Università degli Studi di Modena e Reggio Emilia

Facoltà di Ingegneria – Sede di Modena

Corso di Laurea Specialistica in Ingegneria Informatica

Progetto e realizzazione di un sistema di controllo di una rete di sensori eterogenei di una multiutility

Relatore:
Chiar.mo Prof. Bergamaschi Sonia

Candidato:
Maschi Paolo

Correlatore:
Ing. Barbi Andrea

Anno Accademico 2008/2009

Key words:
reti di sensori
RDBMS
ChannelManager
Driver (Wrapper)

RINGRAZIAMENTI

Ringrazio la prof.ssa Bergamaschi, che con la Sua pazienza, dedizione ed i Suoi preziosi consigli mi ha permesso di realizzare nel migliore dei modi questo elaborato. Ringrazio il mio tutor aziendale e tutto il team della software house X DataNet, in cui ho svolto la mia attività di ricerca, per la straordinaria disponibilità e cortesia dimostratemi. Ringrazio la mia fidanzata Giulia, la mia famiglia (in particolare mio padre in qualità di revisore lessicografico), i miei amici e tutte le persone care che mi hanno accompagnato in questo emozionante “viaggio”. Ringrazio una Persona Speciale, che mi ha saputo ascoltare e soccorrere laddove non poteva l’aiuto umano. Ho avvertito la sua costante presenza in diversi momenti, a testimonianza che la fede è il solo strumento per vincere quelle paure ed angosce che solo Lui sa come far svanire.

INDICE

INTRODUZIONE

Negli ultimi anni un interesse sempre maggiore nel campo del telecontrollo ha portato ad una rapida diffusione dell'utilizzo di device muniti di sensori, migliorando sensibilmente l'attività di rilevazione di misure e di segnalazione di allarmi, con la possibilità di configurare le modalità operative e di aggiornare il firmware di tali device da remoto.

Nel tirocinio svolto presso la software house X DataNet, è stato affrontato l'importante tema del telecontrollo, nell'ambito del progetto software denominato eCentral commissionato dalla multiutility CPL di Concordia. eCentral prevede il controllo a distanza di un insieme eterogeneo di device muniti di sensori. Il software è stato richiesto per:

1. ridurre l'onerosità delle attività di monitoraggio e di intervento ordinario;
2. pianificare gli interventi programmati;
3. individuare tempestivamente le necessità di interventi straordinari ed urgenti;
4. analizzare i dati storici per stabilire azioni correttive e preventive;
5. consentire la gestione integrata di una quantità crescente di device di domini applicativi diversi, richiesta dai crescenti servizi offerti da CPL Concordia.

Lo scopo del tirocinio è stato quello di implementare un insieme di elementi appartenenti all'architettura del sistema software eCentral; tale insieme esegue funzioni:

– di basso livello per realizzare l'interfaccia di comunicazione bidirezionale con device remoti,
– di alto livello direttamente utilizzabili dall'utente per svolgere l'attività di monitoraggio dei device del modello Edor su canale GSM all'interno del sottodominio applicativo di odorizzazione.

Durante lo svolgimento della tesi l'attività di analisi, già iniziata in precedenza, è continuata e, pertanto, le successive fasi del ciclo di sviluppo del sistema software eCentral sono state affrontate con un diverso livello di dettaglio per rispettare le scadenze temporali del tirocinio, approfondendo solo alcuni aspetti.

Si descrivono sinteticamente nel seguito le fasi in cui è stato suddiviso il lavoro del tirocinio:

1. **Analisi:** durante tale fase si è acquisita la terminologia specifica del dominio applicativo di odorizzazione, indispensabile per poter, successivamente, comprendere le dinamiche per lo scambio delle informazioni, individuando e definendo in modo non ambiguo i requisiti del sistema software. Come risultato di tale fase sono stati prodotti output documentali significativi per il progetto.
2. **Progettazione:** durante tale fase si è stabilito come utilizzare le tecnologie HW/SW disponibili; essa è stata suddivisa per comodità in:
   – **progetto di rilascio dell'applicazione eCentral:** si è progettata un'architettura di rilascio in grado di supportare il modello di interazione client-server del sistema software eCentral, che include 2 strati di nodi elaborativi:
     1. il primo strato contiene il server Web Microsoft IIS 7.0, che gestisce le richieste interattive, provenienti da un web–browser, di pagine il cui contenuto è generato dinamicamente dall'esecuzione di script da parte del motore d'esecuzione ASP.NET 3.5, integrato in IIS e responsabile della logica di business;

2. il secondo strato contiene il server database Microsoft SQL Server 2005, che gestisce la memorizzazione dei dati.

Si è deciso di non utilizzare un server applicativo distinto, poiché i componenti applicativi non devono essere riusati da altre applicazioni Web per invocare la medesima logica di business. Per l'accesso ai dati è stato scelto il middleware ADO.NET.

- **Configurazione del first tier:** si è proceduto alla configurazione dei processi di lavoro di IIS e di ASP.NET.
- **Comprensione della struttura del progetto dei sorgenti del software applicativo:** sulla base dell'architettura *enterprise* proposta dal team di sviluppo sono stati individuati i package software disponibili, in cui organizzare le strutture dati per la successiva fase di implementazione.
- **Partecipazione attiva alla creazione del servizio di gestione della comunicazione con i device:** con le nozioni, apprese durante l'analisi, è stato possibile partecipare attivamente alla fase di definizione di un'architettura stratificata per consentire l'interazione di basso livello tra il sistema software eCentral ed i device in remoto.
- **Creazione di una base di dati:** in funzione delle operazioni da simulare, del modello di dati richiesto dall'architettura stratificata e dalla necessità di garantire la massima flessibilità nella gestione della basi di dati, ossia facilità e tempestività di modifica con basse ripercussioni sul codice, si è proceduto alla definizione di un modello dati efficiente da integrare in quello realizzato dal team aziendale. Durante tale fase si è affrontato anche il problema di una gestione temporale delle informazioni da memorizzare.

3. **Implementazione:**
   - dopo aver definito il modello dei dati e l'architettura stratificata di comunicazione con i device, sono stati implementati i componenti software:
     1. *GSMChannelManager*, deputato all'acquisizione e alla spedizione di dati rispettivamente da e verso i device sul canale di comunicazione GSM,
     2. *EdorDriver*, che garantisce l'interoperabilità tra i device e l' "eCentralCore" fornendo l'astrazione di una lingua comune tra le parti;
   - è stata, infine, verificata la capacità del sistema di fornire i risultati attesi da un utente, che svolge l'attività di monitoraggio, sulla base dei dati acquisiti attraverso l'azione congiunta dei componenti software creati.

Il presente elaborato è composto di quattro capitoli:

- il primo capitolo descrive l'attività di analisi finalizzata alla definizione dei requisiti e delle specifiche di progetto e alla formalizzazione di alcuni casi d'uso;
- il secondo capitolo descrive il progetto architetturale HW/SW;
- il terzo capitolo descrive la struttura del progetto dei sorgenti del software applicativo e la base di dati, che costituiscono nel loro insieme il progetto di dettaglio;
- il quarto capitolo descrive l'implementazione di un insieme di elementi appartenenti all'architettura del sistema software e le funzioni server di alto livello.

1 ANALISI

In tale capitolo sono state descritte le premesse di progetto del sistema software eCentral versione 1.0 e le sotto-fasi di determinazione dei requisiti e delle specifiche al fine di garantire che le attività di progetto e la realizzazione del sistema software convergano verso le aspettative attese dal committente in merito al prodotto desiderato.

1.1 Premesse di progetto

1.1.1 Validità

L'applicativo software eCentral, commissionato dalla multiutility CPL Concordia, supporta l'attività di gestione degli impianti dei propri customers del mercato gas.
eCentral, tramite uno o più device di:

- misura,
- acquisizione e/o controllo di vario tipo quali: convertitori di volume, sistemi di controllo di odorizzatori, misuratori di tasso di odorizzante, dispositivi di controllo per la protezione catodica, data-logger, telecontrolli,

installati nei pressi del corrispondente impianto, compie le seguenti attività:

a. acquisizione in real time dei dati associati a misurazioni rilevate sull'impianto o dello stato dello stesso, con semplici politiche automatiche o su esplicita richiesta dell'utente (ronda estemporanea sollecitata);
b. visualizzazione dello stato di odorizzazione della rete e di impianti;
c. programmazione dei device (impostazione dei parametri di funzionamento e di monitoraggio, in particolare, l'aggiornamento del firmware);
d. invio al singolo device di comandi con impatto diretto sugli azionamenti di impianto (ad esempio, chiusura di una valvola), su sollecitazione dell'utente.
e. gestione degli allarmi;
f. gestione dello storico degli eventi ed errori verificatisi nell'impianto;
g. gestione della reportistica.

Dagli scopi di eCentral si escludono:

- il supporto diretto alle attività di pianificazione, gestione ed esecuzione degli interventi programmati;
- l'integrazione con sistemi software di gestione di attività di pianificazione, gestione ed esecuzione degli interventi programmati e di gestione delle richieste di servizio provenienti dai customers di CPL Concordia;
- qualunque tipo di supporto automatico delle attività, ad eccezione della presentazione degli stessi in varie forme (es. grafici, report) o esportazioni in formati standard, anche controllati da meccanismi di filtro;

- qualunque genere di elaborazione funzionale ai processi di billing, qualora non banale;
- supporto diretto alla gestione del processo di fatturazione per l'accesso al software stesso, da parte del customer.

I benefici che si traggono dall'utilizzo di eCentral sono:

- abbattere i costi di manutenzione ordinaria delle apparecchiature in campo, incluse le parti di ricambio (per non sovraccaricare i sistemisti), evitando che il sistema debba essere presidiato;
- gestire i consumi con estrema precisione, con possibilità di personalizzare il servizio offerto ai propri customers;
- identificare perdite e guasti. Si controlla, sulla base dei dati acquisiti dai sensori, che le parti, cioè i customers, rispettino i contratti, non consumando più di quanto stipulato.

Si specifica che nell'ambito di tale tesi si è limitato il campo d'azione dell'applicativo eCentral alla sola odorizzazione.

1.1.2 Definizioni

person	Un utente identificato tramite meccanismi di autenticazione
plant	Un impianto
device	Uno specifico dispositivo HW/SW per l'acquisizione di dati e/o il controllo (per esempio: convertitori di volume, sistemi di controllo di odorizzatori, misuratori di tasso di odorizzante, dispositivi di controllo per la protezione catodica, data-logger), che è installato, quando operativo, nei pressi dell'impianto di competenza e con il quale è possibile stabilire un'interazione diretta remota su canale di comunicazione digitale, secondo differenti schemi di configurazione: i. fornisce sia le prestazioni di misura e/controllo sia quelle di comunicazione remota su uno o più canali di comunicazione (schema di configurazione *stand-alone*); ii. fornisce le prestazioni di misura e/controllo ed è connesso ad un secondo dispositivo hardware/software locale che fornisce le prestazioni di comunicazione remota su uno o più canali di comunicazione, sui cui/quali agisce in nome e per conto del primo dispositivo (la cui presenza è di fatto remotamente mascherata nel rispetto dello schema di configurazione *classic-pair*).
device model	L'insieme più ampio possibile di dispositivi concreti, caratterizzati dal medesimo uso richiesto dal punto di vista progettuale, e del tutto identici a meno di aspetti di identificazione e/o firmware e/o software quali: - serial number; - versione del firmware installato; - parametrizzazione effettuata. A parità di questi (serial number esluso), due device appartenenti al medesimo device model hanno identico comportamento.
gruppo di riduzione	Il complesso (assiemato) costituito da regolatori di pressione, da apparecchi ausiliari, da tubazioni, da raccordi e pezzi speciali, aventi la funzione di ridurre la pressione del gas canalizzato da un valore di pressione in entrata variabile a un valore di pressione in uscita predeterminato, fisso o variabile.
odorizzazione	Poiché in natura il gas è privo di odori, al fine di individuare eventuali perdite da tubazioni ed impianti, viene normalmente aggiunta al gas una sostanza (THT o TBM) che conferisce ad esso un odore caratteristico, facilmente individuabile.

impianto (o cabina) di odorizzazione a lambimento	Un impianto (o cabina) per l'immissione dell'odorizzante in rete per mezzo di un serbatoio a lambimento.
impianto (o cabina) di odorizzazione ad iniezione	Un impianto (o cabina) per l'immissione dell'odorizzante in rete per mezzo di un sistema ad iniezione
punto di odorizzazione	Un punto del sistema di odorizzazione nel quale viene immesso odorizzante in rete (collocato all'interno di un impianto o cabina).
punto di monitoraggio (o di rilievo) di odorizzazione	Punto nel quale viene monitorato il tasso di odorizzazione della rete (collocato all'uscita di un impianto o cabina).
sistema di odorizzazione semplice (o sistema di odorizzazione)	Una rete di distribuzione gas naturale, comprensiva di tutti i punti di rilevo odorizzante, la cui odorizzazione dipende da un unico punto odorizzazione principale (cabina REMI nella quale viene immesso in rete l'odorizzante).
sistema di odorizzazione multiplo	Una rete di distribuzione gas naturale, comprensiva di tutti i punti di rilevo odorizzante, la cui odorizzazione dipende da più punti di odorizzazione principali (cabina REMI nella quale viene immesso l'odorizzante in rete). Fanno parte di questa categoria di sistemi le reti di distribuzione che vengono alimentate da più cabine REMI (due o più) che sono interconnesse tra di loro lato in media pressione. Tali impianti si dicono "fisicamente interconnessi".
LowLevelSignal (LLSignal)	Unità atomica di basso livello di una comunicazione.
HighLevelSignal (HLSignal)	Unità atomica di alto livello di una comunicazione.
device class	Un insieme di device model affini, ossia caratterizzati (per ciò che concerne le funzionalità di interesse di eCentral) da uguali intended-use e da proprietà omogenee.
customer	Una società o un'impresa di distribuzione cliente di CPL. Utenza finale come cliente di CPL.
riempimento	Un caricamento di odorizzante.
ronda estemporanea	Attività di acquisizione dati improvvisata, occasionale, "now", alla quale viene data una risposta immediata. Può essere sollecitata, se lanciata da un utente attraverso opportuno comando, non sollecitata, se effettuata da uno schedulatore con una data frequenza temporale.
contatti reperibili	Utenti che sono potenziali target, su base periodica, di comunicazione non sollecitata in uscita dal sistema; ogni contatto reperibile è, perciò, caratterizzato da una specifica di reperibilità, che definisce gli intervalli di tempo in cui il contatto reperibile è contattabile.
communication failure	Classificati in: 1. on-going communication failure detection; 2. idle-channel diagnostic failure result; 3. device-detected communication failure: communication failure individuato da un device e comunicato successivamente ad eCentral.
stakeholder	Le persone che hanno un ruolo in un progetto software. Qualunque persona influenzata dal sistema e che ha influenza sullo sviluppo del sistema è uno stakeholder. Ci sono due principali gruppi di stakeholder: 1. utilizzatori (o proprietari del sistema); 2. sviluppatori (analisti, progettisti e programmatori).

1.1.3 Acronimi

SNAM	Società NAzionale Metanodotti, che è la principale società italiana di trasporto del gas naturale e l'unico operatore di rigassificazione del gas naturale liquefatto in Italia
LAN	Local Area Network

IP	Internet Protocol
TCP	Transmission Control Protocol
FTP	File Transfer Protocol
AJAX	Asynchronous Javascript and XML
DB/DBMS/R-DBMS	Database/Database Management System/Relational Database Management System
GUI	Graphical User Interface
CSV	Comma Separated Values
KPI	Key Performance Index: set più significativi di numeri per capire come va un sistema di odorizzazione (associati a HighLevelSignal)
PDA	Personal Digital Assistant
ERP	Enterprise Resource Planning
TO	Tasso di odorizzante
DTE	Data Terminal Equipment (computer)
DCE	Data Communication Equipment (modem)

1.1.4 Descrizione generale

1.1.4.1 Scenario di odorizzazione

Un impianto di distribuzione è definito come l'insieme di condotte, raccordi e curve integrate funzionalmente, per mezzo delle quali è esercitata l'attività di distribuzione, ossia di trasporto del gas naturale per la consegna ai customers. Nella tabella 1.1 viene proposto un esempio di sistema di odorizzazione nel rispetto della Parte II del Testo Unico della regolazione della qualità e delle tariffe dei servizi di distribuzione e misura del gas (RTDG):

Tabella 1.1: sistema di odorizzazione

Sistema di odorizzazione	Lunghezza e n. Gruppi di riduzione	Comuni
Provincia 1	Media pressione Km 95 Bassa pressione Km 120 GR 2° salto n. 2 GRF n. 48	Comune X Comune Y Comune Z

Il gas naturale, proveniente dai metanodotti di alimentazione SNAM Rete Gas di proprietà della società di trasporto, transita alla pressione nominale di consegna (70bar) attraverso le **cabine di 1° salto** (denominate *REMI* = regolazione e misura) di proprietà della società distributrice (customer di CPL), nelle quali viene:

1. *preriscaldato*;
2. *ridotto* secondo due differenti modalità:
   2.1. da 70bar a 20bar per alimentare la rete di *alta pressione (AP)*, stabilita come maggiore di 5bar (vedi Figura 1.1);
   2.2. da 70 bar a 4,5bar per alimentare la rete di *media pressione (MP)*, stabilita come compresa tra 40mbar e 5bar (vedi Figura 1.2);
3. *misurato*;

4. *odorizzato* attraverso l'immissione dell'odorizzante (THT/TBM) con sistema a lambimento o a iniezione.

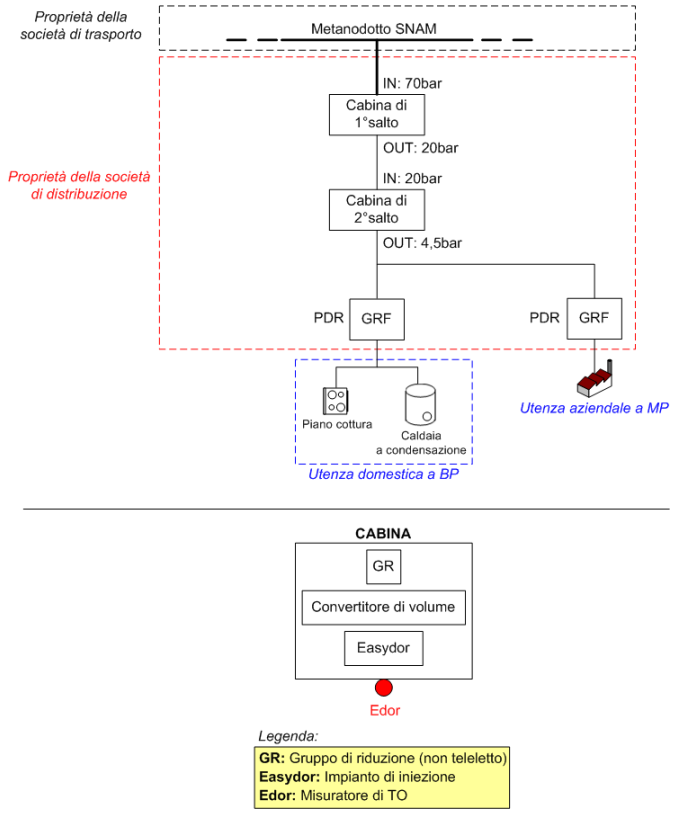


Figura 1.1: prima modalità

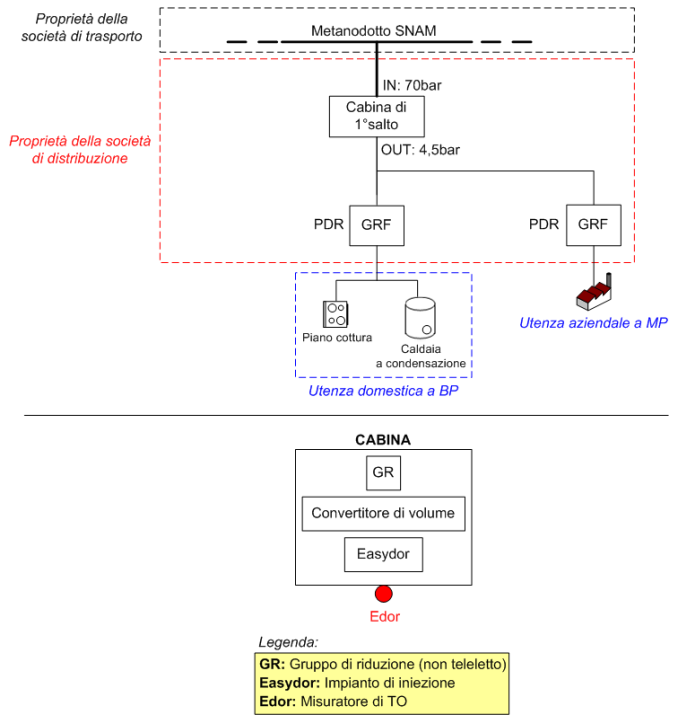


Figura 1.2: seconda modalità

In un punto più vicino possibile alla presa sul metanodotto SNAM è situata la cabina REMI di 1° salto, al cui interno sono ubicati i device, che consentono di determinare e controllare i volumi e le portate.
Nel caso di configurazione '2.1' della cabina di 1° salto, occorre una cabina di 2° salto (generalmente realizzata in armadi metallici) per decomprimere il gas da 20bar a 4,5bar ed immetterlo nella rete di distribuzione a *media pressione (MP).*
Il gas a *media pressione* raggiunge, poi, i punti di erogazione che, tramite l'installazione di opportuni misuratori, alimentano i **punti di riconsegna (PDR)** per la distribuzione locale. Un **PDR** è il punto di confine tra l'impianto di proprietà della società distributrice di gas e l'impianto del cliente finale o utente (famiglia X, azienda Y), in cui è contenuto il gruppo di riduzione finale (**GRF**) e al cui ingresso il gas arriva sempre a *media pressione (MP).*
Per la distribuzione locale il **GRF** alimenta:

- l'utenza domestica a *bassa pressione (BP)*, riducendo ulteriormente la pressione del gas da 4,5bar a 22mbar;
- l'utenza aziendale (industriale) a *media pressione (MP).*

Tutte le reti di distribuzione sono costituite da tubazioni in acciaio.
Come noto la legge 1083/71 pone in capo alla società distributrice la responsabilità e l'onere della corretta odorizzazione del gas naturale distribuito. La gestione del processo di odorizzazione riguarda la rilevazione sistematica del tasso di odorizzante immesso in rete dagli impianti, assicurando il buon funzionamento degli impianti stessi.
Analisi periodiche del gas in rete, come prescritto dalle normative, sono eseguite nei periodi di massima e minima portata da personale specializzato esterno sulla base di quanto prescritto dalla Delibera 168/04 AEEG. Tali verifiche consentono di garantire il controllo dell'effettivo grado di odorizzazione presente in ogni punto della rete, individuando anomalie sulla rete (dispersioni, deficienze gas, ecc.). Poiché il costo dell'attività di manutenzione, realizzata attraverso l'invio di tecnici sul campo per effettuare le misurazioni, è elevato, è emersa l'esigenza da parte della multiutility CPL Concordia di munirsi di un sistema software con cui svolgere un più efficiente e meno costoso telecontrollo. Tipicamente la misura della concentrazione dell'odorizzante viene realizzata a valle dell'impianto di erogazione del gas.
Al dominio applicativo "odorizzazione" afferiscono differenti *Device Class*, perché alla fruizione di un certo servizio a livello applicativo partecipano più classi di device. Per ogni classe il numero di device è stimato tra le 1'000 e le 10'000 unità.

Le *Device Class* sono le seguenti:
1) misura tasso di odorizzante (TO): comprende sensori per il monitoraggio del tasso di odorizzazione del gas naturale, utilizzato come stazione di rilevamento fissa. Il tipo di odorizzante rilevato può essere di due tipi:

– TBM: Mercaptani;
– THT: Tetraidrotiofene.

Tali sensori possono essere installati presso:

– impianti di odorizzazione per il monitoraggio dell'erogazione dell'odorizzante,
– punti intermedi di elevata rappresentatività dello stato della rete,
– terminali di rete per avere una misura precisa di quanto odorizzante giunge all'utente.

2) Iniezione di odorizzante: comprende sistemi di odorizzazione ad iniezione ideati per garantire un tasso di odorizzazione costante indipendentemente dalla portata di gas, dal tipo di odorizzante impiegato e dal tasso di odorizzazione richiesto. La possibilità di controllare esattamente la quantità di odorizzante immessa in rete è dovuta ai controlli retroazionati eseguiti dalla centralina in real-time.

3) Telelettura fiscale: comprende sistemi automatici di lettura dei contatori Acqua, Luce e Gas. Consiste nell'equipaggiare i contatori di sistemi di conteggio elettronici e nel trasferire tali informazioni tramite tecnologie cablate (su cavi elettrici) tipo onde convogliate o tecnologie Wireless (senza fili, ovvero via radio). La telelettura è uno strumento efficace sia per il miglioramento dell'efficienza operativa, sia per la possibilità di una personalizzazione del servizio offerto ai propri customers.

4) Previsione del volume di gas: comprende i dispositivi in grado di leggere frequentemente il volume di gas istantaneo che l'impianto sta erogando periodicamente, con intervallo impostato dall'utente, prelevare e memorizzare un valore di volume, eseguire il processo previsionale (o consuntivo) e, in caso di prevista (o reale) eccedenza di volume, avvertire il customer via SMS/e-mail, per garantire il bilanciamento del sistema. Poiché, infatti, ora i nuovi termini contrattuali si riferiscono alla capacità giornaliera (Cg), il rispetto dell'impegno previsto consente di evitare eccedenze di portata, evitando i conseguenti esborsi per penali da parte del customer.

5) Data logger: comprende device sviluppati per acquisire segnali digitali ed analogici per controllare impianti ed eseguire teleletture di contatori gas, acqua e protezione catodica.

6) Telecontrollo: comprende i dispositivi per la sola lettura a distanza del dato, o la sola modifica a distanza dello "stato del sistema" (attraverso relè, attuatori, etc.); in entrambi i casi con la possibilità di registrare le variabili di interesse su diverse scale temporali ed ottenerne indicatori per vari scopi. Il monitoraggio continuo del funzionamento dei vari componenti, inoltre, permette di avvisare in modo automatico quando è il momento di eseguire manutenzioni preventive o straordinarie e sostituzioni di componenti, con benefici economici e gestionali.

