitarie diverse non consentono all'HL7 di sviluppare un modello universale di elaborazione. Queste differenze, non previste dallo standard, richiedono negoziazione ed accordo tra le applicazioni. Per questa ragione l'HL7 non è un vero e proprio sistema completo e non è un vera interfaccia "plug-and-play".

Lo standard fornisce una struttura comune per la realizzazione di interfacce tra sistemi diversi anche se ogni fornitore implementa lo stesso messaggio HL7 in modi differenti, per gestire tutte le personalizzazioni dei messaggi si è reso necessario realizzare un modulo che sia altamente configurabile e personalizzabile per la gestione di tali messaggi.

La libreria è progettato in modo da uniformare lo scambio di dati, senza modificare le applicazioni sottostanti ma integrando le applicazioni già sviluppate aggiungendo la possibilità di scambiarsi informazioni in HL7, rendendo così possibile l'interoperabilità tra applicazioni eterogenee. In questo modo i sistemi coinvolti saranno in grado di dialogare tra loro e di gestire l'informazione in modo appropriato.

Si è scelto di strutturare il modulo come una libreria per fornire una vasta collezione di funzioni di base evitando di dover scrivere ogni volta le stesse funzioni di uso comune, risparmiando così tempo e costi per lo sviluppo. Gli altri vantaggi che si hanno dall'usare la libreria sono:

- si può modificare la libreria separatamente dal programma, senza limiti alla potenziale vastità di funzioni e strutture dati disponibili;
- possono essere aggiornate senza ricompilare tutte le applicazioni se non vengono modificate l'interfaccia delle funzioni e delle classi;
- le entità definite in una certa libreria possono essere riutilizzate da più applicazioni.

La libreria è stata realizzata in modo da gestire tutte le variazioni e le personalizzazioni dei messaggi grazie a un sistema di transcodifica delle

informazioni e dei tipi di dati presenti nei vari messaggi HL7 e in modo da supportate contemporaneamente le diverse versioni dell'HL7 esistenti sul mercato, dalla più recente versione 3 alla versione 2. E' possibile implementare velocemente e nel modo più facile i messaggi partendo dal dizionario inserito nel database Sys_log80. Questo dizionario definisce il formato e il contenuto dei messaggi che vengono personalizzati per ogni applicazione in base alle esigenze richieste dal cliente. L'applicazione è in questo modo altamente configurabile e permette di gestire anche le personalizzazioni non previste nello standard.

All'interno del modulo è disponibile un dizionario per tutti i tipi di messaggi, un motore di integrazione per la gestione dei messaggi, una serie di adattatori ciascuno per un differente meccanismo di comunicazione e la gestione di un tracking accurato di tutti i messaggi inviati e ricevuti. La libreria opera attraverso due algoritmi distinti, uno per la creazione e il parsing dei messaggi HL7 nel formato pipe e l'altro che si occupa di creare e convertire i messaggi HL7 nel formato XML. L'utilità di definire un formato standard come l'HL7 all'interno delle applicazioni diventa evidente quando i sistemi che devono scambiarsi informazioni sono diversi.

4.2 Il dominio applicativo

Il dominio applicativo è composto da una moltitudine di sistemi informativi eterogenei implementati con metodi di comunicazioni e tecnologie spesso incompatibili tra loro a causa di sviluppi avvenuti in tempi diversi per soddisfare necessità differenti ma legate sempre al tema della sanità. Prendendo come esempio una singola grossa struttura ospedaliera come l'ospedale di Rimini possiamo notare come l'informatizzazione avvenga in modo non omogeneo e a macchia di leopardo, andando a creare sistemi che si trovano a non essere direttamente interoperabili, seppur trovandosi all'interno di una stessa azienda. La conversione delle informazioni da un formato all'altro risulta quindi essere un tema di rilevante importanza.

L'obiettivo di questo lavoro di tesi è dunque quello di progettare ed implementare una soluzione di interoperabilità semantica all'interno dello standard HL7, che permetta la creazione e il parsing di messaggi attraverso la transcodifica delle informazioni presenti nel database.

4.3 L’architettura

L'architettura del sistema è di tipo Client/Server ed è costituita da una coppia di processi che si scambiano messaggi su una rete. Il sistema è formato da due tipi di moduli: il client e il server, che generalmente sono eseguiti su macchine diverse collegate in rete, il client può inviare dati al server attraverso vari protocolli di comunicazione:

- attraverso richieste HTTP, in questo caso il client invierà le richieste tramite i browser che frutta il protocollo HTTP per richiedere le pagine web tramite cui inserire i dati. Quando l’applicazione vuole inviare i dati al Web server per essere processati deve prima richiamare le funzioni della libreria, richiamabili all’interno delle pagine Web, per realizzare il messaggio nel formato HL7;
- tramite un servizio web reso disponibile dal web server, in questo caso i dati verranno inviati in formato HL7 al server che si occuperà di eseguire il parsing e l’acknowledge;
- lo scambio di dati può avvenire attraverso canale di comunicazione socket con protocollo TCP/IP su porte appositamente concordate. La tecnologia utilizzata fino a qualche anno fa prevedeva lo scambio di messaggi via socket, ma con l’arrivo della versione 3, che utilizza messaggi basati completamente su tecnologie XML si è reso necessario anche l’uso dei Web service. In entrambi i casi è possibile specificare il protocollo dati da utilizzare.

Il server Web in ascolto è del tipo Apache e su di esso è installata la libreria realizzata che consente di gestire la ricezione e l’interpretazione dei messaggi. Il server riceve la richiesta e risponde sull'indirizzo del terminale client da cui ha ricevuto i dati. Sul server sarà attivo un servizio HL7 costituito da un listener in grado di interpretare i messaggi in arrivo e di fornire il necessario acknowledge. Ad ogni connessione in ingresso estrae i dati ricevuti e li processa attraverso le apposite funzioni realizzate, una volta terminata questa fase, i dati possono essere trasferiti a una destinazione.

La parte server è costituita dal database MySQL all'interno del quale vengono memorizzate i dati da scambiare e tutte le informazioni necessarie alla

transcodifica delle informazioni. Il servizio utilizzerà le tabella di configurazione presenti su questo database per la traduzione dei messaggi in HL7.

Il modello client/server è particolarmente consigliato per delle reti che necessitano di un elevato livello di fiducia, i principali vantaggi che offre sono:

- risorse centralizzate: il server può gestire delle risorse comuni a tutti gli utenti, dato che si trova al centro della rete, in questo modo si evitano i problemi di ripetizione e contraddizione;
- maggiore sicurezza: dato che si riduce il numero di punti d'ingresso che permettono l'accesso ai dati;
- amministrazione a livello server: considerata la poca importanza dei client in questo modello, hanno meno bisogno di essere amministrati;
- una rete evolutiva: grazie a questa architettura è possibile cancellare o aggiungere i client senza disturbare il funzionamento della rete e senza modifiche importanti.

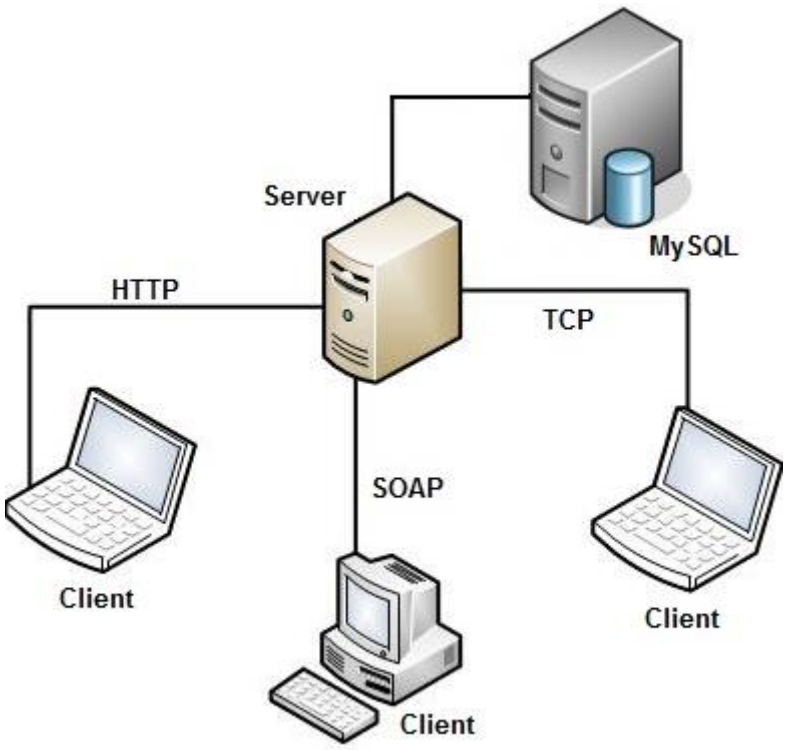

Figura 4.2. Architettura del sistema

4.4 Il database di informazione

Un sistema di transcodifica delle informazioni rappresenta un possibile strumento per superare le problematiche di eterogeneità delle informazioni tra i diversi sistemi informatici. L'utilizzo di un database per la transcodifica oltre a rendere il sistema interoperabile lo rende anche indipendente dai dati di configurazione che si differenziano a seconda del programma sul quale viene installata la libreria. In questo modo non è necessario apportare modifiche a livello di codice per poter configurare la libreria ed è possibile personalizzare i dati in essa contenuti in base alle necessità del programma, potendo anche differenziare la struttura dei messaggi HL7 secondo le richieste del client.

All'interno del database è presente, codificata, la struttura che dovrà avere il documento HL7 differenziata in base al tipo di messaggio che si intende creare, sono presenti anche le informazioni per reperire i dati da inserire all'interno dei singoli nodi, o segmenti nel caso si stia utilizzando il formato pipe, che compongono il documento HL7.

Le azioni che si possono effettuare grazie alla libreria sono anch'esse codificate all'interno del database. In base al tipo di azione che si intende compiere è possibile risalire alle informazioni per effettuare l'accesso al database interessato. La tabella dictionary viene utilizzata per poter uniformare la definizione dei campi che compongono il messaggio XML, in modo da mantenere una struttura affine al modello di messaggio che si intende realizzare.

All'interno di apposite tabelle viene anche mantenuto uno storico dei messaggi scambiati in modo da tenere traccia delle operazioni effettuate e delle login coinvolte.

Entità per il formato pipe

Le entità coinvolte nel processo di codifica dei messaggi HL7 in formato pipe

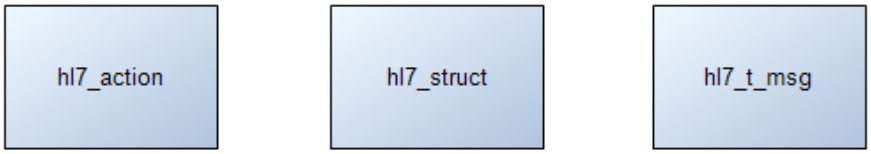

Figura 4.3 Diagramma E/R per i messaggi pipe

- *Hl7_t_msg:* contiene la struttura dei messaggi che inviamo.

Campo	**Descrizione**
ID	chiave primaria
type	indica il tipo di segmento, ovvero l'identificativo del segmento, a cui si riferisce la struttura
seq	indice del campo che compone il segmento
sub_seq	indice del componente che compone il campo
value	valore da inserire nel messaggio
eval	indica se il valore inserito in default è da interpretare come php o da copiare come testo

- Hl7_struct: contiene le informazioni necessarie alla decodifica dei messaggi in entrata.

Campo	**Descrizione**
ID	chiave primaria
type	indica l'identificativo del segmento
repeat	indica se il campo si ripete all'interno del segmento
seq	posizione del campo
sub_seq	posizione del componente
taget_field	transcodifica del componente
check_seq	indica in quale campo dobbiamo andare a verificare il valore per sapere quale valore indica il campo che stiamo analizzando
check_value	valore che deve assumere il campo in caso di controllo

- *Hl7_action:* indica in quale DataBase debba essere eseguita la query.

Campo	**Descrizione**
ID	chiave primaria
seq	identificativo del segmento da cercare nell'array contenete la transcodifica dei dati

field	transcodifica dei dati da cercare nell'array
target_value	valore che assume l'array per il segmento e la transcodifica indicata precedentemente
action	indica il tipo di query da eseguire; insert, update o replace
target_table	indica la tabella su cui deve essere eseguita l'azione

Entità per il formato XML

Andiamo ora ad elencare nello specifico le entità coinvolte nel processo di mappatura dei messaggi HL7 in tag XML.