Nella tesi sono state prese in considerazione le *Device Class* '1)' e '2)'.

I device sono, infine, classificati per modello, per cui si ha una conseguente suddivisione in *Device Model*. L'intersezione fra *Device Model* restituisce l'insieme vuoto, per cui i *Device Model* sono delle partizioni. Nel seguito viene mostrata la tabella 1.2 in cui sono elencate le principali caratteristiche dei *Device Model*.

Tabella 1.2: *Device Model*

Device Model	Canale di comunicazione	Protocollo	Periodicità comunicazioni	Device Class
EDOR	SMS	ASCII	1 / giorno	Odo monitor
EDOR XML	FTP	XML	4 / giorno	Odo monitor
EASYDOR	GSM (C.DATI) FTP	MODBUS XML	1 / giorno	Iniezione
ODOTRONIC	GSM (C.DATI) PSTN	MODBUS MODBUS	1 / giorno 1 / giorno	Iniezione Iniezione

DOSAODOR	X	MODBUS	1 / giorno	Iniezione
ECOM	GSM (C.DATI) PSTN	SNAM SNAM	1 / giorno 1 / giorno	Telelettura fiscale
FIOMEC 10	GSM (C.DATI) PSTN	SNAM SNAM	1 / giorno 1 / giorno	Telelettura fiscale
FIOMEC 12	GSM (C.DATI) PSTN	SNAM SNAM	1 / giorno 1 / giorno	Telelettura fiscale
FIOMEC 21	GSM (C.DATI) PSTN	SNAM SNAM	1 / giorno 1 / giorno	Telelettura fiscale
FIOMEC 22	GSM (C.DATI) PSTN	SNAM SNAM	1 / giorno 1 / giorno	Telelettura fiscale
VESCOM	GSM (C.DATI) PSTN	SNAM SNAM	1 / giorno 1 / giorno	Telelettura fiscale
ITI 782-10/VO	GSM (C.DATI) PSTN	SNAM SNAM	1 / giorno 1 / giorno	Telelettura fiscale
TARTARINI 502/VO	GSM (C.DATI) PSTN	SNAM SNAM	1 / giorno 1 / giorno	Telelettura fiscale
GENERICO	GSM (C.DATI) PSTN	SNAM SNAM	1 / giorno 1 / giorno	Telelettura fiscale
ACTARIS COMPLEX VOLUMETRICO	GSM (C.DATI) PSTN	SNAM SNAM	1 / giorno 1 / giorno	Telelettura fiscale
ACTARIS COMPLEX VENTURIMETRICO	GSM (C.DATI) PSTN	SNAM SNAM	1 / giorno 1 / giorno	Telelettura fiscale
ACTARIS PTZ	GSM (C.DATI) PSTN	SNAM SNAM	1 / giorno 1 / giorno	Telelettura fiscale
ACTARIS SEVC-D	GSM (C.DATI) PSTN	SNAM SNAM	1 / giorno 1 / giorno	Telelettura fiscale
ACTARIS EMETER	GSM (C.DATI) PSTN	SNAM SNAM	1 / giorno 1 / giorno	Telelettura fiscale
EFOR2	MAIL / C.DATI / FTP	SNAM ASCII XML	1 / giorno 1 / giorno	Previsionale
TLOG/ELOG	SMS	ASCII	1 / giorno	Datalogger
GOLIAH	SMS GPRS	? ?	1 / giorno	Datalogger
EMET SMS	SMS	ASCII	1 / settimana	Datalogger
EMET XML	FTP / XML	ASCII	1 / settimana	Datalogger
TBOX	SMS FTP / XML	ASCII	1 / giorno	Telecontrollo
TL5	GSM (C.DATI)	T-PROTO (binario)	1 / giorno	Telecontrollo
MWS	GSM (C.DATI)	ASCII	1 / giorno	Telecontrollo

Come si può osservare in tabella un device invia i propri dati nel rispetto della scadenza temporale e del formato del protocollo di comunicazione definiti per il *Device Model* di appartenenza.

Nella tesi è stato preso in considerazione il *Device Model Edor* afferente alla *Device Class* ‘1)’.

1.1.4.2 Prospettive di impiego

eCentral è un sistema informativo che integra un motore di controllo di sensori per il monitoraggio della corretta odorizzazione del gas naturale. eCentral integra un insieme di funzionalità, tra cui l'interazione con una base di dati, la gestione delle transazioni, un'interfaccia utente evoluta e la distribuzione del carico applicativo. eCentral sfrutta:

1. le potenzialità informative, navigazionali e distribuite del Web;
2. piattaforme integrate per la gestione applicativa (Framework .NET di Microsoft) e dei dati (SQL Server 2005).

Il Web, per la sua diffusione, flessibilità e versatilità si presta bene al ruolo di canale di comunicazione per fornire le informazioni. Esso offre un'infrastruttura, il cui modello di comunicazione si basa principalmente sul protocollo *HTTP* (*HTTPS*), sulla tecnologia *AJAX*, sul linguaggio *HTML* e sul modello client-server. L'utilizzo del Web permette di personalizzare facilmente la GUI e di migliorarne l'efficacia. Poiché l'applicazione eCentral utilizza il Web, non richiede un deployment specifico per ogni tipo di piattaforma, garantendone la massima portabilità. Si ricorda che eCentral non è unidirezionale come un sito Web, ma è bidirezionale, perché interagisce direttamente con l'utente fornendo servizi.

eCentral non è un applicativo completamente a sé stante, poiché tramite apposite API si interfaccia con il servizio di cartografia Web *Google Maps*, per offrire la visualizzazione geolocalizzata delle rete di distribuzione. Esso può, inoltre, interagire in modo unidirezionale con altri sistemi software, come sistemi ERP (ad esempio, l'ERP di ENI), per importare o esportare dati. L'interoperabilità di eCentral con tali sistemi è attuata attraverso opportune interfacce (ad esempio *Web Service*, che sfruttano standard quali HTTP, XML) oppure attraverso la sincronizzazione tra il DBMS di eCentral e quello di tali sistemi. Il modello di interazione del sistema software è di tipo client-server. eCentral è utilizzabile da diverse figure professionali in funzione delle loro esigenze. L'architettura software del sistema eCentral, mediante un'appropriata decomposizione in componenti, dovrebbe consentire, per ogni *Device Model* supportato, un disaccoppiamento (logico, implementativo) tra:

- la funzionalità di sistema
- il protocollo di comunicazione con *Device Model*

ed in generale, dati due *Device Model* differenti, tra le implementazioni dei due rispettivi protocolli di comunicazione.

L'architettura deve essere di tipo non distribuita, quindi l'intelligenza deve essere concentrata nel sistema software, la cui funzione principale è quella di controllare, sulla base dei dati acquisiti dai sensori, che la rete sia correttamente odorizzata. Ogni nodo (device) fa riferimento al sistema software secondo una topologia a stella pura.

1.1.4.3 Caratteristiche utenti (o attori)

Person	Funzione
Utente customer	Visualizzazione dati degli impianti inseriti nelle viste assegnate, stampa report (non permessa la visualizzazione dei dati dei device)

Gestore dei tecnici in campo	Visualizzazione dati dei device inserite nelle viste assegnate, stampa report, lancio di ronde estemporanee sui propri device (nel caso in cui il device lo permette)
Tecnico	Visualizzazione dati dei device inserite nelle viste assegnate, stampa report, lancio di ronde estemporanee sui propri device (nel caso in cui il device lo permette), modifica della configurazione di device già esistenti (aggiornamento del firmware), esecuzione di telecomandi (es. chiusura di una valvola da remoto)
Amministratore	Creazione di device, creazione di tipologie, configurazione di allarmi, creazione di reperibili, configurazione di reperibili, assegnazione di reperibili a device, creazione di sinottici, configurazione di tabelle di device (scelta di visualizzazione per periodi di tempo prestabiliti, per esempio: ultimi 7, 15, 30 giorni), creazione di grafici statistici, creazione di ronde (schedulate e estemporanee)
Super Amministratore	Ha tutte le funzioni dell'Amministratore e può, inoltre, creare utenti e collegarli alle viste e impostarne le proprietà

1.2 Fase di determinazione dei requisiti

La determinazione dei requisiti è stata la prima fase del ciclo di vita dello sviluppo di eCentral ed è stata svolta in modo rigoroso per evitare di commettere errori che sarebbero potuti risultare più dannosi di quelli commessi nelle fasi successive, infatti i costi per gestire i requisiti omessi o non correttamente interpretati possono diventare insostenibili. Si sono rese necessarie, pertanto, la collaborazione con l'Ing. Cavazzoni Andrea (nel ruolo di committente), responsabile del dipartimento ICT di CPL, e la supervisione dell'Ing. Barbi Andrea. La raccolta dei requisiti è stata condotta utilizzando le seguenti due tecniche:

- supporto di prototipi;
- sviluppo cooperativo delle applicazioni (JAD).

Tale fase ha prodotto in output il *documento dei requisiti*, con cui è stata data una definizione informale di tale requisiti, poiché sono stati descritti con asserzioni in *linguaggio naturale*. Una volta accettati dal committente, i requisiti sono stati definiti, classificati, enumerati ed ordinati (secondo certe priorità) nel *documento dei requisiti*. La struttura dei requisiti è gerarchica con relazioni simili a quelle di composizione. Viene nel seguito proposto a titolo di esempio un numero limitato di requisiti tra quelli esaminati [1].

1.2.1 ReqID 1 Misura del livello di odorizzante

Alcuni impianti forniscono il segnale di livello come altezza del liquido in un serbatoio orizzontale, ma al fine di valutare il consumo di odorizzante in litri o kg è necessario applicare una formula matematica che mette in relazione la quantità di odorizzante contenuta nel serbatoio con l'altezza del pelo libero dell'odorizzante all'interno del serbatoio. Tale formula può essere sostituita con tabelle di riferimento, che sono diverse per ogni serbatoio.

1.2.2 ReqID 2 Consumo di odorizzante

Il consumo di odorizzante nel periodo richiede un'elaborazione da parte del sistema. Se il device, che si occupa del rilievo del livello di odorizzante, ne fornisce anche il consumo (giornaliero, settimanale mensile), tale dato viene acquisito e memorizzato dal sistema. Quando l'operatore effettua un'interrogazione per conoscere lo stato di funzionamento nel periodo, il valore di consumo di odorizzante deve essere calcolato come somma dei valori di consumo registrati, se consentita dalla discretizzazione del campionamento. In altri termini, se viene richiesto il valore del consumo di odorizzante in *n* giorni e la sua memorizzazione è giornaliera, è possibile calcolare tale consumo come somma dei consumi dei corrispondenti *n* giorni. Viceversa, se il consumo viene fornito dal device attraverso un campionamento settimanale del suo valore, è necessario effettuare il calcolo partendo dalle misure di livello di odorizzante registrate.

Dopo aver effettuato l'eventuale "correzione" della misura di livello, il sistema effettua la valutazione dei consumi di odorizzante nel periodo *T* richiesto dall'operatore; si possono presentare due casi e, precisamente, il device in campo fornisce:

- il volume di odorizzante presente nel serbatoio di stoccaggio,
- l'altezza del livello di odorizzante all'interno del serbatoio di stoccaggio; in questo secondo caso è necessario definire nell'anagrafica dell'impianto i parametri del serbatoio di stoccaggio per consentire al sistema di calcolare il volume di odorizzante presente nel serbatoio.

Per individuare la presenza di un riempimento all'interno del periodo selezionato, è necessario valutare due misure di livello compiute ad ogni intervallo di tempo ΔT_C (impostato nel device) che può coincidere con il periodo di campionamento del device. Se la differenza tra tali misure è negativa (aumento del livello di odorizzante) e superiore ad una soglia, viene identificato un riempimento nell'intervallo di tempo analizzato. Il periodo di analisi richiesto dall'operatore viene suddiviso in un numero $\Delta T_C + 1$ di intervalli di calcolo; all'interno di ciascun intervallo il consumo sarà dato dalla differenza tra il valore maggiore ed il valore minore di odorizzante, mentre nel periodo sarà dato dalla somma dei $\Delta T_C + 1$ consumi parziali.

Il calcolo del consumo giornaliero di odorizzante deve essere, invece, effettuato mediante la differenza tra i livelli di odorizzante misurati alla stessa ora di due giorni consecutivi (in presenza di un riempimento è necessario suddividere il periodo di calcolo come sopra illustrato).

1.2.3 ReqID 3 Struttura di un punto di monitoraggio

In un tale punto, detto anche di rilievo, possono essere presenti più device; la configurazione massima consentita è rappresentata da:

1. un impianto d'iniezione;
2. un device Edor;
3. un device di telecontrollo;

4. un device di monitoraggio dei consumi gas metano;

Per ogni punto di rilievo deve essere presente almeno uno dei device sopra citati.

È possibile, inoltre, che su tale punto di monitoraggio venga eseguito un rilievo manuale da parte di un operatore (inserimento manuale della misura).

L'impianto di odorizzazione comprende anche un serbatoio di stoccaggio dell'odorizzante, caratterizzato dai seguenti parametri:

– costruttore;
– volume nominale;
– misure per il calcolo del volume di odorizzante contenuto a partire dalla misura di livello;
– tipo di odorizzante impiegato (TBM, THT).

Nel caso di più periferiche, esse provvederanno a fornire le stesse misure; si consideri, ad esempio, il caso del TO, che può essere acquisito da:

– un impianto d'iniezione (Easydor);
– device Edor;
– telecontrollo.

Quando sono fornite contemporaneamente più misure da parte di device diversi associati al medesimo punto di monitoraggio, è necessario prendere in considerazione solamente la misura con la priorità maggiore per il calcolo del KPI.

1.2.4 ReqID 4 Stato del sistema di odorizzazione

Il sistema software deve aggiornare i dati del sistema di odorizzazione con le ultime letture disponibili, per determinarne lo stato corrente, indice dell'odorizzazione della rete. Per definire tale stato è necessario verificare gli ultimi dati, disponibili nello stesso momento (anche se con una certa tolleranza temporale da definire) di:

– portata gas,
– TO,
– livello odorizzante,

di un punto di immissione dell'odorizzante. Si tollera, pertanto, un errore nella sincronizzazione dell'invio di tali misure provenienti dai device, purché tale errore sia contenuto entro un certo margine ammesso. I "buchi" nei dati dovranno essere colmati con algoritmi specifici.

1.2.5 ReqID 5 KPI di un sistema di odorizzazione

Per un sistema si definiscono i seguenti criteri di valutazione dell'odorizzazione:

- **qualità dell'odorizzazione:** come deviazione rispetto al funzionamento ottimale atteso (programmato). Quanto si è vicini al punto di lavoro.
- **sicurezza dell'odorizzazione:** rispetto normative sull'odorizzazione minima, quanto si odorizza in meno rispetto alla soglia della normativa.

- **efficienza del monitoraggio:** quanto si odorizza in più rispetto alla soglia della normativa, ovvio che si spende di più ma la sicurezza è garantita. Poiché l'odorizzante costa molto conviene comunque evitare sprechi per contenere i costi.

Nel calcolo del KPI si deve tenere conto di tutti e tre i criteri.

1.2.6 ReqID 5.1 Qualità dell'odorizzazione

Esprime la bontà dell'odorizzazione compiuta dai punti di odorizzazione (che immettono l'odorizzante nel gas). Per tali punti si definiscono il livello di Tasso di odorizzazione ottimale TO_o comprensivo di un margine di sicurezza ΔTO_o.

La qualità $\mathbf{D_i}$ dell'odorizzazione del punto i-esimo di odorizzazione è espressa dalla differenza tra il suo livello di odorizzazione attuale in un punto (TO) ed il tasso di odorizzazione ottimale (TO_o) comprensivo del margine di sicurezza (ΔTO_o):

$$\mathbf{D_i = TO - (TO_o + \Delta TO_o)}$$

1.2.7 ReqID 5.2 KPI del sistema (istantaneo)

Poiché ogni punto i-esimo di odorizzazione influisce con il proprio peso K_i sull'odorizzazione dell'intero sistema, il KPI viene calcolato dalla seguente formula:

$$\mathbf{KPI = \sum K_i * D_i / \sum K_i} \qquad \text{con } \mathbf{K_i = 0 \div 10} \text{ (10 per impianto di maggiore importanza)}$$

Dovendo il valore del KPI essere compreso tra 0 e 100 si applicherà un'opportuna funzione di normalizzazione. Il valore 100 rappresenta la totale assenza di anomalie strumentali ed una perfetta efficienza.

1.2.8 ReqID 5.3 Sicurezza del sistema (istantaneo)

Se $\mathbf{D_i < 0}$, il sistema è in condizioni di non sicurezza. Questo caso si verifica:

- di notte, quando il consumo di gas si riduce e l'odorizzante ristagna, per cui la "puzza" di gas si percepisce di meno e falsa il calcolo del KPI,
- a causa della insufficiente copertura del sistema, per cui tra punti di odorizzazione molto distanti si può rilevare un tasso di odorizzazione sotto la soglia consentita.

Si potrebbe aumentare il tasso di odorizzante in uscita dal singolo impianto per riportare il sistema in sicurezza, ma questo avverrebbe a scapito dell'efficienza, poiché si avrebbe una sovra-odorizzazione del sistema. Conviene, allora, "penalizzare" il suo KPI, dando un maggiore peso ai D_i negativi, che dovranno essere moltiplicati per un fattore $\mathbf{P_i}$ e precisamente:

$$\mathbf{D_i = D_i * P_i} \qquad \mathbf{se\ D_i < 0}$$

perché la sicurezza deve influenzare il KPI in modo prioritario.

1.2.9 ReqID 5.4 Efficienza del monitoraggio del sistema (istantaneo)

Se $\mathbf{D_i > 0}$ il sistema è correttamente odorizzato, ma un suo valore assoluto ($abs(D_i)$) eccessivamente elevato comporta una sovra-odorizzazione, con conseguente danno economico perla fornitura di odorizzante. Conviene, allora, "penalizzare" il suo KPI, dando un minor peso ai D_i di valore superiore ad una certa soglia (di efficienza dei costi), che dovranno essere moltiplicati per un fattore $\mathbf{S_i}$ e precisamente:

$$\mathbf{D_i = D_i * S_i} \qquad \mathbf{con: 0 < S_i < 1}$$

1.2.10 ReqID 6 Diagnostica

La diagnostica può essere:

1. *esplicita (device)*, quando il device rileva uno stato di allarme e lo segnala al sistema in modo esplicito in una sua comunicazione, ad esempio, inviando un SMS contenente "TO sottosoglia";
2. *implicita (sistema)*, quando il sistema rileva uno stato di allarme, ad esempio, riconoscendo che un device ha il punto di lavoro fuori specifiche, non è più alimentato, etc.

Essa può essere richiesta:

a. se i dati acquisiti sono fuori media (o fuori soglia), ad esempio:
   i. TO troppo basso (sotto la soglia ammessa, quindi fuori legge) per garantire un'odorizzazione in sicurezza della rete;
   ii. TO troppo alto (sopra la soglia stabilita) per garantire un'odorizzazione efficiente della rete;
b. se i dati sono incoerenti (si è compiuto un riempimento di 5lt di odorizzante con una giacenza iniziale di 30lt e il device comunica che la giacenza attuale è di 100lt);
c. se si verifica una condizione di errore del device:
   i. non è alimentato;
   ii. non invia dati da più di 24 ore;
d. se si verifica una condizione di errore del sistema;
e. se si verifica una condizione di errore del canale di comunicazione.

Gli allarmi sono classificati per tipo di *Device Model*.

Il sistema deve essere in grado di rilevare un'anomalia dell'odorizzazione, considerando gli impianti nel loro insieme; ad esempio, se i valori di portata gas di più impianti differiscono per più di un certo Δ, l'eccessiva disomogeneità deve essere interpretata come un allarme.

Quando viene diagnosticato un allarme occorre notificarlo al reperibile competente, che dovrà provvedere alla risoluzione del problema entro un certo intervallo di tempo dal momento in cui ha confermato la ricezione dell'avviso. Il sistema provvederà ad inoltrare l'allarme ad un altro reperibile in caso di mancata risoluzione entro il tempo previsto.

1.2.11 Vincoli di sistema

eCentral consente agli utenti di eseguire in modo interattivo l'applicativo dalla macchina client tramite l'utilizzo di un browser Web; si può invocare ogni funzione mediante selezioni con mouse sugli appositi pulsanti provvisti di label descrittive.

Per quanto concerne l'interfaccia hardware:

- lo strumento applicativo a disposizione dell'utente dovrebbe supportare almeno:
  1) MS Windows 2003/2008 Server
  2) MS Windows XP Professional
  3) MS Windows Vista

  ed essere in grado di girare su una piattaforme PC con le seguenti caratteristiche:
  - CPU Intel® Pentium IV® (o compatibile) 800 MHz o superiore
  - memoria RAM 512 MB (o più)
  - spazio libero su hard disk pari almeno a 300 MB.
- I componenti server di sistema possono limitarsi a supportare i seguenti sistemi operativi:
  1) MS Windows 2003 Server
  2) MS Windows 2008 Server

  ed essere in grado di girare su piattaforma PC con le seguenti caratteristiche:
  - CPU Intel® Core 2 Quad Q6600 (o compatibile) o superiore
  - memoria RAM 4 GB (o più)
  - spazio libero su hard disk pari almeno a 8 GB.

Il protocollo di comunicazione macchina client-server eCentral è il TCP/IP. Per la connessione è consigliato l'utilizzo di una connessione a banda larga di almeno 1Mbit/s.

Si garantisce che il tempo di risposta di eCentral non superi 3 secondi e nelle condizioni di picco si mantenga stabile. Poiché il sistema è tuttora in fase di sviluppo, nel momento della stesura dell'elaborato non risultano ancora definite le policy di sicurezza. Si richiede il log della comunicazione a basso livello (registro dei pacchetti inviati dal protocollo a basso livello) con visibilità in real-time durante una specifica sessione di comunicazione, per meglio capire se il problema risulta essere:

- sul device
- o sul sistema eCentral.

eCentral deve essere sviluppato con l'ambiente integrato Visual Studio 2008.

I meccanismi di comunicazione con i device, ed in particolare quelli chiamati in causa in eventi di comunicazione avviati "su iniziativa del dispositivo", dovrebbero utilizzare un protocollo di comunicazione in cui venga specificato il *Serial Number* e il *Device Model* nel contesto di ogni sessione di comunicazione, per identificare univocamente il device. Se viene fissato il legame tra *Serial Number* e *Device Model* in modo persistente sul DB e nel protocollo il device comunica solo il *Serial Number*, non si è in grado di indentificarlo univocamente, perchè si potrebbe avere lo stesso *Serial Number* ripetuto per diversi *Device Model*.

Un diverso meccanismo di identificazione fa uso dell'end-point di destinazione, ad esempio, se ai device

del *Device Model* Efor è associato un certo modem, essi contatteranno sempre quella SIM di destinazione (lato ricevente di eCentral). Si potrebbe utilizzare come criterio di identificazione la SIM univoca del chiamante oppure la cartella FTP di destinazione (ad esempio, la cartella *X* è utilizzata dagli Efor).

Riassumendo, i criteri di identificazione possono essere:

1) la coppia (*Serial Number* e *Device Model*) univoca nel sistema e con buone garanzie di univocità anche al di fuori del sistema;
2) l'assegnamento predeterminato di indirizzi degli end-point del canale di comunicazione (per esempio: il numero telefonico utilizzato dal device per chiamare, oppure l'indirizzo IP);
3) una combinazione di vari approcci.

Nella tesi è stato utilizzato il secondo caso.

Si precisa che il *Serial Number* e il *Device ID* (Device Identifier) sono due concetti distinti, in quanto il primo dei due viene assegnato al device dal costruttore a livello fisico, mentre il secondo viene utilizzato a livello applicativo.

Il sistema software eCentral supporta i seguenti tipi di canali di comunicazione con i dispositivi remoti:

1. connessione dati GSM;
2. connessione diretta PSTN;
3. SMS;
4. email (POP3 Rfc 1939, SMTP Rfc 2821);
5. FTP (text only file, XML file);
6. TCP/IP "vanilla" socket.

I canali di comunicazione, che si riferiscono a tipi di canale che si appoggiano su risorse a disponibilità particolarmente critica, quali *Connessione dati GSM* e *Connessione diretta PSTN*, dovrebbero essere impiegati con schemi idonei a ridurre la probabilità di conflitti di utilizzo ed a facilitare il troubleshooting. Il sistema software eCentral gestisce, in generale, gli atti di comunicazione in modo da indurre questo comportamento a livello di *layer device driver*. Ove non sussistano vincoli tecnici indotti dal design del device e a meno di indicazioni contrarie prioritarie, le risorse di comunicazione gestite da eCentral non dovrebbero essere assegnate staticamente a device o a gruppi di device. Il canale di comunicazione dovrebbe essere occupato solo al fine di soddisfare le richieste utente estemporanee o le attività pianificate propriamente dette.

La cabina di secondo salto non è alimentata a 220Veff e, pertanto, l'Easydor di tale cabina è alimentato da una batteria. L'Easydor è configurato da firmware per collegarsi al modem con una cadenza temporale prefissata per limitare il consumo della batteria. L'Easydor non è contattabile direttamente e, pertanto, per comunicare con esso, occorre conoscere l'istante in cui si collega al modem, oppure deve essere configurato perché contatti eCentral al risveglio.