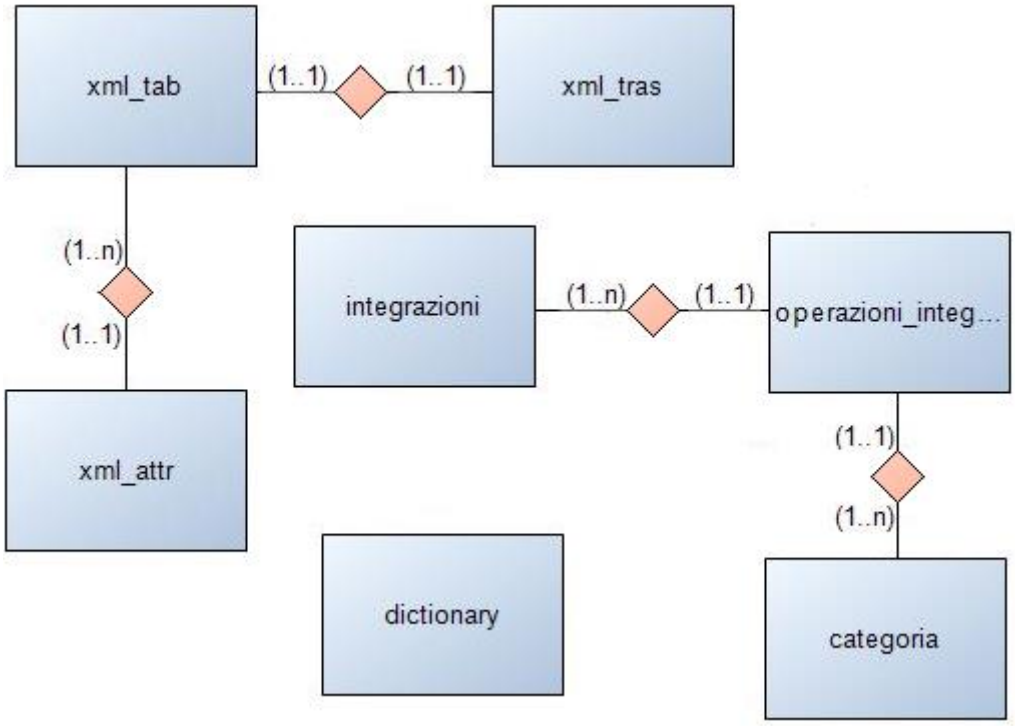

Figura 4.4 Diagramma E/R per i messaggi XML

- *Xml_tab:* la struttura dei messaggi XML varia a seconda del tipo di messaggio che si intende realizzare, qui vengono inseriti tutti i nodi e i sottonodi che andranno a costruire il documento XML.

Campo	Descrizione
ID	chiave primaria
request	permette di filtrare a seconda dei casi la struttura XML
tipo	indica il tipo di messaggio a cui appartiene il nodo
figli	riferimento all'id del nodo padre
nome_tag	nome del marcatore del nodo
default	valore che deve assumere il nodo
eval	indica se il valore inserito in default va interpretato come php o copiato come testo
ord	ordine in cui devono essere scritti i nodi

- *Xml_tras:* indica dove reperire i valori da inserire nel messaggio, nello specifico lega gli id dei nodi creati con gli indici dell'array contenete i valori di input.

Campo	Descrizione
ID	indice del nodo padre
campo	indice dell'array contente i valori di input

- *Xml_attr:* contiene eventuali attributi dei nodi.

Campo	Descrizione
ID	indice del nodo a cui appartiene l'attributo
nome	name dell'attributo
valore	indice dell'array contenente i valore dell'attributo
eval	indica se il valore inserito in default va interpretato come php o copiato come testo
ord	ordine in cui devono essere scritti gli attributi

- *Dictionary:* permette di unificare la struttura dei messaggi convertendo l'identificativo dei dati estratti nel sinonimo corrispondente.

Campo	**Descrizione**
synonym	sinonimo da usare, chiave primaria
words	parole da convertire

- *Integrazioni***:** indica in quali DataBase sono presenti i dati da estrarre.

Campo	**Descrizione**
ID	chiave primaria
server	nome del server su cui deve essere eseguita l'azione
DB	nome del database su cui deve essere eseguita l'azione
login	login da usare per eseguire la connessione
pwd	password per la connessione
tipoServer	tipo di server con cui ci stiamo collegando, m=mysql o=oracle
tipo_integrazione	tipo di integrazione, D=DataBase, W=webservice
crypt	indica se i dati inseriti nel messaggio debbano essere criptati
contenuti	chiave esterna su categoria, indica per quali categorie deve essere eseguita l'azione

- *Categorie:* viene utilizzata per identificare l'evento trigger che ha scatenato l'invio del messaggio

Campo	**Descrizione**
ID	chiave primaria
tipo	identificativo del nodo
etic	valore che deve assumere il nodo

- *Operazioni_integrazioni:* contiene l'azione da eseguire, ovvero la query, in base all'evento trigger e all'integrazione che si vuole eseguire.

Campo	**Descrizione**
ID	chiave primaria
id_integrazioni	chiave esterna su integrazioni, usata per identificare l'integrazione a cui si riferisce l'azione
id_categoria	chiave esterna su categoria, usata per specificare l'evento trigger per cui deve essere eseguita l'azione
azione	query da eseguire
ord	specifica l'ordine in cui devono essere eseguite le azioni

4.5 HL7 formato pipe

Il paradigma centrale del protocollo è lo scambio di messaggi tra applicazioni che vogliono dialogare fra di loro. Il funzionamento del protocollo è abbastanza semplice: ogni volta che due sistemi devono comunicare iniziano a scambiarsi messaggi bidirezionalmente. Prima di essere immessi in rete, i dati da allegare al messaggio verranno passati a un parser, il quale creerà un messaggio contenente tutte le informazioni necessarie. Solo a questo punto il messaggio verrà trasmesso. Il ricevente potrà comprendere la tipologia e decodificare il messaggio estraendo tutte le informazioni in esso contenute inserendole in un array indicizzato, in modo da semplificare la ricerca e l'elaborazione dei dati. Le informazioni vengono elaborate e viene inviato un messaggio di risposta sia nel caso il messaggio venga ricevuto correttamente, che in quello nel quale sono riscontrati errori, in questo caso il destinatario dovrà mandare un messaggio ack di risposta. In questo modo sistemi eterogenei potranno condividere e scambiarsi dati grazie all'uso del protocollo.

Andiamo ora ad analizzare nel dettaglio le varie fasi del protocollo.

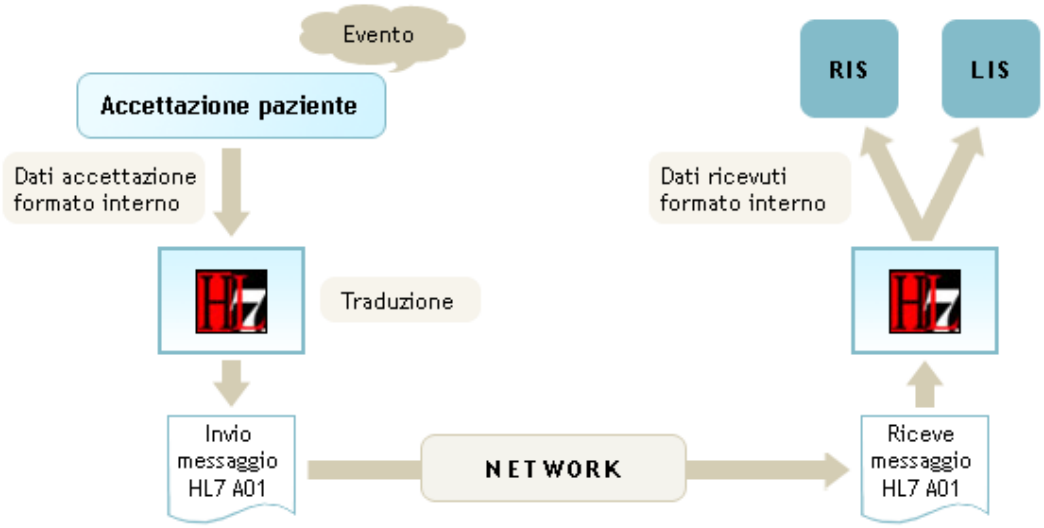

Figura 4.5. Esempio del funzionamento della libreria

4.5.1 Parsing dei messaggi ricevuti

Al ricevimento di un nuovo messaggio HL7 pipe viene chiamata la funzione HandleData() che si occupa di eseguire il parsing dei messaggi in ingresso. La

funzione richiama *parse_msg()* che si occupa di transcodificare le informazioni contenute nel messaggi ricevuto. Come prima cosa la funzione suddivide in segmenti il messaggio in base al separatore di segmento.