1.3 Fase di specifica dei requisiti

In UML il comportamento esteriormente visibile e controllabile del sistema è descritto dai casi d'uso che durante l'analisi rappresentano i requisiti del sistema, enfatizzando cosa il sistema fa o dovrebbe fare. Un caso d'uso rappresenta una funzione di business che può essere controllata singolarmente nel processo di sviluppo ed è esternamente visibile ad un attore. Un attore rappresenta chiunque o qualunque cosa (persona, macchinario, etc.) interagisca (tramite un caso d'uso) con il sistema, per ottenerne un servizio o un qualunque risultato utile. Un caso d'uso rappresenta, pertanto, un'unità funzionale di interesse per un attore. Nel modello dello stato è necessario rappresentare gli attori (ad esempio, per controllarne i permessi di accesso) come un'entità, descritta da attributi. Un attore che non comunica con almeno un caso d'uso è privo d'interesse, mentre un caso d'uso, che non comunica con alcun attore, è permesso e può avere senso. Possono esistere, infatti, casi d'uso che generalizzano o specializzano altri casi d'uso e che non interagiscono direttamente con attori; essi possono essere usati internamente da altri casi d'uso per fornire certi risultati agli attori. Ad esempio, il caso d'uso UCX, che comprende sistematicamente il comportamento di UCY, utilizza la relazione *UCX <<include>> UCY*. La relazione <<*extend*>>, invece, descrive comportamenti opzionali e, dunque, non sempre attivato. Poiché i casi d'uso possono essere dedotti dall'identificazione dei compiti degli attori, una delle domande utilizzate è stata la seguente: "*Chi utilizza il sistema e quali sono i suoi obiettivi i cui risultati hanno un valore misurabile?*".
Ogni caso d'uso è iniziato da un attore ed è una funzionalità completa, esternamente visibile ed ortogonale alle altre. Gli attori possono attivare un caso d'uso o ricevere i risultati da esso.
L'output di tale fase è il *documento delle specifiche* che sostituisce quello dei requisiti. In tale fase è stata compiuta:

- l'identificazione e la descrizione in dettaglio dei casi d'uso e delle loro relazioni,
- l'estensione delle descrizioni iniziali per comprendere il sotto-processo principale (scenario principale di successo) e processi alternativi (estensioni),
- la costruzione di modelli dimostrativi di alcune schermate GUI (viste).

I modelli di specifica sono stati sviluppati in modo indipendente dalla piattaforma HW/SW sulla quale il sistema software eCentral deve essere implementato per non ridurre l'espressività del linguaggio di modellazione e per facilitare la comprensione da parte del committente del vocabolario utilizzato evitando di compromettere un dialogo costruttivo. Alcuni vincoli del committente hanno, tuttavia, influenzato alcune scelte HW/SW.
Nei casi d'uso di eCentral nel seguito proposti, si sono evitate ripetizioni (copia-incolla) di scarso valore ai fini della comprensione delle funzionalità offerte dal Sistema, come ad esempio nel caso di fallimento del Sistema, per descrivere in modo approfondito ciò che contraddistingue i casi d'uso nello scenario di successo. Poiché il Super Amministratore e l'Amministratore hanno poche restrizioni, vengono specificati nei casi d'suo solo quando strettamente necessario [2].

1.3.1 Gestisci schede tecniche device

CASO D'USO UC1: Gestisci schede tecniche device (CRUD = acronimo di Create, Retrieve, Update, Delete)

Portata: eCentral

Livello: Obiettivo utente

Attore primario: Amministratore/Operatore Data Entry. I due attori sono indicati con il nome unico di Utente.

Parti interessate ed Interessi (ciò che serve a soddisfare tutti gli interessi delle corrispondenti parti interessate)**:**

- Utente: vuole creare, modificare, eliminare, ricercare una scheda tecnica di un device in modo preciso e rapido.

Pre-condizioni (opzionale, non è un errore ometterlo, perché va specificato solo se significativo)**:** l'Utente ottiene il modulo di inserimento della scheda tecnica dal Sistema tramite il proprio browser web, dopo essersi autenticato.

Garanzia di successo (o Post-condizioni): la scheda tecnica del device in esame viene registrata, eliminata, aggiornata ed individuata (ricerca) nell'anagrafica ufficiale. Qualora si verificassero degli errori lo stato del Sistema non è modificato.

Scenario principale di successo:

1. L'Utente clicca il pulsante *New Device*.
2. Il Sistema visualizza il modulo di inserimento della scheda tecnica.
3. L'Utente compila la scheda tecnica, scegliendo in un opportuno elenco predefinito il DeviceModel a cui appartiene; setta un flag per indicare che il device è abilitato, mentre ne setta uno diverso per specificare il tipo di anagrafica, scegliendo tra "Anagrafica ufficiale" e "Nursery".
4. L'Utente clicca il pulsante *Save*.
5. Il Sistema richiede la conferma dell'operazione.

L'Utente ripete i passi 1-2-3-4-5 (oppure 3-4-5) fino a quando non conferma l'operazione.

6. L'Utente clicca il pulsante *OK*.
7. Il Sistema registra la scheda tecnica ed assegna automaticamente un ID univoco al device.
8. Il Sistema registra l'operazione in un file di log contenente l'ID dell'Utente e la data-ora dell'operazione.
9. L'Utente clicca il pulsante *Close*.

Estensioni (condizione + reazione):

*a. In qualsiasi momento il Sistema fallisce: si considera non valida l'alterazione dello stato del Sistema a fronte dell'operazione avvenuta. In particolare il Sistema esegue l'abort della transazione. Occorre, quindi, che:
    1. L'Utente si ri-autentichi e proceda con una nuova operazione.
    2. Il Sistema registri l'errore in un apposito repository di log.

*b. L’Utente è inattivo per un tempo maggiore del timeout di sessione:

1. Il Sistema chiude la connessione con il browser e il caso d’uso termina.

1a. L’Utente clicca il pulsante *Delete Single Device*:

1. Il Sistema mostra un campo in cui specificare il codice identificativo del device.
2. L’Utente inserisce il codice identificativo del device.

3a. Il codice identificativo non è presente, perché l’Utente si è sbagliato a digitarlo:

1. Il Sistema richiede un nuovo inserimento del codice identificativo del device.
2. L’Utente inserisce il codice (vai al punto 3b).

3b. Il codice identificativo è presente (vai al punto 4).

3c. Il codice identificativo non è presente, perché il device non è censito (vai al punto 1).

1b. L’Utente clicca il pulsante *Delete Group of Device*:

1. Il Sistema mostra una lista di device.
2. L’Utente compie una selezione multipla dei device da eliminare (vai al punto 4).

1c. L’Utente clicca il pulsante *Edit Device*:

1. Il Sistema mostra un campo in cui specificare il codice identificativo del device.
2. L’Utente inserisce il codice identificativo del device.

3a. Il codice identificativo non è presente, perché l’Utente si è sbagliato a digitarlo:

1. Il Sistema richiede un nuovo inserimento del codice identificativo del device.
2. L’Utente inserisce il codice (vai al punto 3b).

3b. Il codice identificativo è presente (vai al punto 4).

3c. Il codice identificativo non è presente, perché il device non è censito (vai al punto 1).

4. Il Sistema visualizza il modulo di inserimento della scheda tecnica.
5. l’Utente modifica i campi desiderati della scheda tecnica (vai al punto 4).

4a. L’Utente clicca il pulsante *Delete*.

7a. Il Sistema non riesce ad ottemperare alla richiesta:

1. Il Sistema notifica l’Utente dell’errore avvenuto.

7b. Il Sistema elimina la scheda tecnica del device.

Requisiti speciali:

- internazionalizzazione della descrizione del device (no brand).

Elenco delle varianti tecnologiche dei dati: /

Frequenza di ripetizione: /

Varie (Problemi aperti): /

1.3.2 Registra nuovo device dal campo

CASO D’USO UC2: Registra nuovo device dal campo

Portata: eCentral

Livello: Obiettivo utente

Collaborazione: UC2 include UC3

Attore primario: Tecnico, Amministratore

Parti interessate ed Interessi:

- Tecnico: vuole registrare un device in modo preciso e rapido dal campo mediante PDA (oppure mediante chiamata telefonica ad un operatore DataEntry) e vuole osservare facilmente l'esito del test di comunicazione con il device.
- Amministratore: vuole tenere sotto controllo le registrazioni dei device effettuate dal campo e, quindi, sarà opportunamente avvisato.

Pre-condizioni: il Tecnico è identificato ed autenticato.

Garanzia di successo (o Post-condizioni): la scheda tecnica del device viene registrata nell'anagrafica "Nursery".

Scenario principale di successo:

1. Il Tecnico inserisce il codice identificativo (hp: SerialNumber più DeviceModel) del device utilizzando un PDA.
2. Il Tecnico compila la scheda tecnica del device specificando il "Tipo anagrafica" con il flag "Nursery".
3. Il Tecnico richiede un test di funzionamento.
4. Il Sistema richiede la conferma dell'operazione.
5. Il Tecnico clicca il pulsante *OK*.

Il Tecnico ripete i passi 1-2-3-4 fino a quando non conferma l'operazione.

5. Il Sistema registra la scheda tecnica del device nell'anagrafica "Nursery".
6. Il Sistema notifica la registrazione all'Amministratore mediante la spedizione automatica di una email.
7. Il Sistema registra l'operazione in un repository di log contenente l'ID del Tecnico e la data-ora dell'operazione.

Estensioni:

*a. In qualsiasi momento il Sistema fallisce: si considera non valida l'alterazione dello stato del Sistema a fronte dell'operazione avvenuta. In particolare il Sistema esegue l'abort della transazione. Occorre, quindi, che:
    1. Il Tecnico si ri-autentichi e proceda con una nuova operazione.
    2. il Sistema registri l'errore in un apposito log.

1a. Il Tecnico desidera reinserire il codice, perché si è sbagliato o non desidera utilizzare quel codice per identificare il nuovo device:
    1. Il Sistema richiede un nuovo inserimento del codice (vai al punto 1).

6a. Il Sistema rileva un fallimento nella comunicazione del suo servizio di invio email.
    1a. Il Sistema ritenta l'invio e prosegue.
    1b. Il Sistema rileva che il servizio non riprende:
        1. Il Sistema segnala l'errore.
        2. L'Amministratore sceglie un altro mezzo di comunicazione per essere informato (es.

SMS sul cellulare).

Requisiti speciali:

- internazionalizzazione della descrizione del device (no brand).

Elenco delle varianti tecnologiche dei dati: /

Frequenza di ripetizione: /

Varie (Problemi aperti): /

1.3.3 Verifica funzionamento

CASO D'USO UC3: Verifica funzionamento

Portata: eCentral

Livello: sottofunzione

Pre-condizioni: il Tecnico ha effettuato la registrazione della scheda tecnica di un nuovo device nell'anagrafica "Nursery" dal campo mediante PDA.

Garanzia di successo (o Post-condizioni): il device riceve e trasmette i dati correttamente.

Scenario principale di successo:

1. Il Tecnico inizia il test di corretto funzionamento del device.
2. Il Sistema acquisisce i dati da periferiche mediante un opportuno protocollo di comunicazione.
3. Il Sistema visualizza i dati acquisiti dal device.
4. Il Tecnico confronta ciò che gli viene restituito dal Sistema con ciò che legge direttamente dal device.

Il Tecnico ripete 1-2-3-4 finché i dati non coincidono.

5. Il Tecnico richiede una stampa della pagina contenente tutte le informazioni appena inserite e gli ultimi dati letti (modulo di attivazione apparato).

Estensioni:

*a. In qualsiasi momento il Sistema fallisce: il Tecnico deve consultare il file di log per capire il malfunzionamento del Sistema.

Requisiti speciali:

- Il testo deve essere visibile da una distanza di 30 cm.
- Internazionalizzazione della lingua sul testo visualizzato.

Elenco delle varianti tecnologiche dei dati:

protocollo di comunicazione dipendente dal DeviceModel.

Frequenza di ripetizione: /

Varie (Problemi aperti): /

1.3.4 Stato rete (ultimo aggiornamento)

CASO D'USO UC4: Stato rete (ultimo aggiornamento)

Portata: eCentral

Livello: Obiettivo utente

Collaborazione: UC4 include UC5

Attori primari: Tecnico CPL, Utente (Cliente CPL), Amministratore, Gestore dei tecnici in campo (denominati di seguito Utente).

Parti interessate ed Interessi:

- Tecnico, Amministratore: vuole capire in modo chiaro ed immediato lo stato di un sistema di odorizzazione (su cui ha i diritti di accesso) in funzione dell'ultimo rilevamento effettuato sui device. In caso di mancanza dati può recuperare lo stato aggiornato ad una certa ora del giorno in modo estemporaneo sollecitato attraverso opportuni comandi.
- Gestore dei tecnici in campo: nel caso di problemi in più impianti, per cui è necessario organizzare qualche giro di intervento, il Gestore dei tecnici vuole disporre di una visualizzazione degli impianti su carta geografica dell'Italia o del mondo per capirne la disposizione e conoscere l'itinerario migliore per compiere più interventi possibili minimizzando il tempo di viaggio.

Pre-condizioni: l'Utente è identificato ed autenticato. Il device ha superato la taratura ed è censito in anagrafica.

Garanzia di successo (o Post-condizioni): visualizzazione dei dati significativi per stabilire se il target dell'indagine in esame è correttamente odorizzato, con possibilità di conoscere la dislocazione degli impianti costituenti.

Scenario principale di successo:

1. L'Utente seleziona la rappresentazione grafica strutturata.
2. Il Sistema visualizza:
   - un campo di filtraggio (degli elementi di composizione) e due pulsanti d'azione: *Nessun Filtro* e *Applica Filtro*;
   - una griglia di tre colonne indicanti:
     a. *Sistema di odorizzazione*: contiene il nome del sistema di odorizzazione ed un pulsante con label *Esplora*;
     b. *KPI*: contiene una o più icone di immediata comprensione (come ad esempio: LED VERDE/GIALLO/ROSSO o numeri) riassuntive dello stato di funzionamento tenendo conto di allarmi, anomalie o non invio dati per più di 24 ore ed altre elaborazioni;
     c. *Comandi*: per effettuare l'acquisizione estemporanea sollecitata dei dati.

Estensioni:

*a. In qualsiasi momento l'Utente può cambiare rappresentazione grafica:

1a*. L'Utente sceglie *Geolocalizzata* dal *top menu bar*:

1. Il Sistema visualizza una mappa geografica georeferenziata avvalendosi di Google Map.
2. Il Sistema mostra la dislocazione degli impianti, in funzione delle coordinate geografiche. Per ogni impianto è disponibile una label a scomparsa contenente il riassunto dei principali dati di funzionamento (ad esempio, TO e diagnostica come come cumulativo delle segnalazioni di stato del o dei device installati nell'impianto, la data di stop delle comunicazioni, ovvero la data di ultima comunicazione valida, etc.).
3. Il Sistema suggerisce l'itinerario migliore per compiere più interventi possibili

minimizzando il tempo di viaggio.

1b*. L'Utente sceglie *Sinottico* dal *top menu bar*:

1. Il Sistema visualizza la rete in termini di interconnessioni fisiche.

2a. L'Utente seleziona *Esplora*:

1. Il Sistema esplode la struttura di dettaglio del sistema di odorizzazione mostrandone i punti di rilievo di odorizzante (elementi di composizione).

2a. L'Utente seleziona l'elemento di composizione che desidera.

1. Il Sistema modifica il contesto.

2b. L'Utente non seleziona un elemento di composizione.

1. Il Sistema non modifica il contesto.

2b. L'Utente clicca sul campo di filtraggio:

1. Il Sistema permette di inserire l'informazione per eseguire il filtraggio.
2. L'Utente inserisce l'informazione.
3. L'Utente clicca il pulsante a lato per eseguire il filtraggio.
3. Il Sistema visualizza i risultati.

2c. L'Utente, attivando l'opportuno comando, sottomette al Sistema una richiesta di acquisizione misure (in modo estemporaneo sollecitato, non pianificato) per il sistema di odorizzazione selezionato al fine di recuperarne lo stato aggiornato ad una certa ora del giorno:

1a. Il Sistema aggiorna le KPI e produce un apposito log con l'esito dell'acquisizione ed i dati scaricati dai device facenti parte del sistema di odorizzazione.

1b. Nel caso si verificasse un errore, il Sistema:

1. visualizza data, ora e motivo del fallimento dell'acquisizione dati;
2. per ciascun device coinvolto nell'operazione memorizza in opportuno log il numero di tentativi eseguiti, aggiorna la statistica delle connessioni riuscite per mostrare l'affidabilità del device a chi lo prenderà in esame;
3. visualizza un opportuno suggerimento al problema riscontrato;
4. logga l'errore;
5. invia una notifica (mediante SMS, email, fax, chiamata vocale automatica) ai reperibili o ad un altro sistema SW (mediante FTP upload di dati in formato XML) con uno o più codici di errore. I reperibili, sulla base della notifica ricevuta, rimuovono la causa del problema sul campo;
6. invia una email all'Amministratore, in cui si specifica chi ha eseguito l'acquisizione dei dati e il motivo del fallimento.

Requisiti speciali:

- regole di accesso;
- regole di visibilità;
- tenere conto della disponibilità delle risorse di comunicazione (modem), che inficerà sul tempo di recupero misure dai device;
- le informazioni importate dovrebbero essere rese persistenti e mantenute on-line per il tempo stabilito;

- il tipo LED di KPI, che può assumere i colori:
  - verde: per indicare nessun allarme/segnalazione presente,
  - giallo: per indicare una segnalazione,
  - rosso: per indicare un allarme presente

  è calcolato come OR delle indicazioni provenienti da tutti i device degli impianti del sistema di odorizzazione.

Elenco delle varianti tecnologiche dei dati:

Specificare i DeviceModel ed i tipi di dato.

Frequenza di ripetizione: /

Varie (Problemi aperti):

Si suppone di dover chiudere in estate un impianto normalmente perché cala il consumo e, quindi, la portata. A causa delle derive termiche tra giorno e notte, le sonde di rilevazione del tasso di odorizzante possono rilevare un delta di livello di odorizzante inesistente. Se si utilizza l'impianto ad un regime di pressione molto bassa, come in estate, il valore di tale delta, normalmente trascurabile, diventa significativo, perché viene diviso per un valore di portata basso (< 1), producendo falsi positivi negli allarmi. Una possibile soluzione del problema descritto consiste nel disabilitare i device coinvolti per non utilizzare i loro dati per il calcolo delle KPI.

1.3.5 Visualizza carta d'identità

CASO D'USO UC5: Visualizza carta d'identità

Portata: eCentral

Livello: sottofunzione

Parti interessate ed Interessi:

- Utente: vuole essere sempre informato sulla natura di ciò che sta visualizzando (contesto).

Pre-condizioni: l'Utente è identificato ed autenticato.

Garanzia di successo (o Post-condizioni): facilita l'Utente nella comprensione del contesto.

Scenario principale di successo:

Il Sistema visualizza in modo dinamico le informazioni descrittive in merito a ciò che si sta visualizzando, cioè cambiano i nomi degli attributi che costituiscono la carta d'identità in funzione del contesto.

Estensioni: /

Requisiti speciali: /

Elenco delle varianti tecnologiche dei dati: /

Frequenza di ripetizione: /

Varie (Problemi aperti): /

1.3.6 Riassunto Diagnostica

CASO D'USO UC6: Riassunto Diagnostica

Portata: eCentral

Livello: sottofunzione

Collaborazione: UC6 extend UC4, UC6 extend UC8

Parti interessate ed Interessi:

- Utente: vuole essere sempre informato degli eventuali allarmi ed eventi con riferimento al contesto.

Pre-condizioni: l'Utente è identificato ed autenticato.

Garanzia di successo (o Post-condizioni): l'Utente conosce i problemi che si sono presentati e gli eventi del contesto in esame.

Scenario principale di successo:

Il Sistema visualizza in una porzione della schermata intitolata *Diagnostica*, con un link *More*, una sintesi delle segnalazioni rilevanti suddivise in due gruppi:

- *Ultimi allarmi*: per ogni elemento (allarme) viene espressamente indicato il device sorgente; gli elementi sono gli ultimi in ordine temporale (dal più recente) ed il loro numero varia da 10 a 15.
- *Eventi passati*: per ogni elemento viene espressamente indicato il device sorgente in ordine temporale; il numero degli elementi varia da 10 a 15.

Estensioni:

1a. L'Utente seleziona, senza cliccare, un elemento di uno dei due gruppi (allarmi o eventi passati):

1. Il Sistema visualizza informazioni di dettaglio in merito all'evento selezionato.

1b. L'Utente clicca su un evento di uno dei due gruppi:

1. Il Sistema conduce l'Utente alla schermata *Analisi dati*.

1c. L'Utente clicca sul link *More*:

1. Il Sistema conduce l'Utente alla schermata *Diagnostica di dettaglio*.

Requisiti speciali: /

Elenco delle varianti tecnologiche dei dati: /

Frequenza di ripetizione: /

Varie (Problemi aperti): /

1.3.7 Navigazione

CASO D'USO UC7: Navigazione

Portata: eCentral

Livello: sottofunzione

Collaborazione: UC7 extend UC4, UC7 extend UC8

Parti interessate ed Interessi:

- Utente: vuole navigare facilmente nella rete di distribuzione.

Pre-condizioni: l'Utente è identificato ed autenticato.

Garanzia di successo (o Post-condizioni): l'Utente si colloca in modo facilitato e rapidamente sull'oggetto che vuole indagare.

Scenario principale di successo:

1. Il Sistema visualizza in una lista i cinque impianti navigati più di frequente (ranking).
2. Il Sistema visualizza una mappa di rete in cui sono mostrati i nodi "parent", di un livello superiore al contesto, ed i nodi "child", di un livello inferiore al contesto (si ricorda che un figlio può avere più parent, quindi si tratta di una struttura a grafo).
3. Il Sistema visualizza un campo di ricerca rapida in cui inserire il nome del gruppo, del device e dell'impianto che si vuole ricercare.
4. Il Sistema visualizza un pulsante d'azione "ObjectBrowsing" per consentire di esplorare la rete in modo piatto all'Utente che non conosce il nome del target.

Estensioni:

1a. L'Utente clicca su un elemento della lista:

    1. Il Sistema cambia contesto.

2a. L'Utente seleziona senza cliccare la mappa:

    1. Il Sistema visualizza la mappa ingradita.

        1a. L'Utente seleziona un nodo:

            1. Il Sistema cambia contesto.

3a. L'Utente esegue la ricerca rapida:

    1. Il Sistema compie la ricerca e, contemporaneamente, mostra in una lista gli oggetti il cui nome contiene (dall'inizio) ciò che si sta componendo.
    2. L'Utente clicca su un elemento della lista.
    3. Il Sistema cambia contesto.

Requisiti speciali:

Elenco delle varianti tecnologiche dei dati: /

Frequenza di ripetizione: /

Varie (Problemi aperti): /

1.3.8 Stato impianto (ultimo rilevamento)

CASO D'USO UC8: Stato impianto (ultimo rilevamento)

Portata: eCentral

Livello: Obiettivo utente

Collaborazione: UC8 include UC5

Attori primari: Tecnico CPL, Utente (Cliente CPL), Amministratore. I tre attori sono indicati con l'unico nome di Utente.

Parti interessate ed Interessi:

- Utente: vuole visualizzare in modo chiaro e semplificato lo stato di odorizzazione di un impianto.

Pre-condizioni: l'Utente è identificato ed autenticato.

Garanzia di successo (o Post-condizioni): visualizzazione dei dati dello stato dell'impianto.

Scenario principale di successo:

Il Sistema provvede ad aggiornare la GUI ad istanti di tempo scanditi con una certa frequenza temporale.

1. Il Sistema visualizza un sommario dello stato corrente (ultimi dati) in forma tabellare, specificando:
   a. *Livello serbatoio*;
   b. *Delta livello serbatoio*;
   c. *TO impianto iniezione*;
   d. *Portata*;
   e. *Volume*;
   f. *Campo GSM*;
   g. *Livello di odorizzante al caricamento*;
   h. *Odorizzante caricato*;
   i. *Addetto al caricamento*.
2. Il Sistema visualizza informazioni recenti sugli impianti correlati (fisicamente interconnessi) in formato tabellare:
   a. *Impianto correlato*;
   b. *TO Edor (GRF) [mg/smc]*;
   c. *Temperature misura Edor*;
   d. *Tipo Odorizzante*.
3. Il Sistema visualizza i dati in un grafico. Il grafico presenta 3 curve di impianto e curve di TO pari al numero degli Edor dell'impianto in esame.

Estensioni: /

Requisiti speciali:

- internazionalizzazione delle unità di misura.
- valutazione della disponibilità delle risorse di comunicazione (modem).

Elenco delle varianti tecnologiche dei dati: /

Frequenza di ripetizione: /

Varie (Problemi aperti): /

1.3.9 Mostra allegati

CASO D'USO UC9: Mostra allegati

Portata: eCentral

Collaborazione: UC9 extend UC8

Livello: Obiettivo utente

Parti interessate ed Interessi:

- Utente: vuole poter accedere ai documenti associati all'impianto in esame.

Pre-condizioni: l'Utente è identificato ed autenticato.

Garanzia di successo (o Post-condizioni): visualizzazione dei documenti di rilievo relativi ad interventi

di manutenzione effettuati, certificati di calibrazione, certificati di analisi, altra documentazione di rilievo. Tali documenti sono file che possono essere in formato *.doc*, *.xls*, *.pdf*, *.jpeg*.

Scenario principale di successo:

1. Il Sistema visualizza in modo paginato una lista di righe composte da un'icona (graffetta) e dal nome del file del documento associato all'impianto.
2. L'Utente seleziona l'allegato di interesse.
3. Il Sistema apre il file e lo visualizza.

Estensioni:

Requisiti speciali:

Elenco delle varianti tecnologiche dei dati: /

Frequenza di ripetizione: /

Varie (Problemi aperti):

1.3.10 Analisi dati

CASO D'USO UC10: Analisi dati

Portata: eCentral

Livello: Obiettivo utente

Attore primario: Tecnico CPL, Utente (Cliente CPL), Amministratore. I tre attori sono indicati con l'unico nome di Utente.

Parti interessate ed Interessi:

- Utente: vuole ricercare in modo immediato, secondo certi criteri, i dati acquisiti (attività o segnali) di un periodo a scelta.

Pre-condizioni: l'Utente è identificato ed autenticato.

Garanzia di successo (o Post-condizioni): l'Utente ottiene i risultati della ricerca in modo rapido.

1. Il Sistema permette all'Utente di scegliere il tipo di analisi dati fra due possibili: *Complex* o *SimpleDIY* (Do It Yourself).
2. L'Utente seleziona *Complex*.
3. Il Sistema visualizza:
   a. due campi in cui inserire tramite un selettore automatico (per evitare errori di inserimento dell'input) la data di inizio e di fine analisi;
   b. due bottoni mutuamente esclusivi per selezionare:
      i. Rete (report di rete = **RPR**);
      ii. Tipo di Attività, specificabile tramite la selezione di un elemento tra quelli proposti in un menù a discesa (es. riempimenti = **RR**).
4. L'Utente seleziona **RPR** e manda in esecuzione la ricerca premendo il bottone *Search*.
5. Il Sistema mostra i risultati della ricerca in una tabella con le seguenti colonne:
   a. *Punto di rilievo*;
   b. *Volumi gas*;

c. *Odorizzante residuo*;
d. *Consumo odorizzante*;
e. *TO medio impianto iniezione*;
f. *TO medio Edor*;
g. *TO medio Delta livello*.

6. Il Sistema visualizza in un grafico i dati più significativi ed un link *More*.

Estensioni:

2a. L'Utente seleziona *SimpleDIY*:

1. Il Sistema visualizza due bottoni mutuamente esclusivi che consentono di attivare:
   i. il primo: due campi in cui inserire tramite un selettore automatico (per evitare errori di inserimento dell'input) la data di inizio e di fine analisi;
   ii. il secondo: consente di ottenere solo gli ultimi eventi (MAX dei tempi).
2. Il Sistema visualizza un menù a discesa prestabilito in cui si indicano insiemi di segnali.
3. Il Sistema visualizza dei flag di spunta:
   i. *Son-of*: per includere nei risultati gli impianti interconnessi fisicamente.
   ii. *Include activity*: per includere nei risultati le attività.

4a. L'Utente seleziona **RR** e manda in esecuzione la ricerca premendo il bottone *Search*.

1. Il Sistema mostra i risultati della ricerca in una tabella con le seguenti colonne:
   a. *Cabina*;
   b. *Volumi gas*;
   c. *Odorizzante caricato*;
   d. *Giacenza iniziale*;
   e. *Giacenza finale*;
   f. *Odorizzante utilizzato*;
   g. *TO medio*;
   h. *Periodo considerato*.

6a. L'Utente preme il pulsante *Print* per stampare un report:

1. Il Sistema procede a mandare in esecuzione il processo di stampa.

6b. L'Utente vuole confrontare il grafico in esame con un grafico analogo di un secondo target (o più) cliccando sul link *More*:

1. Il Sistema visualizza in una nuova pagina il grafico in esame e una lista di target selezionabili che si voglio confrontare.
2. L'Utente seleziona uno o più target.
3. Il Sistema visualizza gli andamenti dei nuovi grafici sovrapponendolo all'attuale con colori diversi per facilitare la distinzione a livello visivo.

6c. L'Utente clicca su *Export* per esportare il report in un determinato formato (*.xls*, *.pdf*, *.csv*).

1. Il Sistema visualizza in un menu a discesa i sistemi software di destinazione.
2. L'Utente seleziona il sistema software.
3. Il Sistema richiede conferma dell'operazione con *OK*, *Undo*.

4. L'Utente conferma con *OK*.
5. Il Sistema procede all'invio.

5a. Il Sistema rileva un errore nella comunicazione:

1a. Il Sistema ritenta l'invio e prosegue.

1b. Il Sistema rileva che il servizio non riprende:

1. Il Sistema segnala l'errore.

Requisiti speciali: /

Elenco delle varianti tecnologiche dei dati: /

Frequenza di ripetizione: /

Varie (Problemi aperti): /

Requisiti speciali: /

- regole di visibilità dei risultati.
- internazionalizzazione delle unità di misura.
- risposta delle ricerche entro 3 secondi il 90% delle volte.

Elenco delle varianti tecnologiche dei dati: /

Frequenza di ripetizione: /

Varie (Problemi aperti):

1.3.11 Programma device

CASO D'USO UC11: Programma device

Portata: eCentral

Livello: Obiettivo utente

Attori primari: Amministratore, Tecnico (identificati nel seguito come Utente)

Parti interessate ed Interessi:

- Amministratore: vuole apportare modifiche ai parametri di monitoraggio del device (l'intervallo di tempo per la registrazione dei valori che verranno caricati in eCentral dal modulo di acquisizione dati, orario d'inizio rilevazione) o inviare telecomandi per modificare parametri di funzionamento (es. chiusura valvola).
- Tecnico: vuole aggiornare il firmware di device da remoto, talvolta dopo opportuno check della password del device (solo alcuni device lo richiedono).

Pre-condizioni: l'Utente è identificato ed autenticato.

Garanzia di successo (o Post-condizioni): il device funziona correttamente dopo la configurazione compiuta da remoto.

Scenario principale di successo:

1. Il Sistema permette di selezionare il target della programmazione con due pulsanti mutuamente esclusivi: *Group of device* e *Single device*.
2. L'Utente seleziona *Group of device*.
3. Il Sistema permette di selezionare tra: *Modalità operativa* (Comandi) e *Configurazione* (Firmware).

4. L’Utente seleziona *Modalità operativa*.
5. Il Sistema visualizza l’ultimo stato di *Modalità operativa* (es. stato valvole, etc.).
6. Il Sistema visualizza i comandi che si possono inviare.
7. L’Utente seleziona i comandi con possibilità di scelta multipla.
8. L’Utente seleziona *Send*.
9. Il Sistema richiede conferma dell’invio con *OK, Undo*.
10. L’Utente conferma con *OK*.
11. Il Sistema invia i comandi.

L’Utente ripete i passi da 1 a 11.

12. Il Sistema verifica, tramite telelettura, se sono state apportate le modifiche di configurazione al device/gruppo di device.
13. Il Sistema visualizza l’esito del test.

L’Utente ripete i passi da 1 a 11 in caso di errore.

14. Il Sistema registra le modifiche di configurazione apportate al gruppo o al singolo device.

Estensioni:

2a. L’Utente seleziona *Single Device*.

4a. L’Utente seleziona *Configurazione*.

5a. Il Sistema visualizza l’ultimo stato di *Configurazione*.

Requisiti speciali:

- La programmazione pianificata deve tener conto della disponibilità delle risorse di comunicazione (modem), dunque sarà opportuno classificare le attività in “rimandabili” e “non rimandabili”.

Elenco delle varianti tecnologiche dei dati: /

Frequenza di ripetizione: /

Varie (Problemi aperti):

Il Sistema potrebbe doversi autenticare per accedere al device, fornendo una password.

2 PROGETTO ARCHITETTURALE

Il sistema software eCentral, utilizzando interfacce utente grafiche (GUI), è guidato dagli eventi (*event-driven*) dinamicamente generati dalle richieste dell'utente, per cui viene eseguito in modo imprevedibile e causale. Attraverso un approccio di tipo *object-oriented* è stato possibile associare la gestione di ogni evento ad uno specifico oggetto software nello stato corrente d'esecuzione del programma. Si è deciso di utilizzare tale approccio anche perché offre i seguenti vantaggi:

– fornisce astrazione, incapsulamento, riuso, ereditarietà, scambio di messaggi, polimorfismo;
– fornisce riutilizzo del codice, permette un minor tempo di sviluppo ed una maggior qualità del SW;
– si presta facilmente ad un processo iterativo ed incrementale

Nella fase di progetto si è tenuto conto della complessità della trasformazione dei risultati di analisi e di progetto, poiché la semantica del modello relazionale, utilizzata nella successiva fase di implementazione, è piuttosto povera rispetto a quella dei modelli a oggetti.

Il sistema *object–oriented* eCentral è stato progettato per favorire l'integrazione e ciascun modulo è stato creato indipendente dagli altri per quanto possibile. Le dipendenze fra moduli sono state identificate e minimizzate nelle fasi di analisi e progetto, nel tentativo ideale di far corrispondere a ciascun modulo un processo eseguito in risposta ad una particolare necessità dell'utente.

La complessità del progetto ha suggerito una sua suddivisione in "piccoli" moduli. Una volta noto il target da ottenere si è utilizzato l'approccio *top-down* per:

– individuare i componenti (chi) a cui assegnare specifiche funzioni (cosa fa) per raggiungere il target, prestando molta attenzione ad evitare sovrapposizioni e repliche;
– individuare il formato dei dati scambiati dai componenti, evitando cicli di dipendenza funzionale;
– esplodere i dettagli dei componenti.

Tale approccio è stato svolto in modo congiunto con quello *bottom-up*, perché nella fase di progettazione ci si può dimenticare di qualche particolare degli use case.

In sintesi il problema, di tipo complesso, è stato affrontando procedendo con:

– la definizione di un'architettura di riferimento,
– la definizione di massima dei componenti funzionali,
– la definizione delle interfacce di comunicazione fra i componenti,
– la progettazione ed implementazione dei singoli componenti.

In tale fase di progettazione è corretto tenere conto della piattaforma HW/SW sulla quale il sistema software eCentral deve essere implementato. Sono state, inoltre, decise le strategie di soluzione relative agli aspetti client e server del sistema:

1. interfaccia utente (client);
2. database (server);
3. programmi necessari per far interagire le parti 1 e 2 (middleware).

I modelli client-server sono stati estesi per permettere la realizzazione di un'architettura fisica su 2 tier ed una applicativa di 3 strati [1, 3].

2.1 Progettazione dell'architettura hardware e dell'infrastruttura di rete di eCentral

L'architettura hardware dell'applicazione eCentral è situata negli edifici della multiutility CPL, che assume il ruolo di fornitore esterno di servizi per i propri customers e di gestore di tale infrastruttura fisica. Tale scenario di installazione è detto *hosting*.

Il modello di interazione del sistema software è di tipo client-server. L'applicazione si basa sul Web, che ne migliora l'interoperabilità, poiché aumenta la possibilità di raggiungere vari tipi di client (a patto che utilizzino browser compatibili), indipendentemente dalla piattaforma del client stesso.

Il sistema software eCentral presenta un'architettura strutturata su tre livelli applicativi.

1. **Layer di presentazione (o presentation logic)**, che si occupa di gestire la logica di presentazione, cioè le modalità di interazione con l'utente, e contiene le modalità di interfacciamento grafico e le modalità di rendering delle informazioni. Traduce, infine, funzionalità e dati in formato comprensibile all'utente.
2. **Layer di logica applicativa (o business logic)**, che si occupa delle funzioni da rendere disponibili all'utente. Coordina l'applicazione, i comandi ai processi ed esegue i task applicativi.
3. **Layer di accesso ai dati**, che si occupa delle gestione dell'informazione, con accesso alla basi di dati ed a sistemi esterni.

I tre layer applicativi descritti possono essere installati su vari livelli fisici, detti *tier*. Un *tier* fisico rappresenta una macchina con differente capacità di elaborazione. È stata adottata l'architettura fisica "two tiered", nella quale i tre layer applicativi sono suddivisi in due *tier*:

1. una prima macchina server ospita il server Web e la business logic, affinché il client sia "leggero" (strato 1);
2. una seconda macchina server ospita il layer di accesso ai dati (strato 2) [1, 4].

Il sistema software eCentral, per controllare i device da remoto, integra il servizio ECD (vedi paragrafo 2.1.2.1) che è un servizio MS Windows mandato in esecuzione su una macchina distinta. Poiché una singola macchina può ospitare 8 ÷ 16 porte seriali, il numero di device che si possono controllare è alquanto ridotto. Il problema è risolto mediante l'utilizzo di particolari *bridge seriali Ethernet* (denominati Moxa) che, tramite un apposito client in esecuzione sulla macchina, permettono di virtualizzare le porte seriali. Un driver in esecuzione sulla Moxa, contattata over IP, procede a mappare le porte virtuali in quelle fisiche. Le Moxa possono essere replicate per garantire la scalabilità desiderata al sistema. Se la capacità di elaborazione del sistema dovesse comunque risultare insufficiente a sostenere le comunicazioni con un numero crescente di device, è possibile scalare l'ECD su '*n*' computer. L'ECD elabora le comunicazioni in entrambe le direzioni, sia dai device ad eCentral che viceversa, e le rende persistenti in locale su di un server RDBMS di back-end. Quest'ultimo server è protetto da un firewall per garantire che i dati siano accessibili solo agli utenti legittimi (in funzione dei permessi attribuiti) e per impedire che i soggetti non autorizzati ne possano alterare il valore. La topologia di rete device-ECD è

quella a stella, con l'ECD al suo centro. Il browser è il responsabile del controllo dell'informazione visualizzata sullo schermo dell'utente e della gestione degli eventi prodotti dall'utente stesso.
L'architettura di elaborazione è centralizzata, poiché il client ha accesso ad un solo server RDBMS alla volta, ovvero in ogni singola richiesta non è possibile combinare dati di due o più database server.
L'ECD può rappresentare un punto vulnerabile del sistema poiché, acquisendo il controllo di un device, è possibile utilizzarlo per compiere azioni malevoli. Il problema di sicurezza emerso è facilmente risolvibile, attribuendo ai device opportuni GRANT di accesso al RDBMS. Tramite il device compromesso si possono al più inserire dati spuri, ma non cancellare o aggiornare tabelle "sensibili". È, comunque, possibile anche utilizzare il sistema di autenticazione applicato alle soluzioni di Mobile Office realizzate con APN. In questo modo si effettua il controllo dell'accesso tramite SIM Card, garantendo la difesa della Intranet da eventuali tentativi di accesso da parte di utenti non autorizzati. Un'APN richiede, però, accesso via GPRS.
Attualmente alcuni device veicolano i propri dati su canali di comunicazioni insicuri, come GSM, per cui possono essere intercettati ed alterati. Occorre prestare la massima attenzione agli attacchi provenienti dal Web, poiché attraverso tale lato si possono commettere azioni molto gravi, come disattivare un device o cambiarne la configurazione (chiusura di una valvola). Per questo motivo un utente deve sempre autenticarsi al sistema preventivamente. Le credenziali di autenticazione fornite nella pagina di login possono essere trasmesse in chiaro HTTP o crittografate con HTTPS. L'utilizzo di HTTPS può essere esteso anche alla LAN, non solo ad Internet. Una volta effettuata l'autenticazione, si può utilizzare il protocollo HTTP per compiere le azioni di interesse, perché solo le credenziali di accesso devono essere protette. Nella figura 2.1 viene mostrata l'infrastruttura di rete di eCentral.

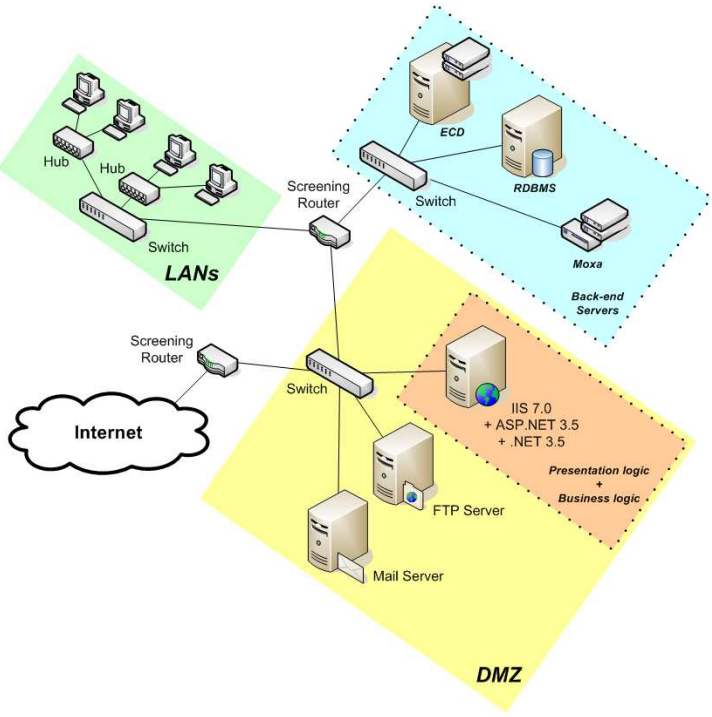


Figura 2.1: infrastruttura di rete di eCentral

La LAN di processo può essere ritenuta un ambiente sicuro: gli accessi fisici al supporto Ethernet sono regolamentati dai gestori dell'infrastruttura, e l'accesso al wireless è controllato per mezzo di protocolli ad-hoc quali WPA/WPA 2 (o in subordine WEP) e con la naturale limitazione della propagazione fisica (dislocazione del sito).
Tale configurazione può essere valutata alla luce degli obiettivi e dei vincoli di progetto dell'architettura.

2.1.1 Vincoli di progetto dell'architettura HW

2.1.1.1 Prestazioni

La separazione fisica, su macchine distinte, del RDBMS (strato 2) dal server Web (strato 1) permette un dimensionamento più adeguato delle macchine coinvolte. Occorre ricordare che l'interprete degli script (*ASP.NET 3.5 + Framework .NET 3.5*) e il RDBMS (*SQL Server 2005*) sono entrambi delle applicazioni a consumo intensivo di memoria e CPU, per cui, se allocate su una stessa macchina con uso condiviso delle sue risorse, potrebbero entrare in conflitto, dando luogo a possibili colli di bottiglia. L'host dedicato all'accesso ai dati può essere potenziato con dischi ad accesso veloce e replicati (almeno *RAID 5*). L'incremento di prestazioni, introdotto dalla macchina dedicata a tale accesso, può essere in parte ridotto dal sovraccarico di trasmissione su rete derivante dalla separazione delle macchine; tale diminuzione è, però, più che compensata dall'aumento di prestazioni derivante dalla separazione, perché i collegamenti sono realizzati su cavi *Ethernet gigabit* (*1000Base-SX*).

2.1.1.2 Scalabilità

La scalabilità è garantita dalla possibilità di operare separatamente su entrambi gli strati. In genere il primo collo di bottiglia risiede nello strato 1, perché il server Web e il motore di esecuzione dei programmi saturano prima di riuscire a sfruttare pienamente la capacità dello strato 2.

2.1.1.3 Disponibilità

Anche se i guasti in tali strati sono isolati, la disponibilità è comunque indebolita dal fatto che ciascun componente costituisce un singolo punto di guasto. Tale limite può essere superato dall'adozione di tecniche di replicazione che non solo migliorano le prestazioni e la scalabilità, ma anche la disponibilità, grazie all'introduzione di risorse ridondanti. Per motivi di carattere economico nell'architettura fisica progettata la replicazione riguarda:

- il server Web con una sua sola "copia fredda";
- la base di dati con modalità di replicazione asimmetrica.

Data la modesta quantità di richieste, non si fa uso di *caching proxy*, per cui non si deve garantire la consistenza delle informazioni.

2.1.3.4 Sicurezza

La configurazione mostrata in figura 2.1 si caratterizza per l'utilizzo di due router a filtraggio di pacchetti, che costituiscono due livelli distinti di difesa. Il router di filtraggio di pacchetti posto sul confine tra *Internet* e la *DMZ* compie rapidamente un primo filtraggio di tipo grossolano dei pacchetti (può implementare ad esempio il filtraggio *Stateful Packet Filtering*) secondo la politica di accesso ai servizi stabilita per la rete. Se il primo router a filtraggio di pacchetti venisse completamente compromesso, il traffico da Internet agli host sulla rete interna non potrebbe attraversare il router compromesso senza subire alcun controllo, perché il secondo firewall (*screening router*) previene questo tipo di violazione di sicurezza. Chi vuole effettuare un'intrusione deve, pertanto, penetrare in entrambi i router a filtraggio di pacchetti prima di compromettere la sicurezza della rete interna. Il secondo firewall può bloccare completamente le richieste HTTP, lasciando passare solo le richieste indirizzate al RDBMS dell'area "Back-end servers" e rendendo così più difficile ai malintenzionati l'accesso ai dati. Per poter raggiungere dall'esterno i servizi pubblici (oltre che ovviamente dall'interno), i corrispondenti server devono essere posizionati all'interno della DMZ. Questo tipo di configurazione, in sintesi, offre i seguenti vantaggi:

- prevede due livelli di difesa per contrastare le intrusioni;
- il router esterno segnala alla rete Internet solo la presenza della DMZ; la rete interna non è, quindi, visibile da Internet;
- analogamente il router interno segnala alla rete interna solo l'esistenza della DMZ; questo implica che i sistemi posti sulla rete interna non possono stabilire connessioni dirette a Internet.

Riassumendo:

- dalla rete interna -

1. il traffico originato dalla rete interna verso Internet è permesso tutto o in parte;
2. il traffico originato dalla rete interna verso la DMZ è permesso in funzione dei protocolli stabiliti.

- da Internet -

1. il traffico originato da Internet verso la DMZ è permesso in funzione dei protocolli stabiliti;
2. il traffico originato da Internet verso la rete interna è tutto vietato (permesse solo le risposte per le richieste della rete interna).

- dalla DMZ -

1. il traffico originato dalla DMZ verso Internet è permesso in funzione dei protocolli stabiliti;
2. sono permesse le richieste nel protocollo proprietario indirizzate al RDBMS originate dalla DMZ [4, 5, 6].

2.1.2 Interazione lato client (JavaScript)

Gli script ASP.NET risolvono il problema della gestione dei form e delle interazioni con il DBMS, ma non possono rispondere ai movimenti del mouse o interagire direttamente con gli utenti. A tale scopo, è necessario disporre di script incorporati nelle pagine HTML, che sono eseguiti dalla macchina client anzichè da quella server. Un browser, che interpreta una funzione *Javascript*, non deve necessariamente contattare il server per inviare una pagina, che, pertanto, viene elaborata in locale e visualizzata quasi immediatamente.

2.1.3 Descrizione e configurazione del first tier

2.1.3.1 Internet Information Services (IIS) 7.0

IIS 7.0, distribuito con Windows Server® 2008 (e con Windows Vista®), può essere considerato il centro della piattaforma applicativa intorno alla quale Microsoft ha costruito una serie di servizi. Esso vanta una forte integrazione con il Framework .NET, soprattutto nelle sue versioni 3.0 e 3.5 [7].

2.1.3.1.1 Componenti di IIS 7.0

L'architettura di IIS 7.0 supportata da Windows Server® 2008 (e da Windows Vista®) è caratterizzata da diversi componenti, che svolgono importanti funzioni per l'applicazione eCentral: mettersi in ascolto di richieste, gestire i processi e leggere i file di configurazione. Questi componenti sono:

1. i "protocol listeners", come ad esempio *HTTP.sys*;
2. i servizi, come *World Wide Web Publishing Service* (WWW service) e il *Windows Process Activation* Service (WAS).

2.1.3.1.1.1 Protocol Listeners

I **protocol listeners** ricevono le richieste in uno specifico protocollo, le inviano a IIS per l'elaborazione e, poi, restituiscono le risposte ai richiedenti. IIS 7.0 utilizza *HTTP.sys* come il protocol listener sia per le richieste HTTP sia per quelle HTTPS.

2.1.3.1.1.1 Hypertext Transfer Protocol Stack (HTTP.sys e TCP.sys)

L'HTTP listener è un driver kernel del sistema operativo di Windows; dall'utilizzo di *HTTP.sys* si traggono i seguenti benefici:

- **kernel-mode caching:** le richieste sono servite senza dover compiere lo switch in *user mode*;
- **kernel-mode request queuing:** le richieste causano meno overhead nel context switching, poiché il kernel le inoltra direttamente al corretto processo di lavoro W3WP.exe di IIS. Se nessun processo di lavoro è disponibile per accettare una richiesta, essa viene salvata in una coda kernel-mode finché non viene presa incarico;
- **request pre-processing e security filtering:** le richieste subiscono un preliminare filtraggio ed una pre-elaborazione.

ISS 7.0 oltre all'*HTTP.sys* dispone di un driver in modalità kernel, denominato *TCP.sys*, per supportare più protocolli come quelli TCP nativi.

2.1.3.1.1.2 World Wide Web Publishing Service (WWW service)

In IIS 7.0 le funzionalità prima gestite dal solo *World Wide Web Publishing Service* (WWW Service) sono, ora, suddivise in due servizi: *WWW Service* (**W3SVC**) e il nuovo servizio *Windows Process Activation Service* (**WAS**). I due servizi sono in esecuzione nello stesso processo *Svchost.exe* con nome utente *LocalSystem*.

2.1.3.1.1.2.1 W3SVC

In IIS 7.0 i processi di lavoro W3WP.exe non sono più gestiti dal WWW Service, che ora opera da *listener adapter* per l'*HTTP listener* (*HTTP.sys*). Il WWW Service è, pertanto, responsabile:

– della configurazione dell'*HTTP.sys* con gli URL su cui l'*HTTP.sys* deve porsi in ascolto;
– dell'aggiornamento dell'*HTTP.sys*;
– dell'invio delle notifiche al *WAS* quando una richiesta entra in coda.

2.1.3.1.1.2.2 WAS

Alla richiesta di un URL da parte di un client, il WAS, al termine della lettura del file *applicationhost.config*, genera un processo di lavoro IIS (W3WP.exe), che esegue il codice necessario a spedire una risposta. Il principale compito del WAS è la gestione degli *application pool*; dopo aver generato un W3WP.exe, infatti, ne controlla lo stato di salute (health), riavvia automaticamente i processi W3WP.exe caduti in errore e garantisce che non siano consumate più risorse di quanto specificato nel corrispondente *AppPool configuration*. Il WAS è reso indipendente dai processi W3WP.exe per permettere ai suoi servizi di essere sfruttati anche al di fuori dell'ambito HTTP. Il WAS detiene anche dati di run-time e di stato.

2.1.3.1.2 Il Process Model di IIS 7.0

Per eseguire in modo sicuro ed affidabile più website ed applicazioni Web o Web-service su una sola macchina, occorre utilizzare un *process model* che, in IIS 7.0, è fornito dal *Windows Process Activation Service (WAS)*.

2.1.3.1.2.1 Application pool

Uno dei requisiti più importanti delle applicazioni del Framework .NET è l'affidabilità. Per creare applicazioni affidabili, che devono eseguire per un lungo periodo di tempo, non è consigliabile costruire un'architettura in cui le applicazioni vengono eseguite all'interno del processo server. Se, infatti, molte risorse sono condivise a livello di processo, un errore nell'esecuzione del codice dell'applicazione determina il crash dell'intero processo server.
Fino alla versione 3 di IIS tutti i siti web ospitati e la stessa piattaforma giravano su di un unico processo chiamato **inetinfo.exe**.
Con la versione 4 di IIS si era iniziato a pensare per la prima volta ad un modello in cui la piattaforma era completamente slegata dai servizi web, per cui un processo di nome **inetinfo.exe** gestiva il WWW e tanti siti web che potevano essere eseguiti dalla stessa macchina, con il vantaggio di poter essere avviati e fermati a piacimento senza per questo impattare le altre applicazioni. Nessun sito web, però, poteva interagire con gli altri ed in fase di debug, tutto veniva spostato sotto il processo principale. Il primo bug, che creava un errore tale da mandare in crisi il processo, obbligava a riavviare il servizio. Tutti gli altri siti venivano spenti e riaccesi, e si perdevano tutte le sessioni di lavoro.
Con la versione 5 di IIS, sono state apportate piccole modifiche al server web e al suo modo di gestione denominato "pooled process", che consentiva di far girare diverse applicazioni in un processo separato dal core della piattaforma chiamato **dllhost.exe**. Era possibile riavviare, fermare e debuggare le applicazioni in maniera totalmente indipendente ed attribuire, inoltre, all'applicazione un profilo di isolamento del processo

(a scelta tra basso, medio e alto), al quale era assegnata un'area di memoria condivisa, in cui poter lavorare. Lo svantaggio rimaneva essere quello di dover eseguire tutte le applicazioni sotto un unico processo [8].