I segmenti così ottenuti vengono analizzati separatamente uno alla volta e viene utilizzato l'identificativo di segmento come chiave per creato l'indice di primo livello dell'array contenente i dati transcodificati, i dati contenuti nel corrispettivo segmento verranno analizzati e transcodificati inserendo le informazioni trovate all'interno dell'array.

Per poter indicizzare nel modo corretto l'array devono essere eseguiti dei controlli per gestire la possibilità che alcuni segmenti si ripetano all'interno del messaggio. Per eseguire questo tipo di controllo la funzione verifica se l'identificativo del segmento che sta analizzando è già presente all'interno dell'array, nel caso sia presente viene aggiunto un indice di secondo livello all'array in modo da identificare i diversi segmenti.

Una volta indicizzato nel modo corretto il primo livello dell'array viene richiamata, per ogni segmento individuato all'interno del messaggio, la funzione *parse_section()* che andrà ad analizzare i dati contenuti all'interno dei segmenti. Come prima cosa i segmenti vengono separa in campi in base al separatore di campo, "|". I campi vengono separati in componenti e viene effettuato un controllo nel database per verificare se il componente si ripete all'interno del messaggio e quindi se dovrà essere eseguita una separazione delle sottocomponenti.

Un campo può essere ripetuto e può assumere significati diversi, a seconda delle necessità, per verificare il significato che assume un campo può essere necessario andare a verificare il valore presente nel campo di check. La posizione e il valore di questo campo viene stabilita in accordo con il client e codificati all'interno della tabella hl7_struct. In base all'identificativo di segmento e all'identificativo del campo, ovvero la sua posizione, viene interrogata la tabella hl7_struct che contiene la transcodifica dei dati. Attraverso le informazioni contenute all'interno del database andiamo ad eseguire la transcodifica dei dati contenuti all'interno dei componenti che formano il segmento gestendo anche i controlli necessari nel caso della ripetizione dei componenti.

La transcodifica dei dati verrà inserita all'interno dell'array che viene strutturato nel seguente modo: l'indice di primo livello specifica il segmento che si è analizzato, l'indice di secondo livello indica il significato del campo

transcodificato e il valore presente all'interno dell'array corrisponde all'informazione passata nel messaggio. La libreria restituirà quindi un array contenete i dati transcodificati del messaggio.

I parser HL7 versione pipe non lavorano tanto diversamente dall'occhio umano: ogni riga del messaggio viene spezzata in un numero di sottostringhe in base ai caratteri di terminazione e leggendo un campo dopo l'altro viene ricostruito l'insieme delle informazioni mediche contenute. Purtroppo l'operazione ha dei costi elevati e purtroppo non è possibile renderlo più efficiente vista la natura fortemente sequenziale della lettura dei segmenti e dei campi. Da notare che tanti più sono i componenti quanto più degradano le prestazioni della lettura del messaggio.

```
MSH|^~\&|G2|INSIEL|MIRTH|OPBG|20131001070603||OML^O21|G213000000013739735|P|2.5
PID|||01706355^^^PK~1557210^^^PK_CUP||POGGI^MARTA^^^^^L|BORCHIA^ANNA|19910115|F|||VIA
BISACQUINO 13^^ROMA^RM^00132^IT^L^^058091~VIA BISACQUINO 13^^ROMA^RM^00132^IT^H^^058091
~^^ROMA^RM^00100^IT^N^^058091||^^^^^^^^^^3387003196|||||PGGMRT91A55H501T|PGGMRT91A55
H501T||2445472^^^PK|||||100^ITALIA||||||DEF-NCE
PD1|||^^^^^^^^120102
PV1||O|CUP|||||||||||||||||5934850|||||||||||||||||||||||||||||7854045|V
ORC|NW|14935187||4890320|||1^^^20131001070500^^R||20131001070532|3713||587485^PITZALIS^CA
RLO|||20131001||^^SIO|rm-casse-sp03
OBR|1|14935187||2572^PRELIEVO VENOSO - 999^SIO|||||||||||||1201108204589551|||||||||||||||||
||||||||||17825^AMB-INF PEDIATRICO~

a:8:{s:3:"MSH";a:5:{ s:6:"sender";s:6:"INSIEL";
                     s:14:"tipo_messaggio";s:3:"O21";
                     s:12:"id_messaggio";s:20:"G2130000000013739735";
                     s:7:"success";s:8:"No Error";
                     s:3:"ack";s:2:"AA";}
    s:3:"PID";a:20:{ s:6:"cod_az";s:8:"01706355";
                     s:4:"cogn";s:8:"POGGI";
                     s:4:"nome";s:4:"MARTA";
                     s:5:"d_nas";s:8:"20080115";
                     s:3:"sex";s:1:"F";
                     s:5:"i_res";s:17:"VIA BISACQUINO 13";
                     s:5:"i_dom";s:17:"VIA BISACQUINO 13";
                     s:10:"comune_nas";s:6:"058091";
                     s:10:"comune_dom";s:6:"058091";
                     s:10:"comune_res";s:6:"058091";
                     s:5:"n_tel";s:10:"3387003196";
                     s:5:"c_fis";s:16:"PGGMRT91A55H501T";
                     s:5:"t_san";s:16:"PGGMRT91A55H501T";
                     s:6:"cittad";s:3:"100";
                     s:5:"p_nas";s:3:"058";
                     s:5:"c_nas";s:3:"091";
                     s:5:"p_res";s:3:"058";
                     s:5:"c_res";s:3:"091";
                     s:5:"p_dom";s:3:"058";
                     s:5:"c_dom";s:3:"091";}
    s:3:"PD1";a:2:{ s:3:"usl";s:3:"102";
                    s:5:"r_usl";s:3:"120"; }
    s:3:"PV1";a:3:{ s:5:"t_acc";s:1:"O";
                    s:5:"c_rep";s:4:"CUP-";
                    s:7:"num_epi";s:7:"5934850"; }
    s:3:"ORC";a:3:{ s:12:"placer_order";s:8:"14935187";
                     s:8:"data_app";s:14:"20131001070500";
                     s:5:"d_pre";s:8:"20131001"; }
    s:3:"OBR";a:4:{ s:7:"c_prest";s:4:"2572";
                     s:7:"d_prest";s:21:"PRELIEVO VENOSO - 999";
                     s:5:"n_imp";s:16:"1201108204589551";
                     s:8:"rep_erog";
                     s:5:"17825"; }}
```

Figura 4.6. Es di messaggio transcodificato

A questo punto i dati contenuti all'interno del messaggio sono stati transcodificati e inseriti all'interno dell'array multidimensionale, in questo modo la ricerca e l'elaborazione dei dati si semplifica notevolmente.