A partire dalla versione 6.0 l'unità di gestione delle applicazioni è denominata *application pool*. Per ogni applicazione web si specifica l'*application pool* di appartenenza, che richiede di settare un'unica versione del Framework .NET. L'*application pool* per eCentral viene mostrato nelle figure 2.2 e 2.3:

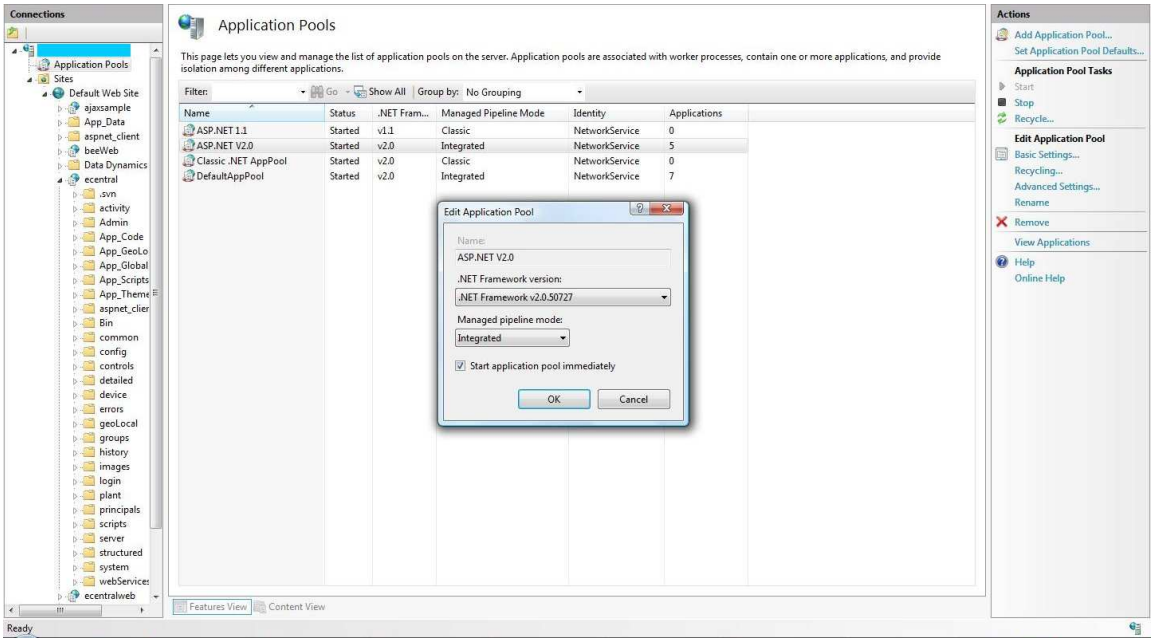


Figura 2.2: Application pool

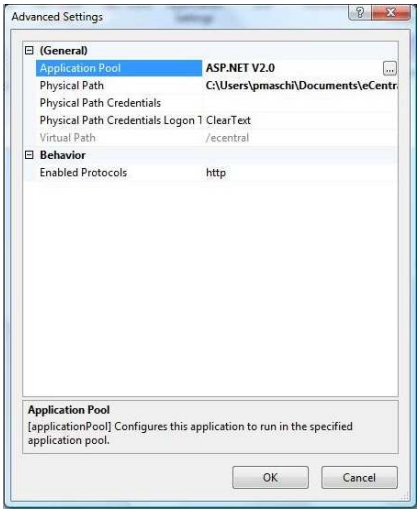


Figura 2.3: Advanced Settings

Se non fosse garantita questa corrispondenza biunivoca tra pool e versione del Framework, l'esecuzione della prima applicazione, che richiede un Framework differente da quello settato, genererebbe un errore.
Un *application pool* è, pertanto, un insieme di applicazioni web, che condividono alcune impostazioni. Da tali pool attingono i processi W3WP.exe per eseguire le applicazioni web (sul medesimo server), delimitandone i confini e prevenendone mutue dipendenze. Come nel seguito mostrato in figura 2.4, per ogni *application pool* il WAS avvia almeno un processo di lavoro W3WP.exe, in cui viene eseguito un particolare tipo di applicazione.

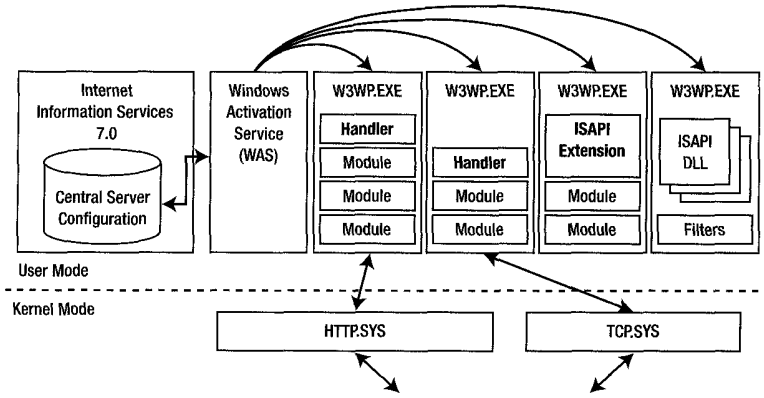


Figura 2.4: W3WP.exe

Se un'applicazione di un pool viene eseguita da più di un processo di lavoro W3WP.exe, si parla di *Web Garden*. Le *ISAPI* (*Internet Server Application Programming Interface*) mostrare in figura 2.4 permettono di integrare delle funzionalità ad IIS, sostituendo o estendendo quelle già presenti. Con un'*ISAPI* è possibile la creazione di due differenti tipologie di moduli: i *filtri* e le *estensioni*. I *filtri* sono i componenti che:

- gestiscono ogni richiesta in arrivo ad IIS per cambiarne il contenuto;
- modificano il comportamento di alcune funzionalità del server Web, come l'autenticazione.

Le *estensioni* si inseriscono tra il server Web ed il motore d'esecuzione degli script per intercettare e gestire le richieste e le risposte che transitano nel server Web.
Il riciclaggio di un pool avviene in modo estremamente efficiente. Il WAS genera un nuovo processo di lavoro W3WP.exe in parallelo al vecchio che sta ancora gestendo le richieste. Quando è attivo, il nuovo processo di lavoro prende le richieste dalla rispettiva coda dell'*HTTP.sys*, mentre quello vecchio si ferma a prelevarne di nuove. Viene eseguito lo shut down del vecchio processo, che ha completato l'elaborazione di tutte le richieste. Per un dato *application pool*, l'*HTTP.sys* accoda, di default, fino ad un massimo di 1'000 richieste, prima di iniziare a rifiutarle.
In IIS 7.0 si continua ad utilizzare il concetto di *application pool* introdotto con l'*IIS 6.0 worker process isolation mode*, per cui le applicazioni ASP.NET sono eseguite da uno o più processi W3WP.exe.
L'elemento *managedPipelineMode* della sezione <*applicatioPools*> del file *ApplicationHost.config* può valere:

– **integrated:** l' "application pool" funziona solo con applicazioni ASP.NET 2.0;

– **classic:** l' "application pool" funziona solo con applicazioni ASP.NET 1.1 e 2.0, che non supportano la modalità integrata.

Come mostrato nella figura 2.5 per default un'applicazione installata nel *Default Web Site* è configurata per essere eseguita nell'*application pool* denominato *DefaultAppPool*, che ha i permessi di un servizio di rete.

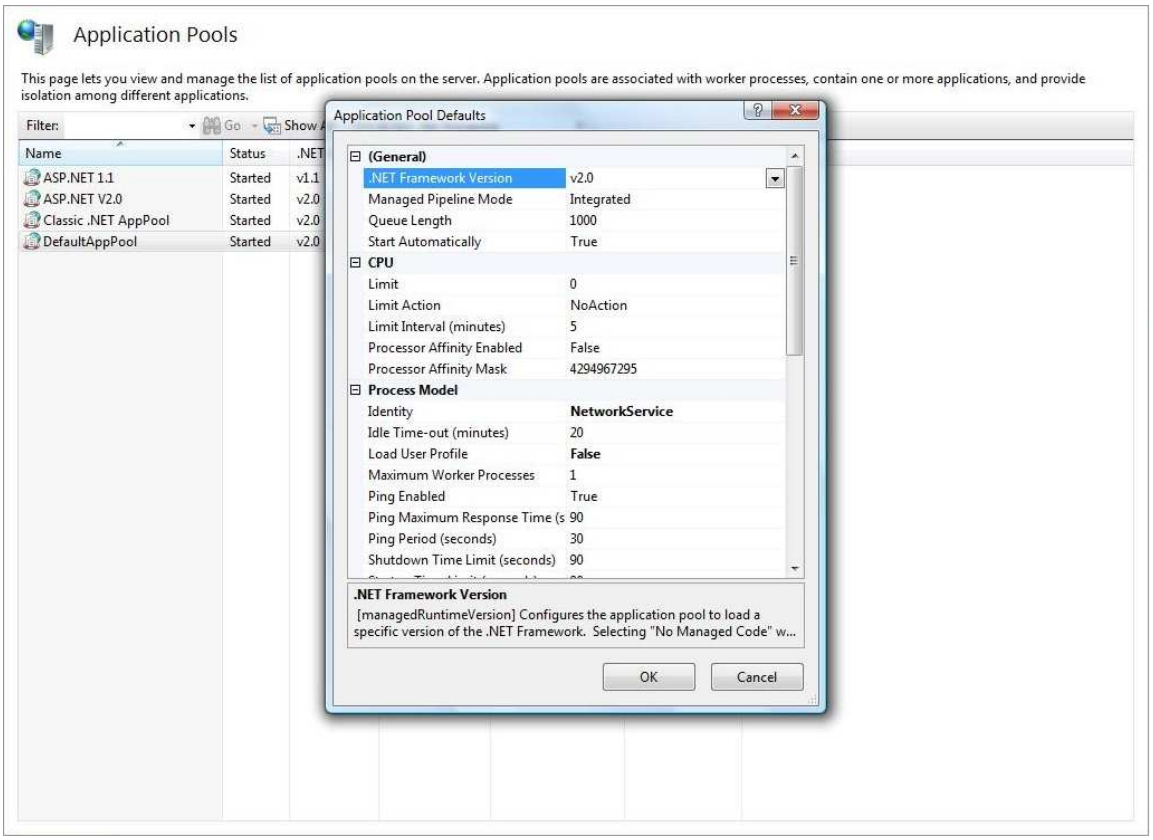


Figura 2.5: DefaultAppPool

Di un *application pool* si possono settare diversi parametri come: *Enable Pinging*, *Rapid–Fail protection*, *Startup time limit*, *Shutdown time limit*, *Idle timeout*, *Request queue limit* (limitare la coda di richiste a livello kernel), *Enable CPU monitoring*, *Web Garden* (massimo numero di processi di lavoro), *SMPProcessor–AffinityMask*, *Recycle in minutes*, *Recycle in number of requests*, etc.

È possibile configurare un account distinto per ogni *application pool* eseguito da uno o più processi W3WP.exe, garantendo un isolamento aggiuntivo attraverso i permessi dell'account assegnato al/ai processo/i di lavoro W3WP.exe. È possibile, pertanto, configurare ed assegnare ad ogni applicazione Web, avente permessi speciali di sicurezza, un *application pool* ad hoc affinchè venga eseguito sotto un account Windows creato con tali permessi. Tale account, però, deve avere almeno i permessi di *Network Service*. Solo le applicazioni che richiedono tali permessi (per garantire l'uniformità di impostazioni delle applicazioni del

pool) saranno inserite in quell'*application pool*. L'account Windows potrebbe, ad esempio, essere aggiunto agli utenti di SQL Server, per permettere solo alle applicazioni configurate con quell'*application pool* di accedere al RDBMS, per non dover salvare username e password nel file *Web.config*. Gli account predefiniti con privilegi diversi sono: *Network Service*, *Local Service*, *Local System*. I permessi di servizio di rete (*Network Service*) sono posseduti da un gruppo Windows, che viene installato con IIS 7.0, denominato *IIS_IUSRS*. Ogni account, che s'intende utilizzare come *application pool*, deve essere un membro di tale gruppo, come l'account *Network Service*. I permessi concessi dall'appartenenza a tale gruppo consistono nella possibilità data all'account:

- di essere eseguito come un servizio in background;
- di accedere alle directory necessarie come ai file temporanei ASP.NET salvati, in cui ASP.NET memorizza le versioni compilate dinamicamente di differenti pagine.

In particolare, se viene settata l'autenticazione anonima, il processo di lavoro W3WP.exe viene eseguito sotto un account (o utente) denominato *IUSR* avente i permessi di un *Local Service*. Occorre creare un nuovo account (di cui si settano manualmente username e password), affinchè un utente anonimo possa avere i diritti sulla rete (*Network service*). Riassumendo IIS 7.0 offre:

- il built–in account *IUSR*, che sostituisce l'account *IUSR_MachineName*;
- il built–in group *IIS_IUSRS*, che sostituisce il gruppo *IIS_WPG*.

Tali account sostitutivi hanno un unico SID (security identifier) e non sono localizzati, cioè mantengono il loro nome indipendentemente dalla versione di Windows installata.

Per consentire l'accesso ad eCentral, nell'Internet Information Services (IIS) Manager si sono compiute le seguenti operazioni:

1. si è cliccato "Edit Permissions…" dopo aver selezionato l'applicazione web *ecentral*;
2. si è cliccato "Edit…" per modificare i permessi:
3. si è cliccato "Add…" per aggiungere il tipo di built–in account;
4. si è cliccato "Locations…" per specificare la macchina locale e non il dominio di rete locale;
5. si sono cliccati in successione "Advanced…" e "Find Now" per visualizzare gli account disponibili;
6. si è scelto l'account membro del gruppo *IIS_IUSRS*;
7. si sono cliccati, infine, "Apply" ed "OK", per dare conferma.

La figura 2.6, mostrata nel seguito, consente di notare il nuovo account aggiunto *IIS_IUSRS* con cui viene eseguita l'applicazione web *ecentral*:

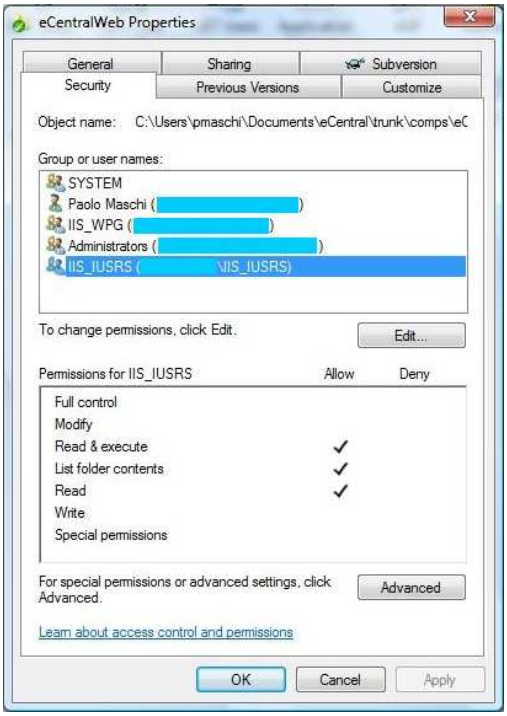


Figura 2.6: eCentralWeb Properties

Per rendere effettive le modifiche è stato necessario riavviare il server Web, cliccando sul pulsante *Restart* dopo aver selezionato il *Default Web Site*.

Come mostrato in figura 2.6 l'account dell'*application pool* di IIS deve possedere almeno i diritti in lettura sulle cartelle della *directory virtuale eCentral*, perché in caso contrario il server Web IIS 7.0 segnalerà un errore di permessi già alla prima richiesta.

2.1.3.1.2.1.1 processModel Element per l'applicationPoolDefaults in IIS 7.0

È stato possibile configurare l'*application pool* ASP.NET V2.0 del processo di lavoro W3WP.exe per eCentral attraverso l'*Internet Information Services Manager (IIS) Manager*, come mostrato nella figura 2.7:

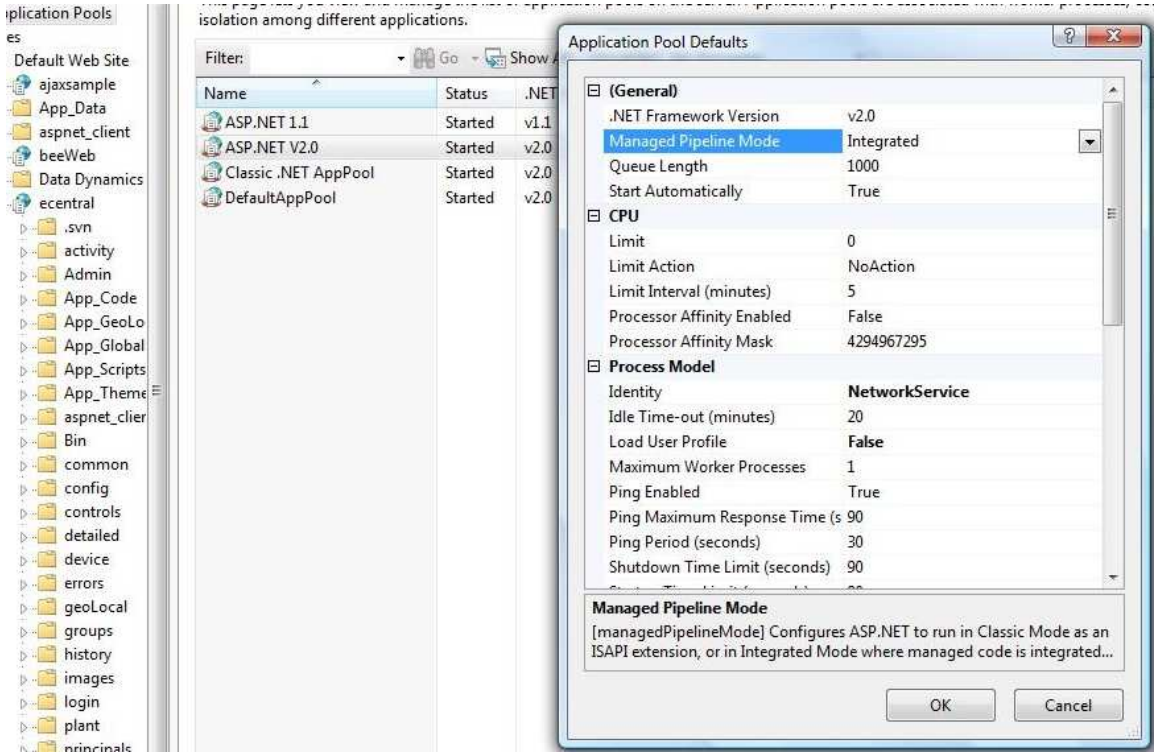


Figura 2.7: Application Pool Defaults

2.1.3.1.3 Diagramma architetturale

Nel diagramma di figura 2.8 vengono mostrati i componenti che costituiscono l'architettura di IIS 7.0 e precisamente:

- il file root *applicationhost.config* in cui è salvata la configurazione di IIS 7.0;
- *HTTP.sys*;
- uno o più processi di lavoro W3WP.exe che ospitano codice di qualunque tipo (ad esempio: codice di pagine ASP e ASP.NET, di moduli, di filtri ed estensioni ISAPI, etc.). In base alla modalità con cui l'amministratore di IIS decide di isolare i suoi website e le applicazioni Web, il numero di questi processi di lavoro può diventare elevato. Un W3WP.exe recupera le richieste pendenti dalla corrispondente coda del *HTTP.sys*;
- il WAS.

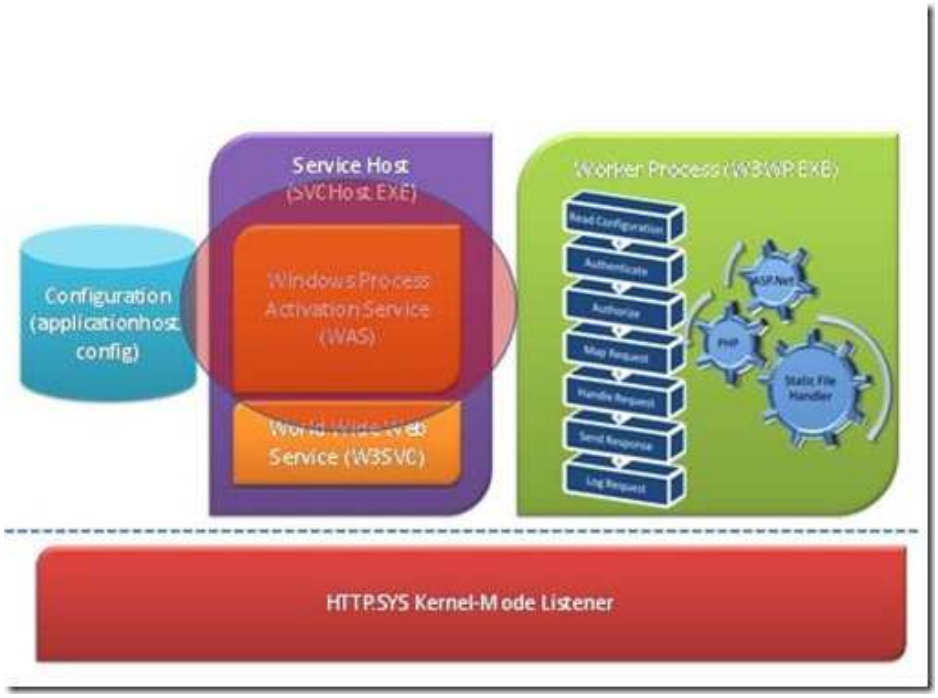


Figura 2.8: Diagramma architetturale

2.1.3.1.4 Descrizione del file ApplicationHost.config

Il file **ApplicationHost.config** della directory **%WINDIR%\System32\InetSrv\Config** contiene le configurazioni di tutti i siti, applicazioni, directory virtuali ed *application pool* del server Web (simile al *machine.config* e al *web.config* per la configurazione del Framework .NET). Questo file sostituisce il file *metabase.xml* che era usato dalla versione IIS 6.0; ciascun processo di lavoro W3WP.exe ha un'istanza del componente lettore con cui recupera direttamente la configurazione. Quando viene apportato un cambiamento ad un file, esso è immediatamente recepito (*picked up*) dal processo di lavoro ed esteso al sito, applicazione o directory virtuale nella macchina. *ApplicationHost.config* è l'unico file di configurazione disponibile al termine dell'installazione del server Web, ma è possibile aggiungere ulteriori file di configurazione *web.config* a qualunque livello della gerarchia dell'URL. Esso include una sezione speciale (*configSections*) per registrare le sezioni di IIS e di WAS che sono soggette a docking-down, per non essere sovrascritte dai file *Web.config* dei livelli sottostanti della gerarchia. Esso è letto nell'ordine in cui si presentano le sezioni, che sono auto esplicative.

Nella directory **%WINDIR%\System32\InetSrv\Config\Schema** devono essere presenti almeno i seguenti file: *IIS_schema.xml*, *ASPNET_schema.xml*, and *FX_schema.xml* files, che definiscono la struttura di configurazione per le sezioni applicate alle caratteristiche rispettivamente di IIS, di ASP.NET e del Framework .NET.

2.1.3.1.5 Gerarchia di configurazione

Oltre al file **applicationHost.config** si utilizzano:

- il file *machine.config* per definire le proprietà delle caratteristiche del Framework .NET;
- il file *web.config* (nella posizione radice della gerarchia dell'URL) per definire le impostazioni globali delle applicazioni ASP.NET.

I tre file *machine.config*, *root Web.config* ed *applicationHost.config*, che devono essere letti in sequenza per garantire la corretta configurazione del sistema, sono mantenuti volutamente distinti per garantire l'interoperabilità di versioni diverse del Framework .NET per la stessa versione di IIS. Successivamente viene letto ogni file web.config, se aggiunto ed applicato alla gerarchia. I file "figli" ereditano le proprietà dei "padri" a partire dal *machine.config* fino all'ultimo file *web.config* (se presente) e l'effettiva configurazione è calcolata per il dato path. La sezione <*system.WebServer*> del file *applicationHost.config* contiene le impostazioni di default dei website e delle directory virtuali, che possono essere sovrascritte per mezzo di appropriate sezioni <configuration> del file *web.config* all'interno dell directoy virtuale dell'applicatione web o directory virtuale di livello più basso all'interno della gerarchia del website. Mentre i file *machine.config*, *root Web.config* e *applicationHost.config* sono applicati ad ogni sito, applicazione o directory virtuale del sistema, la configurazione contenuta nei file *web.config* opzionali si applica verso il basso solo a partire dalla locazione in cui si trovano. Tale comportamente di ereditarietà è svolto per default.

2.1.3.1.6 Elaborazione di una richiesta HTTP in IIS 7.0

In figura 2.9 viene mostrata l'elaborazione di una richiesta HTTP in IIS 7.0, che viene descritta nel seguito:

1. un browser invia la richiesta HTTP di una risorsa sul server Web; tale richiesta viene intercettata dal *HTTP.sys*.
2. l'*HTTP.sys* contatta il *WAS* per ottenere le informazioni di configurazione.
3. il *WAS* legge il file *applicationHost.config* per recuperare le informazioni di configurazione richieste;
4. il *WWW Service* riceve le informazioni di configurazione, come l'*application pool* e il *site configuration*;
5. il *WWW Service* utilizza tali informazioni di configurazione per settare correttamente l'*HTTP.sys*;
6. il *WAS* avvia un processo di lavoro W3WP.exe per l'*application pool*, a cui è rivolta la richiesta;
7. il W3WP.exe elabora la richiesta e restituisce una risposta all'*HTTP.sys*;
8. il client riceve una risposta [9, 10, 11].

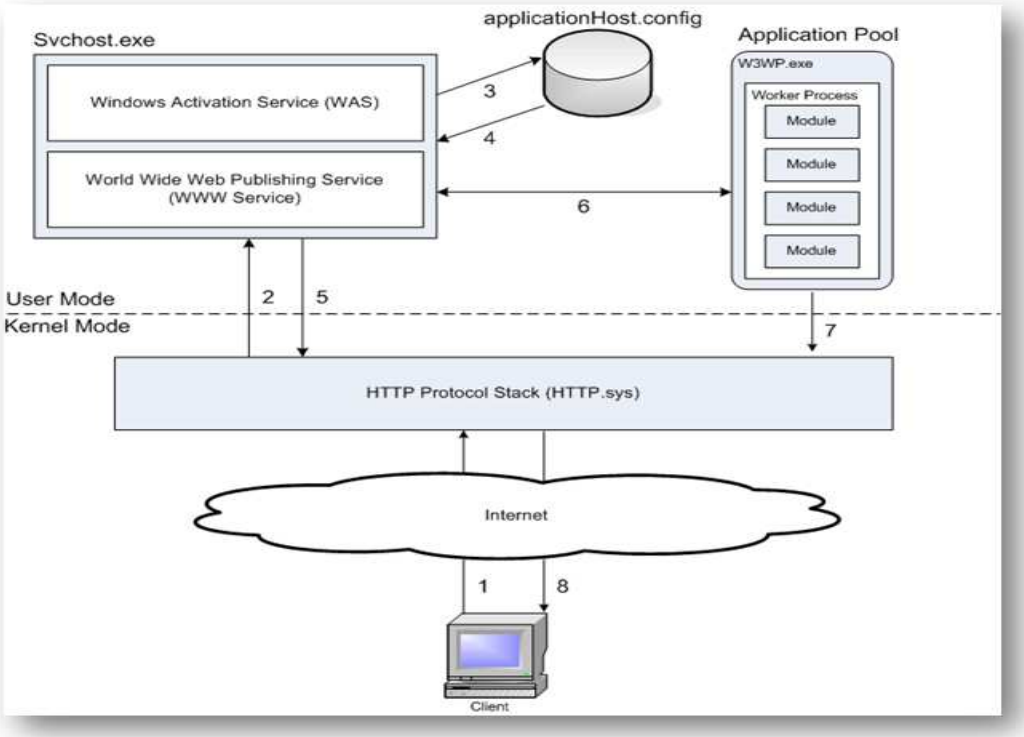


Figura 2.9: elaborazione di una richiesta HTTP in IIS 7.0

2.1.3.2 Motore d'esecuzione degli script ASP.NET

Il motore d'esecuzione degli script ASP.NET estende i servizi del server Web **Internet Information Services (IIS)** di Microsoft.

"ASP.NET is a server–side technology for developing Web applications based on the Microsoft .NET Framework"

L'applicazione Web eCentral è resa interattiva mediante l'uso della tecnologia ASP.NET, con cui gli utenti possono eseguire una logica di business con un browser Web. ASP.NET è una tecnologia lato server, poiché il codice (o script) ASP.NET, con cui si rende dinamico il contenuto di una pagina Web, viene interpretato ed eseguito dal motore d'esecuzione ASP.NET. Prima di spedire in risposta la pagina ASP.NET, il motore la rende conforme alle cosiddette tecnologie lato client, come HTML, JavaScript e Cascading Style Sheets (CSS), in quanto esse rappresentano gli unici linguaggi che il browser è in grado di interpretare. I controlli delle pagine dell'applicazione ASP.NET, che contengono l'attributo *runat = server*, sono compilati in assembly. Gli script delle pagine ASP.NET hanno accesso a tutte le risorse del server database SQL Server 2005 tramite la libreria standard di accesso ai dati ADO.NET. In figura 2.10 viene mostrata l'interazione di un utente con l'applicazione Web eCentral.

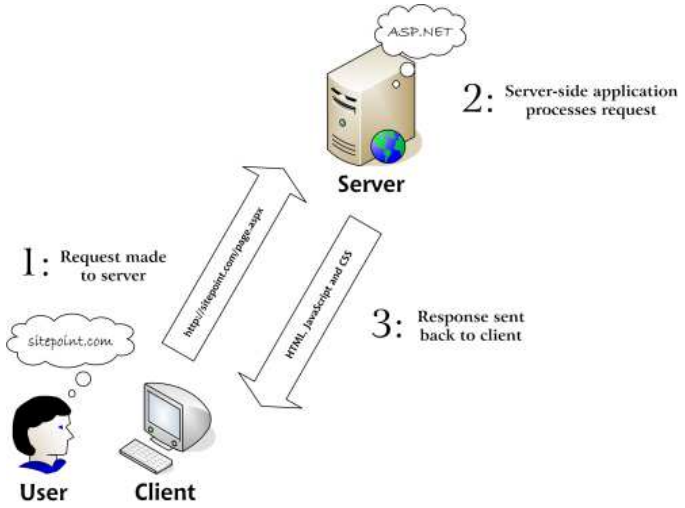


Figura 2.10: l'interazione di un utente con l'applicazione web eCentral

Per poter utilizzare ASP.NET sono stati installati e mandati in esecuzione i seguenti componenti software:

- **Internet Information Services 7.0 (IIS)**;
- **Framework .NET 3.5**: raccoglie tutte le tecnologie necessarie per costruire applicazioni Windows, applicazioni Web e Web services in un unico package e li rende disponibili in più di 40 linguaggi di programmazione (i più popolari sono C# e Visual Basic) [12, 13].

2.1.3.3 Interazione tra IIS e ASP.NET

Il motore ASP.NET si basa su un'architettura estensibile conosciuta come HTTP pipeline (nel seguito denominata *pipeline ASP.NET*), che gestisce le richieste dei client per l'applicazione Web eCentral (più di una semplice directory virtuale) e fornisce le risposte corrispondenti. Come suggerisce il nome, l'HTTP pipeline è strutturata come una catena, che può essera estesa e personalizzata a piacimento, con moduli e handler HTTP. I **moduli HTTP** forniscono funzionalità generali come *autenticazione*, *autorizzazione*, *gestione dello stato della sessione*, *ruoli*, *profili*, *logging* e *compressione della richiesta*. Gli **handler HTTP** sono competenti dell'elaborazione di un file sulla base della sua estensione specificata nell'URL. I moduli e gli handler sono denominati "native" se sono implementati in C/C++ oppure "managed", se sono scritti in un linguaggio ammesso dal Framework .NET. Ogni richiesta passa attraverso uno o più moduli HTTP, ma raggiunge un unico handler HTTP, incaricato di elaborarla. La richiesta HTTP, terminata l'elaborazione, ritorna al punto di partenza ripercorrendo i moduli HTTP.

2.1.3.3.1 Due tipi di application pool: classica ed integrata

IIS 7.0 supporta due tipi di *application pool*. Il primo di questi è chiamato *classic mode* ed è identico a quello offerto da IIS 6.0, con il solo supporto per filtri ISAPI classici (moduli "unmanaged"). Il secondo viene chiamato *integrated pipeline* ed offre la possibilità di condividere l'esecuzione di moduli "managed" e "unmanaged", in risposta ai normali eventi che l'applicazione Web scatena tra richiesta e risposta. In tale modalità integrata, infatti, IIS 7.0 consente ai moduli ASP.NET di collegarsi direttamente al server Web di base per mezzo di un modulo HTTP a livello globale chiamato *ManagedEngine*, che carica la DLL di supporto ASP.NET (*webengine.dll*) nella pipeline di elaborazione della richiesta di IIS 7.0.

2.1.3.3.1.1 Modalità classica (o ISAPI mode)

La *modalità classica* fa riferimento alla modalità di integrazione del runtime ASP.NET nel server Web IIS versione 6.0, denominata *IIS 6.0 worker process isolation mode*. Nel rispetto di tale modalità le richieste ASP.NET, dopo aver attraversato i filtri "unmanaged" in IIS, sono instradate all'estensione ISAPI *aspnet_isapi.dll* [14].

IIS 7.0 non carica il *ManagedEngine* nei processi di lavoro in *modalità classica* e, di conseguenza, la pipeline di elaborazione della richiesta non può risolvere o richiamare i moduli ASP.NET per richieste di qualunque tipo (o estensione). IIS 7.0, invece, attiva diversi gestori degli eventi, che sono basati su *IsapiModule* (*aspnet_isapi.dll*), per elaborare i tipi di contenuto ASP.NET in modo simile a IIS versione 6.0. Questo fatto è evidente dall'analisi della sezione dei gestori degli eventi in *system.webServer* nel file di configurazione *applicationHost.config*. Ad esempio, se si cerca la stringa *.aspx*, si individua una voce che punta a *PageHandlerFactory–Integrated* (caricato nei processi di lavoro del pool di applicazioni in modalità integrata secondo l'impostazione *preCondition="integratedMode"*) e si trovano altre voci che puntano a *PageHandlerFactory–ISAPI–2.0* e *PageHandlerFactory–ISAPI–2.0–64* (caricati nei processi di lavoro del pool di applicazioni in modalità classica secondo l'impostazione *preCondition="classicMode"*).

Un'estensione ISAPI (scritta in C++, Visual Basic, PHP, Perl) è una DLL che elabora la richiesta di una risorsa (URL) con una precisa estensione. L'estensione ISAPI che elabora le richieste per ASP.NET è l'*aspnet_isapi.dll*. La richiesta deve attraversare i moduli nativi di autenticazione ed autorizzazione della pipeline di IIS prima di raggiungere l'estensione ISAPI competente.

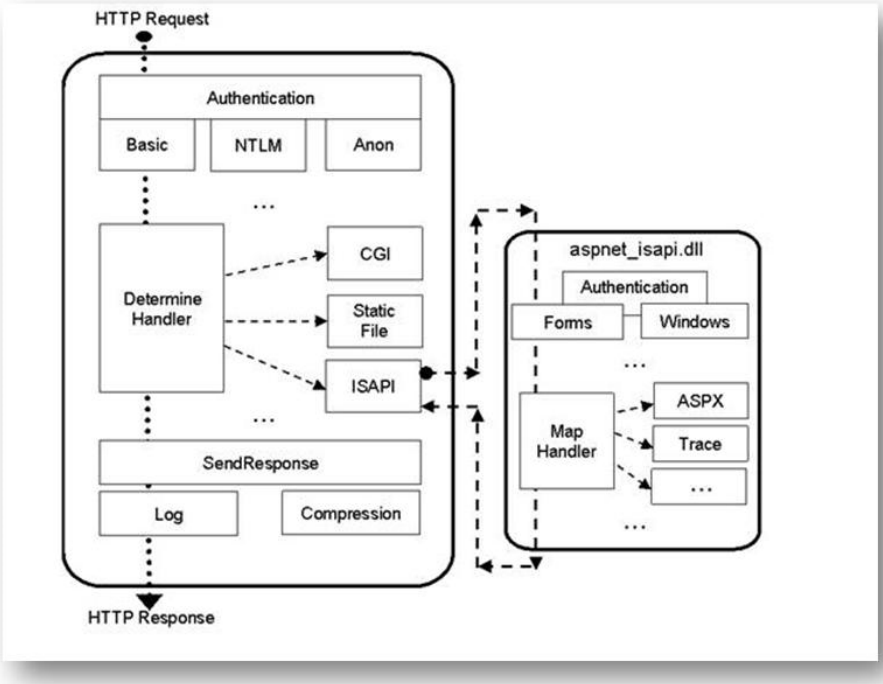


Figura 2.11: percorso di una richiesta in IIS

Nella figura 2.11 viene mostrata una richiesta contenente codice ASP.NET che, dopo aver attraversato alcuni moduli nativi (*riquadro a sinistra*), viene instradata da IIS all'estensione ISAPI competente in base alla configurazione della *scriptmap*, in cui sono contenuti i *file mappings* (estensione del file → estensione ISAPI competente). Essendo il file richiesto di estensione *.aspx*, la richiesta viene inoltrata alla DLL ASP.NET ISAPI (*aspnet_isapi.dll*) e da questa al runtime .NET che ospita la *pipeline ASP.NET*. In IIS 6.0 l'*aspnet_isapi.dll* e il *runtime .NET* vengono eseguiti nel medesimo processo di lavoro IIS W3WP.exe, per cui la loro comunicazione è di tipo *in–process*. In IIS 5.x, invece, se il livello di isolamento è medio o elevato, quando la richiesta ASP.NET entra nell'*aspnet_isapi.dll* (eseguita nel processo IIS denominato **inetinfo.exe**) viene generato un processo separato denominato *aspnet_wp.exe*, che carica ed ospita il runtime .NET. In IIS 5.x, pertanto, ogni richiesta viene instradata dall'*aspnet_isapi.dll* all'*aspnet_wp.exe* mediante *Named Pipe* (poco efficiente), come mostrato nella figura 2.12:

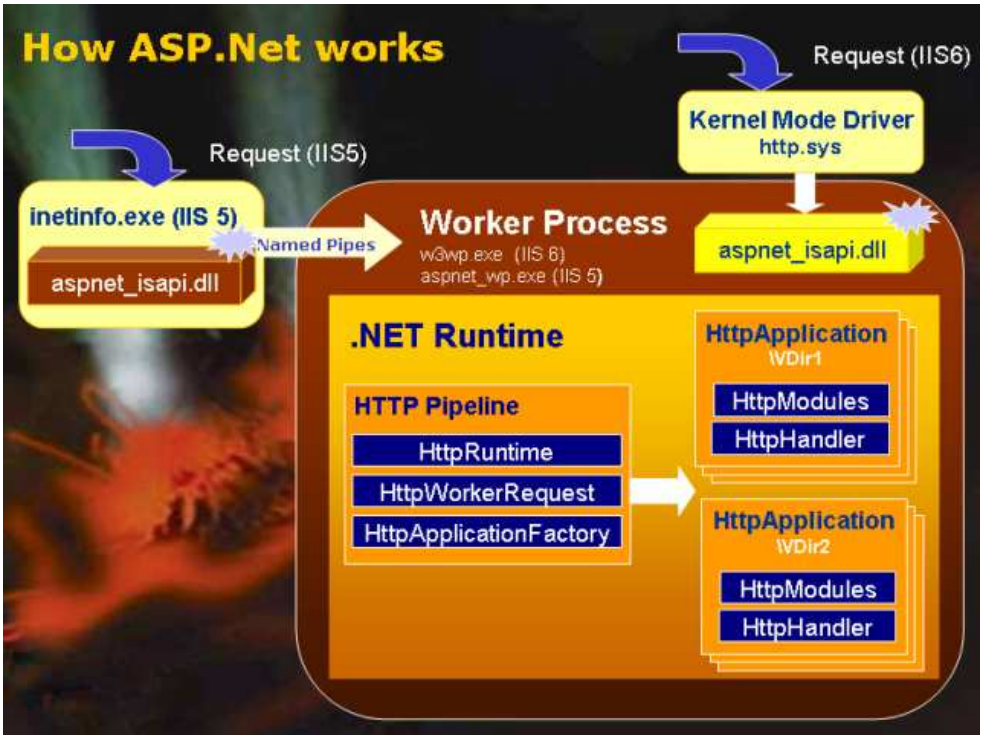


Figura 2.12: IIS 5.0 vs IIS 6.0

Poiché nella *modalità classica*, dunque, si fa affidamento sulle estensioni ISAPI (ed in particolare modo all'*aspnet_isapi.ddl* per richieste rivolte al motore ASP.NET), se i file con estensione diversa da *.aspx* (come *.php*) non sono mappati verso l'*aspnet_isapi.dll*, le corrispondenti richieste non sono inoltrate al runtime .NET, per cui i controlli della pipeline ASP.NET, come l'autenticazione, non vengono eseguiti su tali richieste; è necessario, pertanto, duplicare la logica di autenticazione per le applicazioni non mappate.
La *modalità classica* di IIS 7.0 mantiene, pertanto, la compatibilità con le versioni precedenti di IIS attraverso l'utilizzo di un'estensione ISAPI per richiamare il runtime .NET. Di norma questa opzione richiede modifiche minime o nulle alle applicazioni esistenti. Se un'ISAPI DLL è configurata per processare una certa richiesta, IIS 7.0 tratta tale ISAPI DLL come un qualunque altro suo handler, senza eccezioni [15].

2.1.3.3.1.2 Modalità integrata

Con la **modalità integrata** la pipeline ASP.NET non è contenuta in quella di IIS, ma si sovrappone interamente ad essa (fornendo un wrapper su di essa), per cui IIS ha un'unica pipeline. I moduli "managed" e "native" vengono, pertanto, richiamati nello stesso istante, in risposta agli stessi eventi. I moduli di ASP.NET possono essere eseguiti prima o dopo il corrispondente modulo di IIS 7.0, o addirittura sostituirsi ad esso.
I moduli nativi usano l'API del server Web principale di IIS per registrarsi a gestire specifici eventi nella pipeline come: *BeginRequest*, *AuthenticateRequest*, *AuthorizeRequest* e *ExecuteRequestHandler*. Il

ManagedEngine fornisce il supporto necessario a integrare anche i moduli ASP.NET nella pipeline di elaborazione della richiesta. Con questa nuova architettura IIS 7.0 può richiamare i moduli nativi ed ASP.NET in qualsiasi momento durante l'elaborazione della richiesta, indipendentemente dal tipo di contenuto della richiesta. IIS 7.0 è dotato, infatti, di moduli "managed" *WindowsAuthentication* e *FormsAuthentication*, che generano eventi *OnAuthenticate* quando la pipeline di elaborazione della richiesta genera l'evento *AuthenticateRequest*. A questo punto un modulo "nativo", come *RequestFilteringModule* o *IpRestrictionModule*, può gestire l'evento *BeginRequest* in modo da rifiutare richieste critiche ed eseguire successivamente il codice di autenticazione ASP.NET personalizzato in corrispondenza dell'evento *AuthenticateRequest*; infine, se la richiesta è di tipo php, il *FastCgiModule* esegue il motore di scripting di php (*php–cgi.exe*) in corrispondenza dell'evento *ExecuteRequestHandler* per elaborare le pagine php.
Tale modalità garantisce prestazioni migliori in quanto non si deve percorrere due volte la stessa pipeline, una volta per i moduli "native" (gestiti solo da IIS) ed una per quelli "managed" (gestiti solo da ASP.NET); fornisce, inoltre, modularità per la configurazione e l'amministrazione, perchè permette di estendere e personalizzare IIS in ogni fase del ciclo di vita di una richiesta aggiungendo un modulo o un handler personalizzato. Permette, inoltre, di mandare in esecuzione solo i moduli e gli handlers da utilizzare, riducendo sia l'utilizzo di risorse per il processo W3WP.exe che eventuali punti di debolezza sfruttabili per attacchi dal Web. La personalizzazione della pipeline di elaborazione delle richieste di IIS non richiede l'implementazione di API specifiche per IIS. È possibile, ad esempio, definire nella cartella *App_Code* dell'applicazione Web e registrare nel *web.config* un modulo (come il "managed-code" *ASP.NET forms authentication*) per le richieste IIS di una qualunque estensione di file (come file statici, classici ASP, PHP, CGI) [16, 17].
In figura 2.13 sono mostrate le fasi di elaborazione di una richiesta con codice ASP.NET in cui sono coinvolti i moduli HTTP:

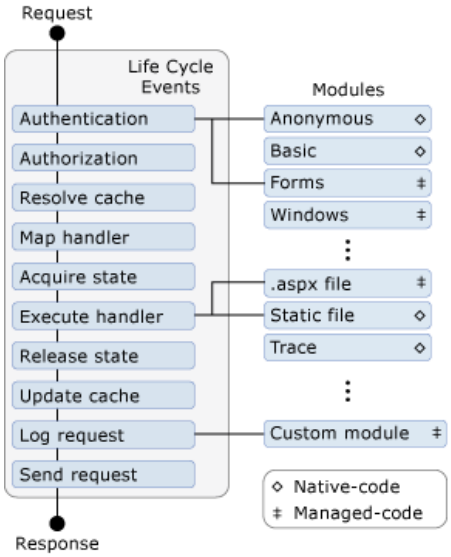


Figura 2.13: ciclo di vita di una richiesta

Se una certa estensione non è gestita da alcun handler HTTP, la richiesta della risorsa con quella particolare estensione viene gestita dal server Web direttamente. Affinchè tale richiesta sia accettata e processata dal server Web, occorre che sia configurata con tipo MIME valido, altrimenti viene rifiutata.
Come effetto collaterale causato da questa nuova architettura si ha che le applicazioni ASP.NET esistenti potrebbero richiedere modifiche al codice e alla configurazione, che dovrebbero essere eseguite per garantire che tali modifiche non creino conflitti tra i moduli nella pipeline di elaborazione della richiesta. Se non è possibile trasferire immediatamente un'applicazione Web esistente, è possibile attivare la *modalità classica* nel pool di applicazioni usato dal processo di lavoro per eseguire l'applicazione Web.
Poiché la modalità della pipeline è impostata a livello di pool di applicazioni, IIS 7.0 può eseguire le applicazioni Web in *modalità integrata* e in *modalità classica* contemporaneamente, per cui processi di lavoro W3WP.exe dovranno semplicemente eseguire diversi pool di applicazioni, ma potranno essere ospitati nello stesso server. Si ricorda che la *modalità integrata* e quella *classica* influiscono solo su come IIS 7.0 integra ASP.NET nella pipeline di elaborazione della richiesta, per cui tutti gli altri moduli nativi vengono caricati senza precondizioni di modalità pipeline sia nella modalità *integrata* che in quella *classica* [18].

2.1.3.3.2 Moduli e handler HTTP

I *moduli HTTP* sono equivalenti ai *filtri ISAPI ("unmanaged/native")*, mentre gli *handler HTTP* lo sono per le *estensioni ISAPI*.
Un *handler HTTP* è una DLL che, in base all'estensione del file richiesto nell'URL, cattura dalla coda di gestione la richiesta di un file e la serve. Il più comune handler è l'*ASP.NET page handler* che processa i file con estensione *.aspx*.
Un *modulo HTTP* è una classe, composta dai metodi *Dispose* e *Init*, che implementa l'interfaccia *IHttpModule* ed accetta come parametro un'istanza della classe *HttpApplication*, che rappresenta l'applicazione corrente. Gli HttpModule sono caricati ed instanziati all'avvio dell'applicazione, attraverso la lettura nel web.config delle classi registrate nella sezione:

```
<configuration>
    <system.Web>
        <httpModules>
```

2.1.3.4 Configurazione del website eCentral

Il *Default Web Site* coincide con il server Web IIS 7.0 a cui è associata la directory fisica *C:\Inetpub\wwwroot*, come nel seguito mostrato in figura 2.14:

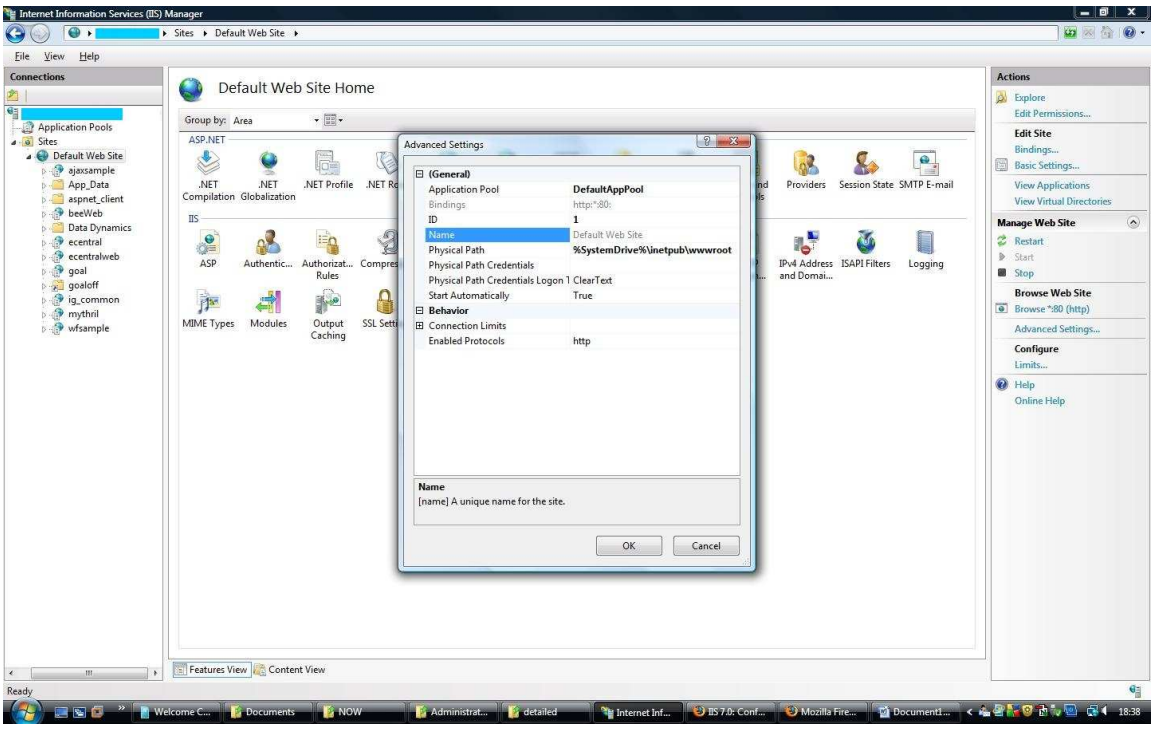


Figura 2.14: Default Web Site

In *C:\Inetpub\wwwroot* vengono normalmente collocate le directory fisiche associate alle directory virtuali (gestione più logica), ma nulla vieta, comunque, di utilizzare una directory fisica diversa, come ad esempio:

(*Label:* dir 1) *C:\Users\pmaschi\Documents\eCentral\trunk\comps\eCentralWeb*

alla quale accedono gli utenti tramite browser, specificando l'URL della directory virtuale associata alla precedente (dir 1), come ad esempio:

(*Label:* dir 2) *http://pmaschi.olimpo.intranet/ecentral/login/login.aspx*

Le directory virtuali sono di due tipi:

– *directory semplici*, repository di file accessibli dal mondo esterno, che ereditano dal padre, il Website, qualunque configurazione, comportamento (sicurezza, esecuzione, etc.) e stato;
– *directory virtuali, configurate come applicazioni distinte* con i propri ed autonomi: stato di sessione, comportamento e configurazione; eCentral è una directory di tipo virtuale.