Le informazioni contenute nell'array vengono analizzate per poter estrarre le informazioni necessarie a individuare l'azione da campiere sul database richiesto nel messaggio. La funzione *action()* si occupa di svolgere questo compito, nello specifico andrà ad identificare il tipo di evento che ha scatenato l'invio del messaggio, cioè estrarrà il valore del message type presente nell'array con la transcodifica dei dati, in base al quale interrogherà la tabella hl7_action da cui estrarrà il tipo di query da eseguire e il DataBase su cui dovrà lavorare. Richiamando la funzione *build_query()* l'applicazione andrà a creare la query richiesta nel messaggio.

La funzione andrà a realizzare la query in base ai bati estratti dal database per l'azione richiesta nel messaggio. Nel caso in cui la query da realizzare sia di aggiornamento la funzione dovrà individuare la chiave primaria dei dati ricevuti in ingresso effettuando una ricerca nel DB.

Per creare la query la funzione estrae dal database i nomi delle colonne che compongono la tabella interessata e i corrispettivi dati dall'array contente la transcodifica dei dati. Questi dati vengono uniti secondo la sintassi del tipo di query da realizzare; una volta creata la query viene eseguita e il risultato elaborato in modo da creare un messaggio HL7 di risposta.

4.5.2 Creazione dei messaggi

La creazione di un messaggio HL7 nel formato pipe viene effettuata creando un segmento alla volta grazie alla funzione *create_section()* che prende in input il tipo di segmento che si vuole realizzare. La funzione estrae da hl7_t_msg i campi che compongono il segmento e richiama a sua volta *create_sequence()* che si occupa di creare i singoli campi. Questa funzione interroga a sua volta il database per individuare i componenti che compongono il campo e recupera i valori da inserire dentro i campi.

Richiamando la funzione *get_sequence_length()* viene calcolato quanti campi compongono il segmento. La funzione *fill_sequence()* crea la struttura del segmento andando a inserire i campi, i componenti e eventuali campi vuoti dato. E' importante mantenere la lunghezza prefissata dei segmenti e inserire i campi vuoti quando i valori non sono presenti dato che i segmenti devo avere una lunghezza fissa e che la struttura dei messaggi HL7 è order-sensitive.

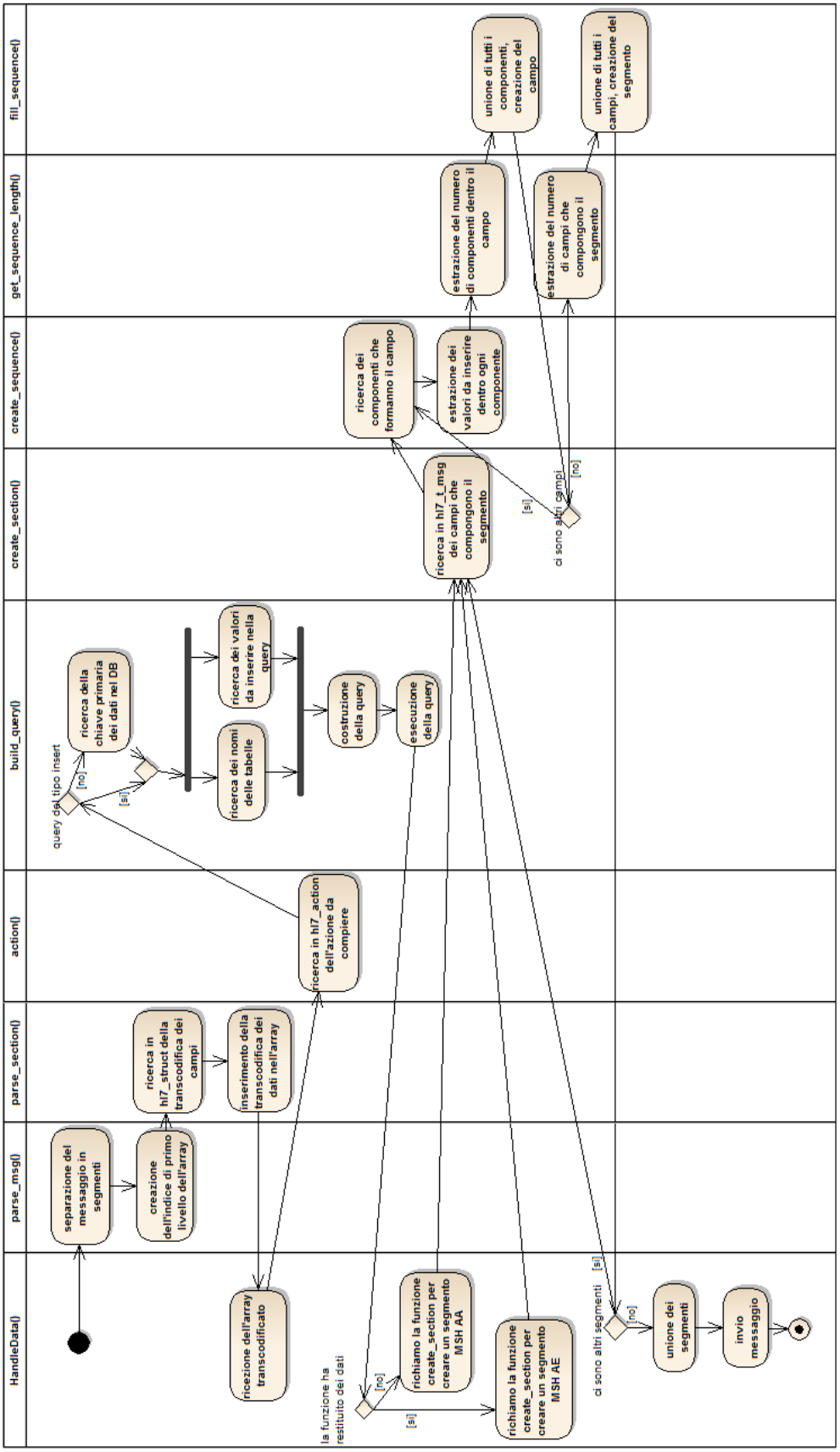

Figura 4.7. Activity diagram per i messaggi in formato pipe

4.6 HL7 formato XML

Le libreria è sviluppata in modo da poter supportare anche le versioni più recenti dell'HL7 che utilizzano l'XML come formato per la realizzazione dei messaggi.

Il protocollo più recente è migliore rispetto alla versione pipe che adotta del testo delimitato da caratteri speciali, nonostante ciò sia dovuto al fatto che i progettisti della versione pipe fossero obbligati a fare questa scelta, dato che non era ancora presente un linguaggio adatto a descrivere documenti strutturati.

XML regala ad HL7 una facilità di estensione assente nella precedente versione pipe: l'adozione di XML infatti consente un parsing e una validazione molto semplificata per via dell'uso di ottimi strumenti esterni quali Dom e XML Schema.

L'utilizzo di una struttura ad albero, e del linguaggio XML consente di avere una serie di innumerevoli vantaggi. La fase di parsing è molto avvantaggiata, la struttura ad albero consente accessi molto rapidi, visto che ogni volta che si vuole scendere dalla radice a una foglia, è richiesto tempo O(log n).

Un grande merito di HL7 XML è quello di aver adottato un modello delle informazione e un formato di interscambio dati che consentono al protocollo di essere pienamente estendibile: l'utilizzo di XML consente di poter integrare in maniera molto più rapida e pulita le varie interazioni introdotte da altri soggetti.

```
<![CDATA[<?xml version="1.0"?><ACK>
<MSH>
 <MSH.1>|</MSH.1><MSH.2>^~\&amp;</MSH.2>
 <MSH.3><HD.1>LOG80</HD.1><HD.2></HD.2><HD.3></HD.3></MSH.3>
 <MSH.4><HD.1>LOG80</HD.1><HD.2></HD.2><HD.3></HD.3></MSH.4>
 <MSH.5><HD.1>JCAPS</HD.1><HD.2></HD.2><HD.3></HD.3></MSH.5>
 <MSH.6><HD.1>NOEMALIFE</HD.1><HD.2></HD.2><HD.3></HD.3></MSH.6>
 <MSH.7><TS.1>20131026095147</TS.1></MSH.7>
 <MSH.8></MSH.8>
 <MSH.9><MSG.1>ACK</MSG.1><MSG.2>1</MSG.2><MSG.3>ACK</MSG.3></MSH.9>
 <MSH.10>1361868707</MSH.10>
 <MSH.11><PT.1>P</PT.1><PT.2></PT.2></MSH.11>
 <MSH.12><VID.1>2.5</VID.1><VID.2><CE.1></CE.1></VID.2><VID.3><CE.1></CE.1></VID.3></MSH.12>
 <MSH.13></MSH.13><MSH.14></MSH.14><MSH.15></MSH.15><MSH.16></MSH.16><MSH.17></MSH.17>
 <MSH.19><CE.1></CE.1><CE.2></CE.2><CE.3></CE.3><CE.4></CE.4><CE.5></CE.5><CE.6></CE.6></MSH.19>
 <MSH.20></MSH.20>
</MSH>
<MSA>
 <MSA.1>AA</MSA.1><MSA.2>2013102610743157</MSA.2>
</MSA>
</ACK>]]>
```

Figura 4.8. Esempio di messaggio ACK in formato XML

4.6.1 Ricezione messaggio

Quando si riceve un documento che contiene dati bisogna, innanzitutto, essere in grado di leggerli. L'azione che compie un programma quando legge un messaggio e ne interpreta il contenuto si chiama parsing, questa azione viene affidata alla funzione *HL7_Search()*.

Per manipolare i messaggi XML viene usato il DOM XML che definisce un modo standard per l'accesso e la manipolazione di documenti XML. La classe DOM (Document Object Model) XML è una rappresentazione in memoria di un documento XML che viene presentato come una struttura ad albero. Consente di leggere e modificare un documento XML a livello di codice.

Una volta ricevuto il documento, il parser si occupa di costruire un albero di oggetti che rappresenti il contenuto e l'organizzazione dei dati contenuti. In questo modo l'albero esisterà in memoria e l'applicazione potrà attraversarlo e modificarlo in ogni suo punto. Ovviamente il prezzo da pagare è il costo di computazione iniziale per la costruzione dell'albero ed il costo in memoria.

Il software analizza il messaggio richiamando *HL7_search().* Questa funzione si occupa dell'analisi del contenuto del messaggio filtrando i dati trovati in base al tipo di informazione che si sta cercando e che le è stata passata in input. La funzione viene richiamata per individuare l'evento trigger che ha scatenato l'invio del messaggio, una volta individuato viene utilizzato per eseguire una query di ricerca nel DB nella tabella integrazioni in modo da individuare le specifiche necessarie per connettersi al server su cui dovrà essere eseguita l'azione di aggiornamento o ricerca richiesta nel messaggio.

I dati estratti vengono poi passati alla funzione *execute()* che esegue la connessione con il server in base hai parametri precedentemente individuati e si occuperà di costruire ed eseguire la query.

La struttura della query viene estratta dalla tabella operazioni_integrazioni filtrando i dati in base all'evento trigger e al server precedentemente individuato. Alla query estratta vanno aggiunti i dati estratti dal messaggio ricevuto, questa operazione è affidata alla funzione *parse_action().*

La funzione *parse_action()* estrae i dati presenti nel messaggio HL7 e li inserisce correttamente all'interno della query estratta dal DB.

```
$query = "SELECT V_DSPAZPAZ.*, ";
$query = $query."to_char(PAZ_DNAS,'YYYY-mm-dd') d_nas, ";
$query = $query."to_char(PAZ_DCES,'YYYY-mm-dd') d_dec ";
$query = $query."FROM V_DSPAZPAZ ";
$query = $query."WHERE PAZ_CODI IS NOT NULL ";
$cogn_nome = $cogn.'%'.$nome;
$query = $query."AND PAZ_COGN LIKE ";
$query.=" '".str_replace("'","''",trim(strtoupper($cogn_nome))."%' ";
if(!empty($d_nas))
 $query = $query."AND to_char(PAZ_DNAS,'YYYY-mm-dd') = '".dataUSA($d_nas)."' ";
if(!empty($sex))
 $query = $query."AND PAZ_SESS = '$sex' ";
if(!empty($c_fis))
 $query = $query."AND PAZ_CFIS = '$c_fis' ";
if(!empty($t_san))
 $query = $query."AND PAZ_TSAN = '$t_san' ";
if(!empty($cod_az))
 $query = $query."AND PAZ_CODI = '$cod_az' ";
$query = $query."ORDER BY PAZ_COGN, PAZ_DNAS ";
```

Figura 4.9. Esempio di query estratta da operazioni_integrazioni

La query completa restituita in output dalla precedente funzione viene eseguita sul server e i risultati vengono elaborati per essere inviati al client.