Per associare eCentral contenuta nella directory

C:\Users\pmaschi\Documents\eCentral\trunk\comps\eCentralWeb

alla pipeline ASP.NET (anziché ad un server ASP o PHP) occorre seguire i passi descritti nel seguito:

– nell'*Internet Information Services (IIS) Manager* si preme right–click su *Default Web Site* e si seleziona *Add Application*; l'*Add Application Wizard* appare come in figura 2.15:

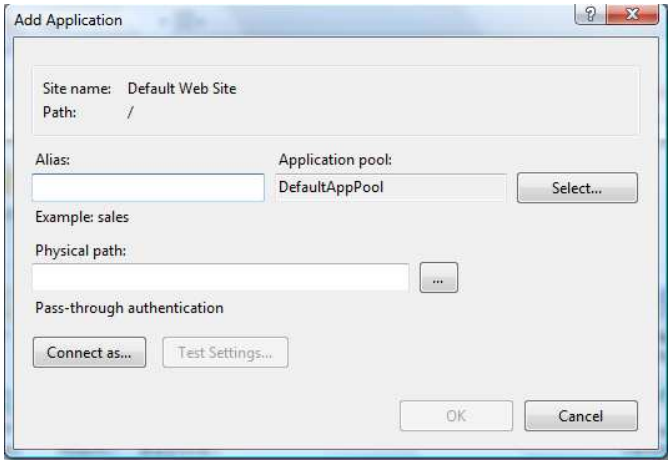


Figura 2.15: Add Application

- si sceglie un **alias** (ad esempio *ecentral*) per attribuire il nome al processo, in cui verrà eseguita l'applicazione eCentral; si seleziona l'**application pool** con "Select…" e nel campo "Physical path" si specifica il percorso assoluto dell'applicazione eCentral.
- se i file, esposti (pubblicati) dall'applicazione eCentral, sono memorizzati in una rete condivisa, occorre connettersi con le credenziali specifiche, indicandole, cliccando sul bottone **"Connect as"**.

Cliccando su OK, *ecentral* appare come figlio del *Default Web Site*, come illustrato in figura 2.16.

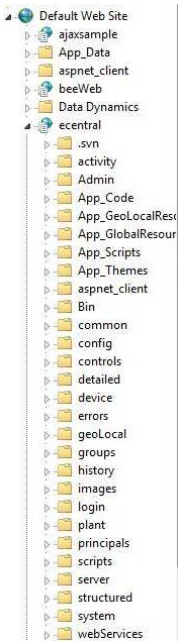


Figura 2.16: struttura del website di eCentral

È possibile aggiornare in ogni momento qualunque parte di eCentral senza dover riavviare il server Web ed

interrompere la gestione delle richieste esistenti. ASP.NET offre tale vantaggio, perchè non usa i file ASP.NET della directory virtuale, ma le copie dei file nella directory *C:\Windows\Microsoft.NET\v3.5\Temporary* creati durante la fase di compilazione, che pertanto risultano locked. ASP.NET crea un pool di oggetti "Application" quando il dominio applicativo viene caricato per la prima volta e usa uno di questi oggetti per servire ogni richiesta. Ciascuna richiesta ottiene, pertanto, l'accesso esclusivo ad uno di questi oggetti, che viene riutilizzato quando la richiesta termina.

2.1.3.4.1 File "machine.config" del Framework 2.0

Tutte le applicazioni .NET ereditano le impostazioni di configurazione di base e quelle predefinite da un file denominato *C:\Windows\Microsoft.NET\Framework\v2.0.50727\Config\Machine.config*, utilizzato per le impostazioni di configurazione del server. Per alcune di queste impostazioni non è possibile eseguire l'override nei file di configurazione presenti a livelli inferiori della gerarchia.

Nelle applicazioni client .NET, come il servizio MS Windows che implementa l'ECD, per eseguire l'override delle impostazioni ereditate vengono utilizzati file di configurazione denominati *ApplicationName.config*. Per eseguire la medesima operazione, nelle applicazioni ASP.NET, come eCentral, vengono utilizzati file di configurazione denominati *web.config* [19].

In tale file:

1. sono registrati i providers, che definiscono pattern per compiere funzioni, come la gestione delle *membership role–based security* e dei *profili*. Un provider è un insieme di classi con metodi per fornire servizi ad ASP.NET.;
2. è configurato il processo di lavoro *aspnet_wp.exe* nella sezione <*processModel*>.

Nella sezione <*processModel*>, in particolare, si configura:

1. il riciclaggio dei domini applicativi;
2. lo specifico account Windows, sotto il quale è eseguito il processo di lavoro, determinandone i privilegi.

Occorre settare a *true* l'attributo *enable* della sezione <*processModel*>, affinché ASP.NET venga eseguito rispettando la configurazione specificata in tale sezione.

Nella sezione <*machineKey*> si setta la chiave specifica del server per criptare i dati con un determinato algoritmo (3DES, SHA1, etc.). ASP.NET la usa automaticamente per proteggere i cookie dei *form d'autenticazione* (cookie di sessione) e per proteggere i dati del *view state* (vedi paragrafo view state).

Come in ASP.NET 1.x anche nella versione 2.0, quando si utilizzano gli *application pool*, molti dei settaggi della sezione <*processModel*> del *machine.config* vengono ignorati, in quanto configurabili per ogni *application pool* dei processi di lavoro W3WP.exe di IIS 7.0.

Nella versione 2.0 i settaggi non ignorabili nel *machine.config* sono i seguenti:

autoConfig, maxIoThreads, maxWorkerThreads, minIoThreads, minWorkerThreads, requestQueueLimit, responseDeadlockInterval

Il parametro *autoConfig* è una novità della 2.0 e consente di configurare in modo ottimale gli attribute *maxWorkerThreads*, *maxIoThreads*, *minFreeThreads* (HttpRutime), *minLocalRequestFreeThreads* (HttpRuntime), *maxConnection* (connectionManagement Element).
Ai parametri *maxWorkerThread* e *maxIOThread* sono stati aggiunti i corrispondenti valori minimi (*minWorkerThread* e *minIOThread*) [20].

2.1.3.4.2 Configurazione di ASP.NET per eCentral con il file Web.config

L'elemento principale della gerarchia di configurazione di ASP.NET è un file denominato *web.config* principale, presente nella stessa directory del file *machine.config*. Il file *web.config* principale eredita tutte le impostazioni del file *machine.config* e contiene impostazioni utilizzabili per tutte le applicazioni ASP.NET che eseguono una versione specifica del Framework .NET. Poiché ogni applicazione ASP.NET eredita le impostazioni di configurazione predefinite dal file *web.config* principale, è necessario creare il file *web.config* esclusivamente per le impostazioni che eseguono l'override di quelle predefinite.
La gerarchia di configurazione di ASP.NET presenta le caratteristiche seguenti:

- consente di utilizzare i file di configurazione che si applicano alle risorse presenti nella directory di appartenenza del file e a tutte le directory figlio;
- consente di inserire i dati di configurazione nell'ambito appropriato: l'intero computer, tutte le applicazioni Web, una singola applicazione o una sottodirectory all'interno dell'applicazione;
- consente l'override delle impostazioni di configurazione ereditate dai livelli superiori della gerarchia di configurazione. Consente inoltre di bloccare le impostazioni di configurazione in modo da impedirne l'override da parte di quelle presenti in un livello inferiore;
- consente di organizzare in sezioni i gruppi logici delle impostazioni di configurazione [19].

La configurazione di ASP.NET per l'applicazione eCentral è inserita nella sezione radice <*configuration*>.
La creazione del file *web.config* di eCentral prevede le seguenti sezioni:

```
<?xml version="1.0"?>
<configuration>
</configuration>
```

L'utilizzo di unico file di configurazione per diverse applicazioni Web (anziché uno per ogni applicazione) facilita la gestione e garantisce la sicurezza. L'elemento <configuration> contiene a sua volta una sezione <*system.Web*>, che verrà descritta nel seguito. Allo stesso livello di <system.Web> sono presenti altre sezioni:

- <*appSettings*> usato per salvare settaggi personalizzati;
- <*connectionStrings*> utilizzato per salvare le stringhe di connessione alla base dati usata

dall'applicazione, per poter apportare modifiche senza dover ricompilare l'intera applicazione. Nel seguito è mostrato un esempio per eCentral:

<connectionStrings>

<add name="ECENTRALConnectionString" connectionString="Data Source=PRODESNA\SQLSERVER2005;Initial Catalog=ECENTRAL;Persist Security Info=True;User ID=;Password=*" providerName="System.Data.SqlClient"/>*

</connectionStrings>

In tale elemento si specificano il nome della stringa di connessione, che è *ECENTRALConnectionString*, ed i seguenti attributi:

- **Data Source:** è il DBMS contenuto nella macchina virtuale PRODESNA;
- **Initial Catalog:** è il nome della base di dati;
- **Persist Security Info=true:** per limitare l'accesso alla base dati; è, infatti, necessario proteggere le informazioni di connessione: ID utente, password e nome della base dati. Con tale elemento è possibile ottenere informazioni riservate, compresi ID utente e password, dalla connessione stessa dopo la sua apertura;
- **User ID=* Password=*:** non si utilizza l'autenticazione Windows (nota anche come protezione integrata), ma si specifica una combinazione di user ID e password validi per il DBMS SQL Server; l'autenticazione è completamente a carico di SQL Server;
- **providerName="System.Data.SqlClient":** specifica il provider di dati del Framework .NET di SQL Server, che è l'insieme di classi utilizzato per accedere al DBMS SQL Server.

Anzichè utilizzare il file *machine.config*, ASP.NET permette di apportare modifiche alla configurazione del server Web mediante le seguenti sezioni del file *web.config*:

- **configSections:** permette di memorizzare informazioni personalizzate usate dall'applicazione in modo strutturato (liste o gruppi di settaggi correlati) e con differenti tipi di dato, non solo stringhe. È possibile specificare settaggi correlati concettualmente all'interno di gruppi come nell'esempio:

  <sectionGroup name="system.Web.extensions" type="System.Web.Configuration.SystemWebExtensionsSectionGroup, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35">)

  oppure singolarmente, come nell'esempio:

  *<section name="**scriptResourceHandler**" type="System.Web.Configuration.ScriptingScriptResourceHandlerSection, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" requirePermission="false" allowDefinition="MachineToApplication"/>*

  Lo ***scriptResourceHandler*** è il gestore degli script. Il *PublicKeyToken* è il token associato all'assembly per verificare l'autentiticità della firma del costruttore.

Per salvare le informazione di una sezione, si utilizzano degli attributi, ad esempio:

– l'attributo "name" indica il nome della sezione;
– l'attributo "type" indica il nome della classe C# corrispondente;
– etc.

Per una maggiore comprensione si propone il seguente esempio di carattere generale:

```
<section>
    <newElement attributo1="true" attributo2="xyz" />
```

Alla sezione di nome *newElement* corrisponde la classe C# mostrata nel seguente esempio:

```
public class NewElement: ConfigurationSection
{
    [ConfigurationProperty ("attributo1", IsRequired=false, DefaultValue=true)]
        public bool Attributo1
        {
        get { return (bool) base ["attributo1"];}
        set { base ["attributo1"] = value;}
        }
        …
}
```

Quando viene eseguita l'applicazione, ASP.NET legge l'informazione dall'elemento contenuto nel file di configurazione per creare un'istanza della classe **NewElement**. Si, può, poi accedere in ogni momento all'informazione dell'oggetto instanziato. Per recuperare, ad esempio, nella pagina desiderata l'informazione dalla sezione personalizzata <newElement … />, si usa la seguente stringa di codice:

```
Configuration config = WebConfigurationManager.OpenWebConfiguration("/");
```

Viene in questo modo recuperato un oggetto Configuration, che contiene tutte le informazioni di configurazione di eCentral. Con il metodo *GetSection("newElement")* dell'istanza *config* si recupera, poi, l'informazione della sezione *newElement*, come nel seguito mostrato:

```
NewElement sezionePersonalizzata = (NewElement) config.GetSection("newElement");
sezionPersonalizzata.attributo1;
```

- **system.codedom:** permette la configurazione dei compilatori (C# e Visual Basic), mostrata nel seguito:

```
<system.codedom>
    <compilers>
                <compiler language="c#;cs;csharp" extension=".cs"
                type="Microsoft.CSharp.CSharpCodeProvider,System, Version=2.0.0.0,
                Culture=neutral, PublicKeyToken=b77a5c561934e089" warningLevel="4">
                  <providerOption name="CompilerVersion" value="v3.5"/>
```

```
                    <providerOption name="WarnAsError" value="false"/></compiler>
                <compiler language="vb;vbs;visualbasic;vbscript" extension=".vb"
                type="Microsoft.VisualBasic.VBCodeProvider, System, Version=2.0.0.0,
                Culture=neutral, PublicKeyToken=b77a5c561934e089" warningLevel="4">
            <providerOption name="CompilerVersion" value="v3.5"/>
            <providerOption name="OptionInfer" value="true"/>
            <providerOption name="WarnAsError" value="false"/>
        </compiler>
    </compilers>
</system.codedom>
```

- **system.WebServer**: permette di specificare la configurazione del server Web attraverso le seguenti sezioni:
  - <*defaultDocument*>, con cui si indica il documento da visualizzare di default per una richiesta che non specifica un filename preciso. Quando si richiede una directory virtuale, senza specificare il nome di un file, IIS cerca una pagina con il nome di uno dei documenti di default, come index.aspx oppure home.aspx. Se non è presente la pagina di default, IIS mostra i contenuti della locazione richiesta. Nella sezione <directoryBrowse> si specifica l'attributo **enabled="false"** per impedire agli utenti di accedere direttamente ad un qualunque file o sottodirectory della directory virtuale;
  - <*httpCompression*>, con cui si indica la configurazione di compressione per le risposte;
  - <*customHeaders*> della sezione <httpProtocol>, con cui si indicano gli header HTTP;
  - <*modules*>, con cui si indicano i moduli HTTP;
  - <*handlers*>, con cui si indicano gli handler HTTP.

  Una porzione di tale sezione è mostrata nel seguito:

```
<system.WebServer>
    <validation validateIntegratedModeConfiguration="false"/>
    <modules>
            <remove name="ScriptModule"/>
            <add name="ScriptModule" preCondition="managedHandler"
            type="System.Web.Handlers.ScriptModule, System.Web.Extensions,
            Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
            …
    </modules>
    <handlers>
    <remove name="WebServiceHandlerFactory–Integrated"/>
    <remove name="ScriptHandlerFactory"/>
```

```
<remove name="ScriptHandlerFactoryAppServices"/>
                    <remove name="ScriptResource"/>
                    <remove name="WebServiceHandlerFactory–ISAPI–2.0"/>
                    <add name="ScriptHandlerFactory" verb="*" path="*.asmx"
                    preCondition="integratedMode"
                    type="System.Web.Script.Services.ScriptHandlerFactory, System.Web.Extensions,
                    Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
                    <add name="ScriptHandlerFactoryAppServices" verb="*"
                    path="*_AppService.axd" preCondition="integratedMode"
                    type="System.Web.Script.Services.ScriptHandlerFactory, System.Web.Extensions,
                    Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
                    <add name="ScriptResource" preCondition="integratedMode"
                    verb="GET,HEAD" path="ScriptResource.axd"
                    type="System.Web.Handlers.ScriptResourceHandler, System.Web.Extensions,
                    Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
                    …
        </handlers>
        <defaultDocument>
            <files>
                <clear/>
                <add value="home.aspx"/>
            </files>
        </defaultDocument>
        <directoryBrowse enabled="false"/>
</system.WebServer>
```

Con il valore “integratedMode” dell’attributo **preCondition** negli elementi della sezione <handlers> si specifica la modalità “integrata” di IIS 7.0.

2.1.3.4.2.1 Sezione <system.Web>

Questo elemento del file *web.config* contiene tutti i settaggi di configurazione specifici per ASP.NET. Con tali settaggi si configurano diversi aspetti dell’applicazione eCentral abilitando servizi come sicurezza, gestione dello stato, tracciamento delle pagine (ovvero la visualizzazione di informazioni di diagnostica nella pagina) ed etc. Nella creazione del file *web.config* di eCentral si specificano le seguenti sezioni:

- **compilation:** contiene gli elementi <assemblies> a supporto del runtime dell’infrastruttura del Framework .NET. Con l’attributo *debug=“true”* si permette di usufruire di un debug, che segnala la riga di codice

errata. Una porzione di tale sezione è mostrata nel seguito:

<compilation debug="true">

<assemblies>

<add assembly="System.Design, Version=2.0.0.0, Culture=neutral, PublicKeyToken=B03F5F7F11D50A3A"/>

<add assembly="System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=B77A5C561934E089"/>

<add assembly="System.Web, Version=2.0.0.0, Culture=neutral, PublicKeyToken=B03F5F7F11D50A3A"/>

…

</assemblies>

</compilation>

- **globalization:** set di caratteri rispetto al quale la pagina HTML è definita; tale sezione è mostrata nel seguito:

  <globalization culture="auto:it–IT" uiCulture="auto:it–IT"/>

- **authentication:** determina come verificare l'identità del client quando richiede una pagina; tale sezione è mostrata nel seguito:

  <authentication mode="Forms">

  <forms name="eCentralAUTH" loginUrl="~/login/login.aspx" defaultUrl="~/plant/plant.aspx" protection="All" timeout="240" slidingExpiration="true" cookieless="useCookie" requireSSL="true"/>

  </authentication>

  IIS è stato configurato per l'accesso anonimo per poter utilizzare in eCentral la modalità di "autenticazione forms". Tale modalità utilizza un *ticket di autenticazione* (creato quando un utente si logga al sito) che permette di aver una traccia delle azioni dell'utente attraverso il sito. Il "ticket di autenticazione form" viene cifrato e su di esso viene calcolato un digest; entrambi sono spediti all'interno di un cookie. Quando tale cookie viene restituito, il server IIS confronta il digest tramesso con il digest calcolato sul cookie per controllare se ha subito alterazioni. La cifratura è d'obbligo per evitare che un utente malizioso possa vedere o modificare le credenziali di autenticazione, con le quali potrebbe accedere al sistema.

  Se un utente richiede una pagina senza aver in precedenza effettuato il login, viene rediretto alla pagina di login, in cui devono essere inserite le sue credenziali di autenticazione come: username e password. Tali credenziali sono, poi, passate al server IIS e validate attraverso una query su una tabella apposita di SQL Server. La pagina di autenticazione di eCentral è mostrata in figura 2.17:

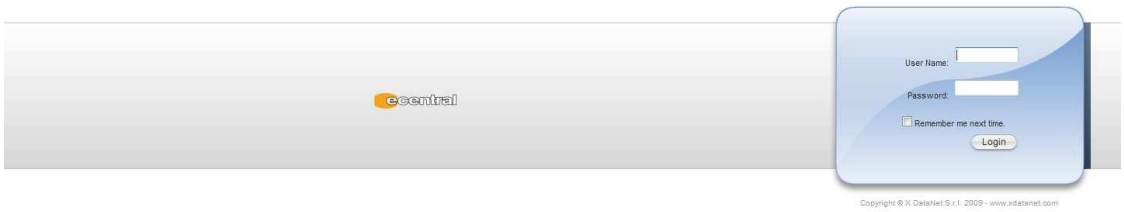


Figura 2.17: eCentral login

Una volta autenticato l'utente viene rediretto alla pagina originale (home.aspx).

La fase di autenticazione del ciclo di vita della richiesta di una pagina ASP.NET è gestita, tramite form di login, dalla classe *FormsAuthenticationModule*, che è un modulo HTTP. L'applicazione ASP.NET eCentral concede l'accesso agli utenti anonimi in due fasi:

1. IIS, essendo configurato per utilizzare un'autenticazione anonima (account *IUSR*), genera il ticket di autenticazione, che viene successivamente passato ad ASP.NET;
2. ASP.NET compie la sua autenticazione con il metodo specificato dall'attributo **mode** dell'elemento **authentication**.

Quanto appena descritto è mostrato in figura 2.18 di pag. 67.

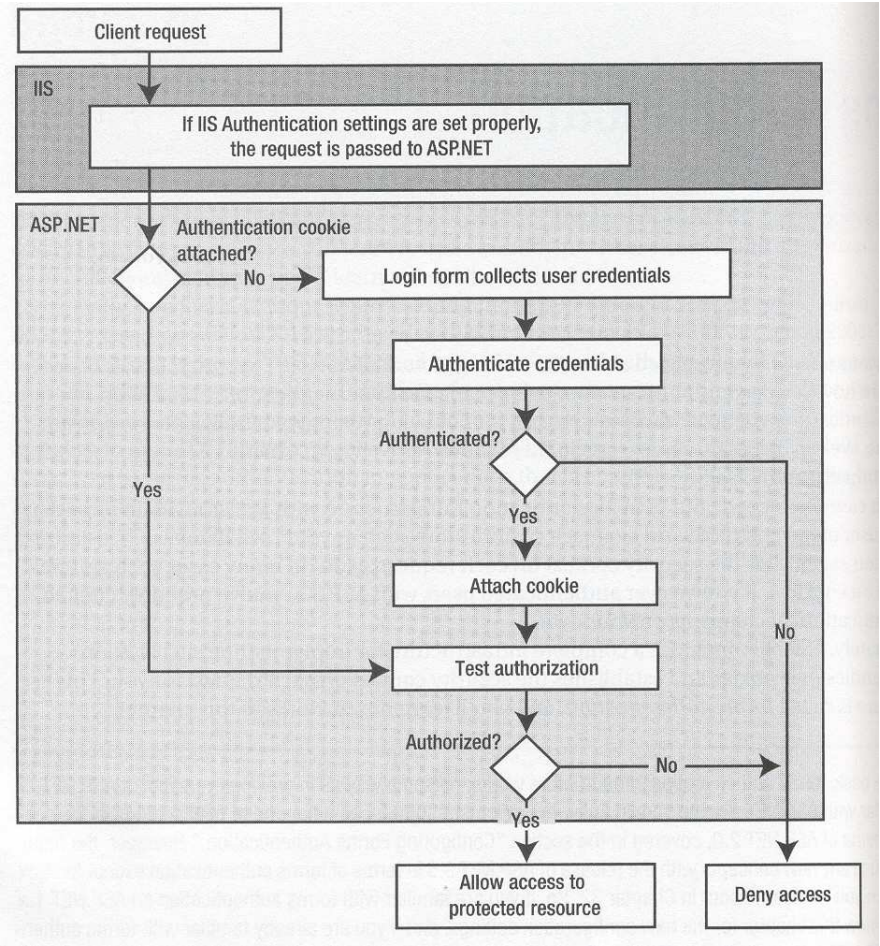


Figura 2.18: autenticazione

La descrizione degli attributi utilizzati nell'elemento "forms" sono:

- **loginUrl:** fa riferimento alla pagina di login personalizzata di eCentral. La cartella login, che contiene l'omonima pagina, dovrebbe richiedere Secure Sockets Layer (SSL), per assicurare l'integrità delle credenziali quando sono passate dal browser al server Web;
- **protection:** è settato ad **All** per specificare la confidenzialità (autenticità e non ripudio) e l'integrità del ticket del form di autenticazione (userID + password). Come conseguenza il ticket di autenticazione viene cifrato usando l'algoritmo specificato nell'elemento **machineKey** e può essere firmato usando l'algoritmo di hashing sempre specificato nello stesso elemento. Gli algoritmi che possono essere utilizzati sono SHA1, MD5, AES, 3DES;
- **timeout:** è usato per specificare il numero di minuti concessi prima che il cookie di autenticazione scada. Trascorsi 240 minuti, l'utente deve loggarsi di nuovo;

- **slidingExpiration:** è settato a true, affinché il timeout del cookie di autenticazione sia fatto ripartire da zero ad ogni richiesta di una pagina. Se mancano, ad esempio, 5 minuti allo scadere del tempo di vita del cookie e l'utente fa una richiesta, il tempo di vita assume nuovamente il valore di 240 minuti. Il tempo di vita del cookie viene, pertanto, esteso;
- **defaultUrl:** è settato con la pagina di default *~/plant/plant.aspx* di eCentral;
- **Cookieless:** è settato a "UseCookie" per forzare l'ASP.NET runtime ad utilizzare i cookie nella modalità form authentication. La form authentication non potrà funzionare se il browser non supporta i cookie e, pertanto, sarà redirezionato all'infinito alla pagina di login;
- **requireSSL:** è settato a "true" per indicare l'utilizzo del server SSL.

La classe **UrlAuthorizationModule** è utilizzata per garantire ai soli utenti autenticati di accedere ad una pagina, impedendo di bypassare il form di autenticazione. Tale configurazione viene specificata nella sezione mostrata nel seguito:

<authorization>

*<deny users="**?**" />*

</authorization>

Il '**?**' è una wildcard che esegue il "match" di tutti gli utenti anonimi.