I risultati vengono passati alla funzione *section()* che si occuperà di creare un messaggio di risposta. Ad ogni messaggio ricevuto, così come previsto dallo standard HL7, viene generato e di conseguenza inviato, un messaggio ACK di risposta.

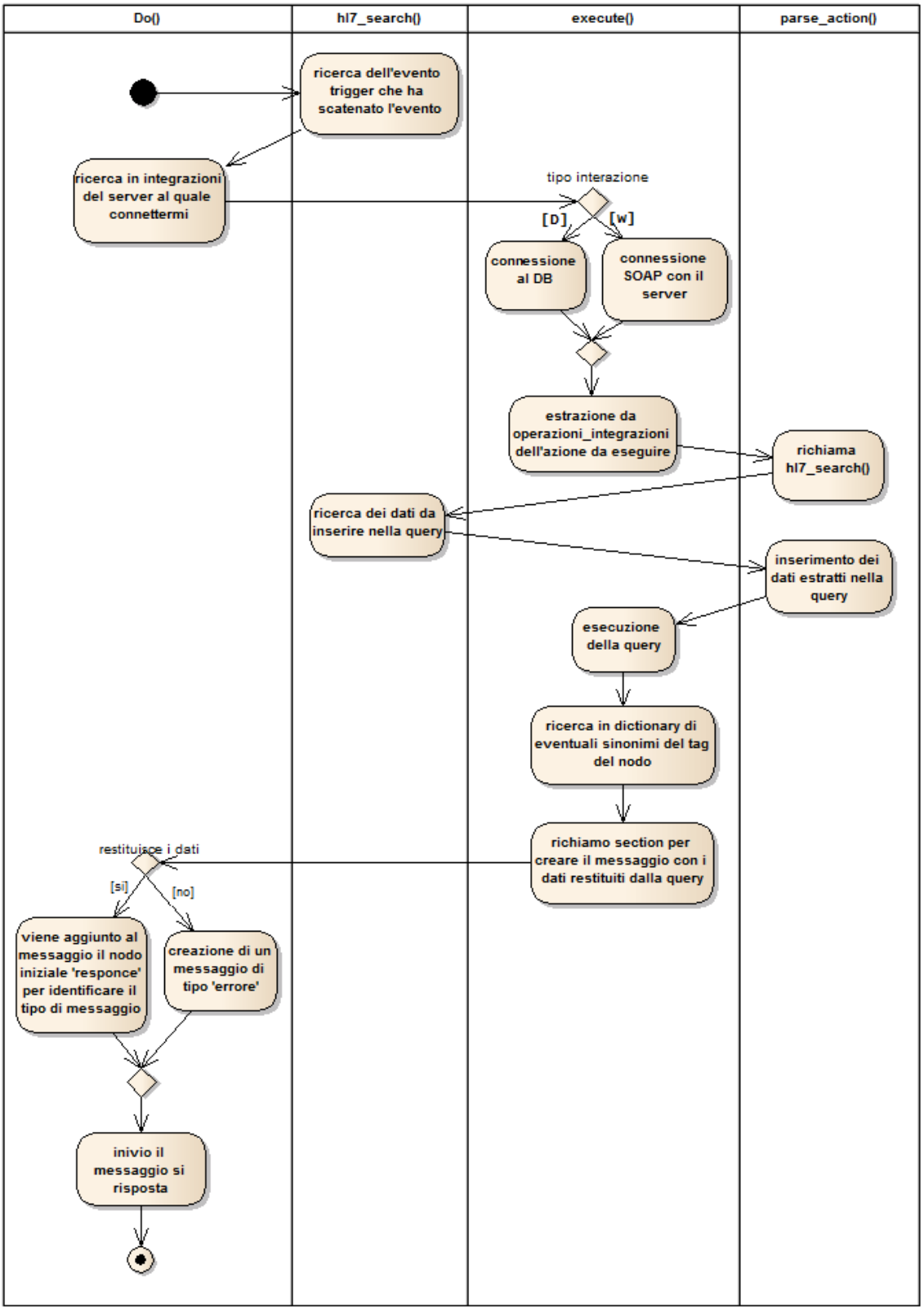

Figura 4.10. Activity diagram per il parsing dei messaggi

4.6.2 Creazione messaggio

La funzione *section()* si occupa di creare messaggi HL7 in formato XML. Essa estrae dalla tabella xml_tab la struttura del messaggio XML da realizzare in base al tipo di messaggio che le è stato passato in input.

I singoli nodi che compongono il messaggio sono realizzati dalla funzione *sectionNode()*. La funzione verifica nella tabella xml_tras quali valori vanno inseriti all'interno del messaggio e li recupera, l'elemento viene creato e appeso al documento DOM.

Viene poi richiamata la funzione *setAttribute()* per verificare se il nodo che sto realizzando ha degli attributi, la verifica viene effettuato controllando i valori all'interno di xml_attr. La funzione viene richiamata per verificare se il nodo ha eventualmente dei sottonodi, in caso positivo viene richiama ricorsivamente finché non ha appeso tutti i sottonodi.

Il nome del tag da assegnare al nodo è specificato all'interno della tabella dictionary, in questo modo vengono definiti nel modo corretto i nodi che compongono il messaggio evitando problemi derivanti da tag personalizzati che potrebbero portare a problemi di riconoscimento o anche di collisioni tra le diverse applicazioni.

Il messaggio viene inviato all'applicazione destinataria con uno dei canali standard di comunicazione.

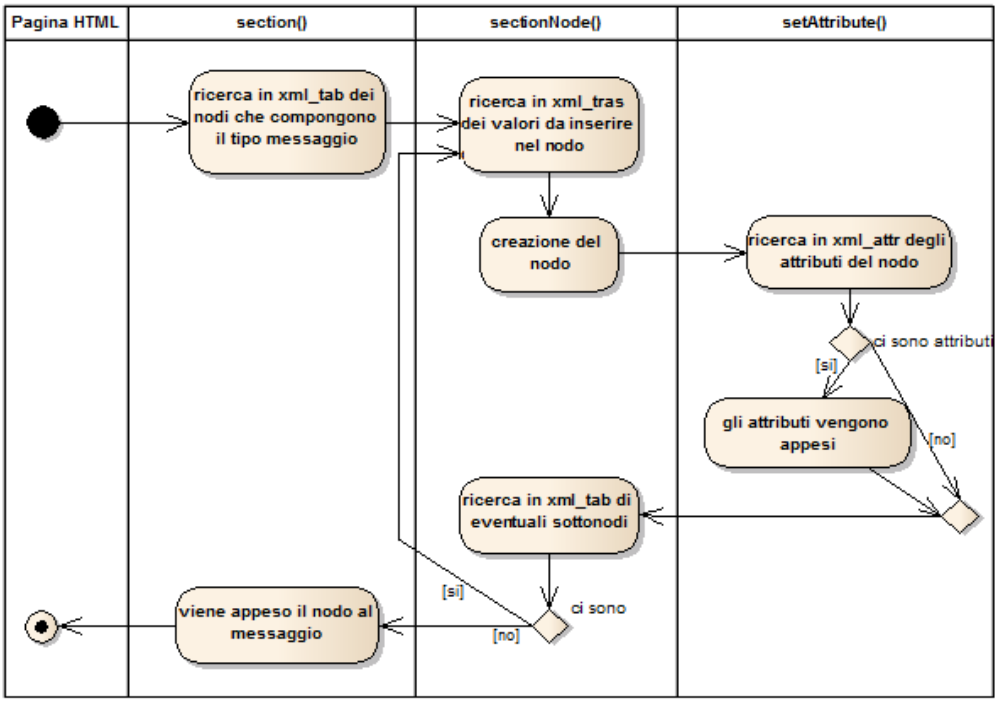

Figura 4.11. Activity diagram per la creazione dei messaggi

4.7 Testing

La fase di verifica si è estesa per tutta la durata del progetto, ed ha riguardato sia la fase di parsing che la fase di creazione dei messaggi.

Per verificare il corretto funzionamento della libreria sono stati presi come esempio dei casi d'uso spiegati nel capitolo 2.3.

Lo sviluppo della fase di parsing dei messaggi ha richiesto notevoli sforzi per interpretare nel modo corretto i messaggi ricevuti. La difficoltà principale è stata la gestione della ripetizione di segmenti con lo stesso identificativo di segmento all'interno dello stesso messaggio, ciò portava alla sovrascrizione dei segmenti all'interno dell'array contenete la transcodifica dei dati. Per gestire questo problema è stato necessario introdurre nell'array un secondo indice che permettessi di identificare i diversi segmenti ripetuti all'interno del messaggio.

Durante questa fase sono stati analizzati anche i miglioramenti ottenuti. Prima dello sviluppo della libreria era disponibile una funzione che gestiva i messaggi manualmente caso per caso. Grazie alla libreria sviluppata i messaggi ora vengono gestiti automaticamente tramite la transcodifica delle informazioni del database, mantenendo così la possibilità di implementare tutte le personalizzazioni necessarie. I tempi per l'aggiornamento sono diminuiti: per gestire un nuovo caso d'uso è sufficiente inserire i dati per la transcodifica nel DB e non è più necessario andare a scrivere manualmente le azioni da campiere per interpretare il nuovo tipo di messaggio, in questo modo viene ridotta anche la possibilità di errore. Si ha avuto un notevole miglioramento anche per i tempi di esecuzione, prima erano necessari mediamente 4 secondi per eseguire il parsing di un messaggio HL7, integrando la libreria sviluppata i tempi si sono ridotti a solamente 1 secondo, anche le righe di codice sono diminuite drasticamente passando da più di 16.000 a circa 500, riducendo così anche la possibilità di errore.

La libreria ha quindi portato numero vantaggi per la gestione dei messaggi HL7 soprattutto in termini di tempi e costi di sviluppo.

5
Conclusioni

L'analisi relativa ai metodi di interoperabilità esposta in questa tesi ha affrontato il tema della conversione delle informazioni secondo formati differenti ma operanti sullo stesso dominio, caso di particolare interesse nel dominio sanitario, caratterizzato da sistemi informativi eterogenei e sviluppati in tempi e secondo standard diversi l'uno dall'altro.

I risultati ottenuti hanno permesso la creazione di una procedura automatica per la creazione e l'interpretazione dei messaggi nelle versioni 2 e 3 dello standard HL7, procedura che consente di gestire le personalizzazioni non previste all'interno dello standard e l'allineamento della conoscenza in possesso del sistema alle ultime specifiche dello standard.

Questo strumento essendo una soluzione personalizzabile e basata su standard offre grandi vantaggi. Permette di interconnettere tutti i sistemi e gli applicativi di un sistema informativo sanitario senza la necessità di apportare modifica alle applicazioni già sviluppati riducendo così i tempi di esecuzione.

La libreria è quindi in grado di abilitare in modo automatico, usando le conoscenze esplicitata dallo standard, l'interoperabilità tra i diversi applicativi presenti all'interno di un sistema informativo sanitario.

L'integrazione completa di tutte le possibili caratteristiche in un contesto ospedaliero è un possibile sviluppo che si potrebbero aggiungere al sistema realizzato.

Un importante sviluppo futuro che si potrebbe realizzare nel sistema realizzato è legato alla sicurezza e alla collaborazione. Per la realizzazione di questo progetto si è focalizzata l'attenzione sulla progettazione e sull'implementazione di un metodo di interoperabilità, definendo comunque i requisiti relativi al

controllo degli accessi e alla collaborazione tra utenti nell'opera di creazione di messaggi. Potrebbe risultare di interesse lo studio delle problematiche connesse a sistemi collaborativi nel dominio sanitario con la definizione di requisiti per il controllo degli accessi alle informazioni definendo adeguate regole per autorizzare o negare l'uso di una risorsa.

La conversione tra standard distinti potrebbe essere un altro sviluppo futuro. L'allargamento del numero degli standard considerati per le operazioni di conversione comporterebbe uno studio attento degli standard e un'estensione degli algoritmi di parsing e creazione dei messaggi secondo diversi formati.

6
Riferimenti bibliografici

[1] **Un quadro di riferimento sulle tecnologie dell'informazione nel settore sanitario**
[http://staff.icar.cnr.it/cannataro/unicz/didattica/medicina/ISEI-MED-II/ModuloReti/Monografie/QuadroRiferimentoInformaticaSanitaria.pdf]

[2] **Health Level 7**
[http://www.hl7.org]

[3] **Health Level 7 v2 brief description**
[http://www.interfaceware.com/hl7.html]

[4] **HL7 pipe struttura dei messaggi**
[http://www.labcomm.co.uk/it/analyser-connectivity/hl7-message-format]

[5] **HL7 RIM**
[http://www.medinfo.dist.unige.it/didattica%5CSei2%5C03_HL7_V3.pdf]

[6] **Socket**
Andrea Gini – Paolo Aiello – Lorenzo Bettini – Giovanni Puliti, 2003, *Manuale pratico di Java: dalla teoria alla programmazione,* Tecniche Nuove, Milano

[7] **Web service**
Richard Monson-Haefel (2004), *J2EE Web Services,* Addison-Wesley, Boaton

[8] **XML**
Devan shepherd (2002), *XML guida completa*, APOGEO srl, Milano

[9] **XML**
[http://digilander.libero.it/shppotta/origine.htm]

[10] **MySQL**
[http://it.wikipedia.org/wiki/MySQL]

[11] MySQL
Luke Welling - Laura Thomason (2004), *MySQL Tutorial,* Pearson Education Italia srl, Milano

[12] Interoperabilità dell'HL7
[http://www.omnicom.bs.it/soluzioni-hl7-intro.html]

7

Indice delle figure