Se un utente non autenticato tenta di accedere ad una pagina, il modulo di autenticazione form ridireziona l'utente alla pagina di login specificata dall'attributo **loginUrl** dell'elemento **forms**, dovendo ogni richiesta contenere il ticket di autenticazione. Nel seguito viene descritto un esempio concreto di scenario di autenticazione, con IIS 7.0 in esecuzione in locale:

1. si richiede la pagina:
   http://localhost/ecentral/detailed/diagnosticsdetailed.aspx della directory virtuale *ecentral*. IIS permette la richiesta, perchè l'accesso anomino è abilitato nella corrispondente sezione del file **ApplicationHost.config**. ASP.NET conferma che l'elemento **authorization** include il tag <deny users= "?" />;
2. ASP.NET cerca di recuperare il ticket di autenticazione; in caso contrario l'utente è rediretto alla pagina di login (login.aspx), come specificato dall'attributo **LoginUrl** dell'elemento **forms**. L'informazione della pagina di login è inserita in una query string usando come chiave *RETURNURL*. Il server HTTP invia al browser:
   302 Found Location:
   http://localhost/ecentral/login/login.aspx?ReturnUrl=%2fecentral%2fdetailed%2fdiagnosticsdetailed.aspx
3. il browser richiede la pagina "login.aspx" ed include il parametro RETURNURL nella query string;
4. il server restituisce la pagina di login e "200 OK HTTP status code";
5. il browser visualizza la pagina di login (in cui l'utente inserisce le sue credenziali) e spedisce le credenziali inserite dall'utente al server nella query string di chiave *RETURNURL*;

6. il server recupera con la chiave *RETURNURL* le credenziali di autenticazione, che valida con una query su una tabella di SQL Server. Il codice nella pagina di login crea un cookie che contiene un ticket di autenticazione forms settato per quella sessione. La classe **Membership** fornisce il metodo **ValidateUser** per questo scopo come nel seguito mostrato:

```
if (Membership.ValidateUser(userName.Text, password.Text))
{
   if (Request.QueryString["ReturnUrl"] != null)
   {
      FormsAuthentication.RedirectFromLoginPage(userName.Text, false);
   }
   else
   {
      FormsAuthentication.SetAuthCookie(userName.Text, false);
   }
}
else
{
   Response.Write("Invalid UserID and Password");
}
```

7. per l'utente autenticato il server Web redireziona il browser all'URL originale, specificato nella query string dal parametro *RETURNURL*, ed invia:

   302 Found Location:

   http://localhost/ecentral/detailed/diagnosticsdetailed.aspx
8. seguendo la redirezione il browser richiede nuovamente la pagina aspx *diagnosticsdetailed.aspx*; questa richiesta include il cookie di autenticazione form;
9. la classe **FormsAuthenticationModule** individual il cookie di autenticazione forms ed autentica l'utente. Se l'autenticazione è stata compiuta con successo, la classe **FormsAuthenticationModule** popola la proprietà **User** (contenente l'informazione dell'utente autenticato) dell'oggetto **HttpContext**;
10. una volta verificato il cookie di autenticazione, il server Web concede all'utente l'accesso e gli restituisce la pagina di aspx *diagnosticsdetailed.aspx*.

Si ricorda che la pagina aspx di login ha i controlli *RequiredFieldValidator* e *RegularExpressionValidator*, che permettono di inserire solo i caratteri permessi in un'espressione consentita nei campi *username* e *password.*

- **pages:** definisce i settaggi delle pagine (come i controlli .ascx), che possono essere sovrascritti con la direttiva Page. Nel seguito è mostrata una porzione della sezione <pages> del file web.config di eCentral:

```
<pages theme="default">
    <controls>
        <add tagPrefix="asp" namespace="System.Web.UI" assembly="System.Web.Extensions,
        Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
        <add tagPrefix="asp" namespace="System.Web.UI.WebControls"
        assembly="System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
        PublicKeyToken=31BF3856AD364E35"/>
        <add tagPrefix="dm" src="~/controls/deviceModel.ascx" tagName="deviceModel"/>
        <add tagPrefix="hGraphics" src="~/controls/history/historyGraphics.ascx"
        tagName="historyGraphics"/>
    …
    </controls>
</pages>
```

L'attributo "namespace" definisce la collezione di direttive importate da utilizzare durante la precompilazione.

– **httpHandlers:** definisce le classi che processano le richieste HTTP ricevute da eCentral. Le richieste per i file con estensione .aspx sono automaticamente gestite dalla classe *System.Web.UI.PageHandlerFactory*, che esegue il ciclo di vita di una pagina aspx. Il *PageHandlerFactory* non è in realtà un hanlder HTTP, ma è una classe factory che crea l'appropriato HTTP handler sulla base dell'informazione della richiesta. Il *PageHandlerFactory*, se necessario, invoca il sistema di compilazione ASP.NET, restituisce l'esatto tipo di handler corrispondente all'URL richiesto e, poi, ne crea un'istanza. Nel seguito viene mostrata una porzione della sezione <httpHandlers>:

```
<httpHandlers>
    <remove path="*.asmx" verb="*"/>
            <add path="*.asmx" verb="*" type="System.Web.Script.Services.ScriptHandlerFactory,
            System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
            PublicKeyToken=31BF3856AD364E35" validate="false"/>
            <add path="*_AppService.axd" verb="*"
            type="System.Web.Script.Services.ScriptHandlerFactory, System.Web.Extensions,
            Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"
            validate="false"/>
            <add path="ScriptResource.axd" verb="GET,HEAD"
            type="System.Web.Handlers.ScriptResourceHandler, System.Web.Extensions,
            Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"
            validate="false"/>
</httpHandlers>
```

Gli attributi utilizzati nell'elemento httpHandlers sono:

- **path:** specifica l'estensione del file;
- **verb:** specifica il tipo di richiesta, HTTP GET o POST (con la wildcard * si –considerano entrambi i modi);
- **type:** è la classe dell'handler HTTP; tale attributo si compone di 2 parti separate dalla virgola:
  1. il nome completo della classe;
  2. il nome dell'assembly DLL che contiene la classe;
- **validate:** settato a false indica ad ASP.NET di caricare la classe corrispondente solo quando giunge realmente la richiesta; potenzialmente la segnalazione dell'errore può essere ritardata, ma viene migliorato lo start–up time [21].

- **httpModules:** permette di definire le classi aventi la possibilità di reagire ad eventi di livello globale dell'applicazione. Il template di tale sezione è simile a quello precedentemente visto per l'httpHandler. Nel seguito viene mostrata una porzione della sezione <httpModules>:

```
<httpModules>
  <add type="[COM+ Class], [Assembly]" name="[ModuleName]" />
  <remove type="[COM+ Class], [Assembly]" name="[ModuleName]" />
  …
  <clear />
</httpModules>
<httpModules>
  <add name="ScriptModule" type="System.Web.Handlers.ScriptModule, System.Web.Extensions,
  Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
  …
</httpModules>
</system.Web>
```

- **runtime**: permette di definire l'affidabilità e le prestazioni dell' "application pool".

```
<runtime>
<assemblyBinding xmlns="urn:schemas–microsoft–com:asm.v1">
<dependentAssembly>
        <assemblyIdentity name="System.Web.Extensions" publicKeyToken="31bf3856ad364e35"/>
        <bindingRedirect oldVersion="1.0.0.0–1.1.0.0" newVersion="3.5.0.0"/></dependentAssembly>
<dependentAssembly>
        <assemblyIdentity name="System.Web.Extensions.Design"
        publicKeyToken="31bf3856ad364e35"/>
<bindingRedirect oldVersion="1.0.0.0–1.1.0.0" newVersion="3.5.0.0"/>
</dependentAssembly>
```

</assemblyBinding>

…

</runtime>

2.1.3.4.2.2 Sezione <appSettings>

In tale sezione sono definiti i settaggi di una qualunque pagina dell'applicazione eCentral. Si usa il tag <add> per aggiungere una variabile con un nome identificativo unico (key) ed un contenuto (value). Nel file Web.config di eCentral si specifica:

<appSettings>

<add key="dbServer"value="PRODESNA\SQLSERVER2005"/>

<add key="dbName" value="ECENTRAL"/>

<add key="dbUser" value="…"/>

<add key="dbPassword" value="…"/>

<add key="bypassLogin" value="True"/>

</appSettings>

L'informazione di interesse, contenuta nella sezione <appSettings>, viene recuperata nel codice di una pagina Web utilizzando la classe *ConfigurationSettings* (del namespace *System.Configuration*). Tale classe espone una proprietà chiamata **AppSettings**, contenente la collezione (costruita dinamicamente) dei settaggi disponibili per l'applicazione eCentral specificata all'interno della sezione <appSettings>.

Nel seguito viene mostrato un esempio di utilizzo della classe AppSettings.

Se in una pagina .aspx si vuole recuperare il nome del server DBMS, nel code behind di tale pagina occorre importare il namespace *System.Configuration* e nel metodo Page_Load, che viene invocato al caricamento della pagina, occorre scrivere:

```
protected void Page_Load (object sender, EventArgs e)
{
    String NomeServer = ConfigurationManager.AppSettings["dbServer"];
}
```

2.1.3.4.3 Viewstate

Una volta rispedita una pagina al server Web, ASP.NET riceve tutte le informazioni che l'utente ha inserito in ogni controllo <input> nel tag <form>, carica la pagina Web nel suo stato originale (basandosi sul layout e defaults definiti nel file .aspx) e aggiorna tale pagina in accordo con le nuove informazioni inserite e spedite dall'utente. Il server perde traccia dello stato in cui si trovavano i dati spediti in risposta al client, essendo gli oggetti deallocati dalla sua memoria ed il protocollo HTTP di tipo stateless. Per risolvere il problema

ASP.NET si avvale del *view state*. Al termine dell'esecuzione del codice della pagina (ma prima della traduzione in HTML e della spedizione al client) ASP.NET esamina le proprietà di tutti i controlli sulla pagina. Nel caso di modifica di una qualunque di queste proprietà rispetto al suo stato originale, ASP.NET si annota ogni modifica in una collezione di coppie nome–valore, che viene infine serializzata in una stringa Base64 (con la codifica Base64 si garantisce che non siano presenti caratteri speciali, eventualmente non validi in HTML). Tale stringa finale è inserita in un campo nascosto della sezione <form>. Ad una successiva spedizione della pagina al server, il motore ASP.NET in sequenza:

1. ricrea la pagina e gli oggetti controllo, mantenendo per la pagina lo stato che essa aveva nella sua prima richiesta, perché non conserva memoria dell'ultimo stato su cui ha lavorato;
2. deserializza l'informazione del view state e aggiorna tutti i controlli, riportando la pagina nello stato in cui si trovava prima di essere spedita in risposta al client;
3. ricostruito l'ultimo stato, aggiorna la pagina con le informazioni attuali ricevute dal client e, pertanto, alla fine del processo la pagina riflette lo stato corrente in cui appare al client;
4. scatena gli appropriati eventi, a cui il codice può reagire per servire le elaborazioni richieste dal client.

Il view state non permette di memorizzare l'informazione sensibile in modo sicuro, perché viene scambiata con il client, mentre dovrebbe rimanere sul server. Il viewstate può essere sottoposto ad un processo di hashing per prevenire manomissioni e criptato per prevenire una decodifica.

2.1.3.4.4 Gestione della sessione

La natura "non connessa" di Internet rende il rilascio di eCentral significativamente più complesso rispetto a quello di un'applicazione stand alone, in quanto nessuno stato è mantenuto tra una richiesta e l'altra dello stesso client. Il sistema software "eCentral" non include un server applicativo per monitorare lo stato d'esecuzione dell'applicazione.

Poiché il server Web può monitorare un elevato numero di utenti collegati, viene impostato per ciascuna sessione un un time out. Se l'utente non è attivo per un tempo superiore al timeout, il server si disconnette dal client che alloca inutilmente risorse senza chiedere servizi. Con lo stato della sessione il server memorizza l'informazione relativa alla navigazione di pagina in pagina di un utente, mentre con il viewstate di una sola pagina del sito. Ogni client, che accede all'applicazione, ha una differente sessione ed una distinta collezione di informazioni. In una pagina, la proprietà **Session** è deputata alla memorizzazione della sessione. ASP.NET associa ad ogni sessione un unico identificativo a 120–bit, di cui genera il valore usando un algoritmo proprietario, per garantire, da un punto di vista statistico, che esso sia univoco ed abbastanza casuale, impedendo agli utenti malevoli di determinarlo attraverso meccanismi di reverse engineering. Questo ID è solo un frammento dell'informazione trasmessa tra il client e il server Web. Quando il client presenta l'ID di sessione, ASP.NET cerca la sessione corrispondente, carica i dati serializzati dallo *state server*, li instanzia in oggetti, collocandoli in una collezione speciale a cui il codice può accedere. Questo processo avviene

automaticamente, mediante il modulo *SessionStateModule* nel namespace *System.Web.SessionState* della pipeline ASP.NET. Lo stato della sessione è reso persistente negli *state providers*, che sono delle classi che implementano l'interfaccia *IHttpSessionState*. Tale modulo genera l'ID di sessione, recupera i dati di sessione dagli *state provider* esterni e lega i dati al contesto della richiesta. Gli *state provider* sono mostrati in figura 2.19.

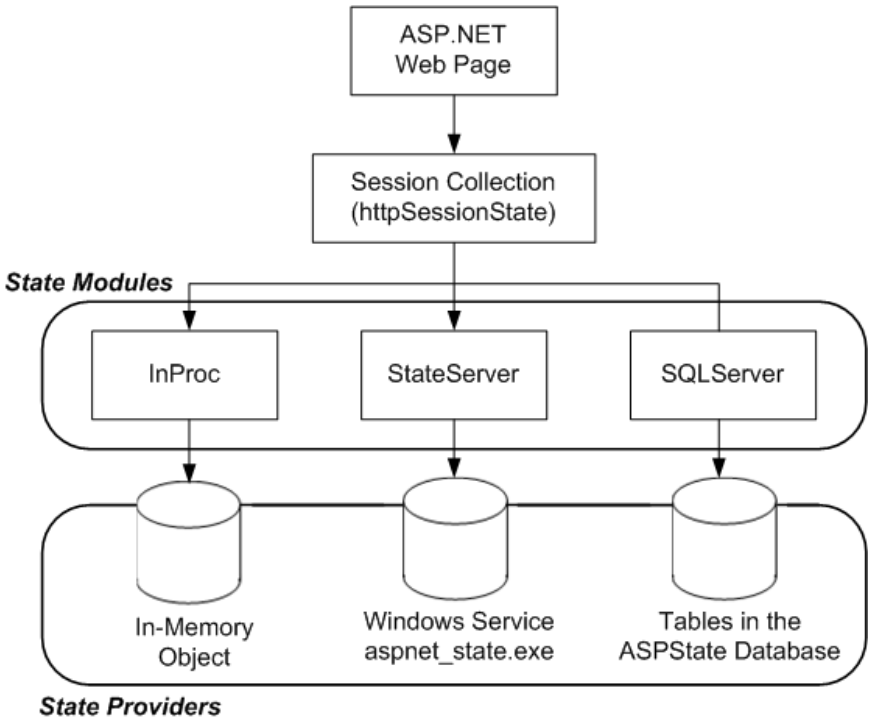


Figura 2.19: state provider

ASP.NET prevede tre *state provider*, che permettono di salvare l'informazione rispettivamente in un processo, in un servizio separato o nel database SQL Server. Affinché lo stato della sessione sia mantenuto, il client deve presentare l'appropriato ID di sessione ad ogni richiesta; per tenere traccia di tale ID tra una richiesta e l'altra si è scelto di utilizzare i cookie. L'ID di sessione è trasmesso in uno speciale cookie, denominato *ASP.NET_SessionID*, creato automaticamente ASP.NET quando viene utilizzata la collezione **Session**. Per interagire con lo stato della sessione da una pagina, occorre recuperare l'oggetto *Session* dalla collezione *Session["nomeOggetto"]* della classe *System.Web.SessionState.HttpSessionState*. Lo stato della sessione è globale per l'utente corrente dell'applicazione. Lo stato di sessione viene perso se:

1. l'utente chiude e riavvia il browser;
2. l'utente accede contemporaneamente alla stessa pagina con un differente browser, anche se la sessione continuerà ad esistere se si accede alla pagina attraverso il browser originale;
3. scade la sessione, a causa di un periodo di inattività maggiore della durata del timeout;

4. si termina la sessione richiamando *Session.Abandon()*;
5. il dominio applicativo viene ricreato, dopo l'aggiornamento dell'applicazione o la modifica della sua configurazione.

Nei primi due casi, in realtà, la sessione rimane nella memoria del server fino alla sua scadenza, ma risulta inaccessibile. Dovendo configurare eCentral per gestire lo stato della sessione, è stato utilizzato il codice nel seguito mostrato:

```
<sessionState
        mode="StateServer"
        stateNetworkTimeout="14400"
        stateConnectionString="tcpip=loopback:42424"
        timeout="10080"
        cookieless="UseCookies">
</sessionState>
```

In particolare nel seguito sono stati elencati gli attributi ed i valori ad essi assegnati:

- **Mode = "StateServer":** indica ad ASP.NET di utilizzare un servizio Windows separato per la gestione dello stato; anche se eseguito sullo stesso server Web, esso è caricato al di fuori del processo principale di ASP.NET, garantendo un livello di protezione base al riavvio del processo ASP.NET, ma con un incremento del ritardo nel traferimento dell'informazione di stato tra i due processi;
- **stateNetworkTimeout="14400":** specifica il numero di secondi di inattività (idle) della connessione di rete TCP/IP tra il server Web e quello di stato, prima che la richiesta sia cancellata;
- **stateConnectionString="tcpip=loopback:42424":** specifica al valore 127.0.0.1 l'indirizzo TCP/IP ed al valore 42424 la porta, richiesti per contattare lo *state server* ospitato dal server locale.
- **Timeout="10080":** specifica il numero di minuti di inattività (idle) di una sessione prima della sua scadenza;
- **Cookieless="UseCookies":** indica ad ASP.NET di utilizzare i cookie, anzichè la query string dell'URL. Se si disabilitano i cookie nel browser, non si può tenere traccia della sessione. Il cookie si dice *persistente* se viene conservato fino alla data ed ora specificata, mentre se è *non persistente*, il cookie viene scartato alla chiusura del browser.

2.1.3.4.5 Logging: Log4net

log4net è l'utility non a pagamento che fornisce all'applicazione eCentral il "log manager", con cui i log sono stati gestiti in due modi:

- in un semplice file di testo;
- in una tabella di SQL Server 2005.

Per integrare *log4net* nell'applicazione è stato necessario:

1. scaricare il suo assembly ed inserire nel progetto un suo riferimento;
2. configurarlo nel web.config (o app.config), aggiungendo nella sezione <*Configuration*><*Configsections*> la sezione:
   <*section name="log4net" type="log4net.Config.Log4NetConfigurationSectionHandler, log4net"*/>
   al cui interno è stata collocata la sezione di configurazione vera e propria di log4net, il cui codice viene mostrato nel seguito:

```
<log4net>
    <appender name="LogFileAppender"
        type="log4net.Appender.RollingFileAppender">
        <param name="File" value="C:\eCentral.log"/>
        <param name="AppendToFile" value="true"/>
        <param name="CountDirection" value="1"/>
        <param name="MaximumFileSize" value="25MB"/>
        <param name="MaxSizeRollBackups" value="50"/>
        <layout type="log4net.Layout.PatternLayout">
            <param name="ConversionPattern" value="%d [%t] %–5p %c
(%property{log4net:HostName}) – %m%n"/>
        </layout>
    </appender>
    <appender name="ECENTRAL_AdoNetAppender_SqlServer"
type="log4net.XDataNet.eCentral.eCentralAdoNetAppender">
        <bufferSize value="0"/>
        <connectionType value="System.Data.SqlClient.SqlConnection, System.Data,
Version=1.0.3300.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
        <connectionString value="Data Source=PRODESNA\SQLSERVER2005;integrated
security=false;persist security info=True;Initial Catalog=ECENTRAL;User ID=*;Password=*"/>
        <commandText value="INSERT INTO LogEvent (LogTime, LogLevel, LoggerName,
        HostName, PID, OpClass, ThreadID, Message, Exception) VALUES (@log_date, @log_level,
        @logger, @hostName, @PID, @OpClass, @thread, @message, @Exception)"/>
        <parameter>
            <parameterName value="@log_date"/>
            <dbType value="DateTime"/>
            <layout type="log4net.Layout.PatternLayout" value="%date{yyyy'–'MM'–'dd
HH':'mm':'ss'.'fff}"/>
        </parameter>
```

```
<parameter>
    <parameterName value="@log_level"/>
    <dbType value="String"/>
    <size value="5"/>
    <layout type="log4net.Layout.PatternLayout" value="%level"/>
</parameter>
<parameter>
    <parameterName value="@logger"/>
    <dbType value="String"/>
    <size value="128"/>
    <layout type="log4net.Layout.PatternLayout" value="%logger"/>
</parameter>
<parameter>
    <parameterName value="@hostName"/>
    <dbType value="String"/>
    <size value="64"/>
    <layout type="log4net.XDataNet.eCentral.eCentralHostnameLayout"/>
</parameter>
<parameter>
    <parameterName value="@PID"/>
    <dbType value="Int32"/>
    <size value="4"/>
    <layout type="log4net.Layout.PatternLayout" value="%message"/>
</parameter>
<parameter>
    <parameterName value="@OpClass"/>
    <dbType value="String"/>
    <size value="20480"/>
    <layout type="log4net.Layout.PatternLayout" value="%message"/>
</parameter>
<parameter>
    <parameterName value="@thread"/>
    <dbType value="Int32"/>
    <size value="4"/>
    <layout type="log4net.Layout.PatternLayout" value="%thread"/>
</parameter>
```

```
            <parameter>
                <parameterName value="@message"/>
                <dbType value="String"/>
                <size value="20480"/>
                <layout type="log4net.XDataNet.eCentral.eCentralMessageLayout"/>
            </parameter>
            <parameter>
                <parameterName value="@Exception"/>
                <dbType value="String"/>
                <size value="10000"/>
                <layout type="log4net.Layout.ExceptionLayout"/>
            </parameter>
        </appender>
        <!—<root>
    <level value="ALL"/>
  </root>—>
        <logger name="ECENTRAL_Log">
            <level value="ALL"/>
            <appender–ref ref="ECENTRAL_AdoNetAppender_SqlServer"/>
            <appender–ref ref="LogFileAppender"/>
        </logger>
    </log4net>
</configuration>
```

Un **<appender>** rappresenta la modalità di gestione del log ed ha una serie di parametri che ne consentono la personalizzazione. Con l'appender *LogFileAppender* si crea sotto la radice del file system *C:* il file di log denominato *eCentral.log*, che viene riempito fino al raggiungimento della dimensione massima di 25MB. Una volta raggiunta tale dimensione il file verrà rinominato "eCentral.log.1", successivamente "eCentral.log.2", …, fino ad un massimo di 50 file di backup (come specificato nell'elemento *<maxSizeRollBackups>*). Ciascuna linea di log ha il formato specificato nell'elemento **<layout>** e per ognuna di esse si è deciso di stampare: l'istante di tempo, il livello di severità, la classe chiamante (nome del logger nel file web.config), il nome dell'host, l'ID della Person, OpClass, l'ID del thread, il messaggio, l'eccezione.

Nell'elemento <ConversionPattern> del web.config, *(%property{**log4net:HostName**})* è una proprietà di built–in di log4net, come mostrato nel seguito:

*<param name="ConversionPattern" value="%d [%t] %–5p %c (%property{**log4net:HostName**}) – %m%n"/>*

La presenza dell'elemento <*appender–ref ref="LogFileAppender"*/> risulta indispensabile per l'esecuzione del logging sul file.

Con l'appender *ECENTRAL_AdoNetAppender_SqlServer* si utilizza ADO.NET per l'inserimento dei record di log nella tabella *LogEvent* nella base di dati attraverso la query mostrata nel seguito:

INSERT INTO LogEvent (LogTime, LogLevel, LoggerName, HostName, PID, OpClass, ThreadID, Message, Exception) VALUES (@log_date, @log_level, @logger, @hostName, @PID, @OpClass, @thread, @message, @Exception)

La presenza dell'elemento <*appender–ref ref="ECENTRAL_AdoNetAppender _SqlServer"*/> risulta indispensabile per l'esecuzione del logging su DB.

Nell'elemento **<root>** si specificano i livelli di criticità del log (DEBUG, INFO, WARNING, ERROR, FATAL) per i vari appender registrati; con <level value="ALL"> si sono specificati tutti.

Ad un oggetto di interfaccia ILog esposto da log4net sono associati più *appender* che, attivati con l'elemento <*appender–ref* … />, intercettano un'informazione di log, che viene memorizzata contemporaneamente nei formati corrispondenti (file, tabella DB, etc.).

3) per utilizzare log4net nell'applicazione, è stato creato un wrapper di comodo, denominato **eCentralLogger**, con cui è stato customizzato l'oggetto di interfaccia ILog di log4net, come mostrato nel seguito:

Costruttore di eCentralLogger

```
private static ILog _log = null;
public static ILog log
{
    get
    {
    if (_log == null)
    {
    // Log4Net configuration
    // Si recupera il file web.config
    FileInfo fi = new FileInfo(AppDomain.CurrentDomain.SetupInformation.ConfigurationFile);
    // Se viene creato l'oggetto corrispondente al file web.config…
    if (fi.Exists)
    // Con l'oggetto XmlConfigurator, si legge la configurazione di log4net nell'omonimo elemento del file
    // web.config
    XmlConfigurator.Configure(fi);
    else
    {
            // Se non c'è il file web.config si procede ad una configurazione base di lof4net
```

```
            BasicConfigurator.Configure();
    }

// si recupera l'oggetto ILog con il metodo GetLogger
_log = LogManager.GetLogger(eCentralConstants.LoggerNames.eCentralLog);
}
return _log;
}
}
```

Se necessario, si richiama uno dei cinque metodi di **eCentralLogger** in funzione del corrispondente livello di criticità, che si intende registrare, utilizzando il codice mostrato nel seguito:

```
eCentralLog.Debug("Operazione eseguita")
eCentralLog.Info("Operazione eseguita al secondo tentativo")
eCentralLog.Warn("Operazione eseguita con errori");
eCentralLog.Error("Operazione non completata");
eCentralLog.Fatal("Il Sistema remoto non risponde");
```

Ciascun metodo consente di accettare, oltre al messaggio di testo, anche un oggetto *Exception* che viene accodato al testo.

2.1.4 Descrizione ed implementazione del second tier

2.1.4.1 ECD

L'architettura denominata "Engine Control Devices" (ECD) si basa su di una suddivisione in due strati interagenti:

- il primo costituito dai moduli ChannelManager, deputati all'acquisizione e alla spedizione di dati rispettivamente da e verso i device su di uno specifico canale di comunicazione;
- il secondo costituito dai moduli Driver, che garantiscono l'interoperabilità tra i device e l' "eCentralCore" fornendo l'astrazione di una lingua comune tra le parti.

I dati, per raggiungere l' "eCentralCore", seguono un percorso minimo (due soli step: acquisizione e traduzione), garantendo al tempo stesso sia elevate prestazioni che un'efficiente autonomia di gestione delle due fasi. I moduli presentano un elevato grado di coesione, ovvero tutte le operazioni in essi contenuti hanno lo stesso scopo e non è ragionevole scinderli in altri moduli. Essi, inoltre, sono caratterizzati da una **coesione orientata ai dati**, poiché i metodi fanno riferimento ad una precisa struttura dati. I moduli presentano anche un buon grado di accoppiamento; si parla in tal caso di **accoppiamento per passaggio di controllo**, in quanto i Driver richiedono i ChannelManager per l'acquisizione/invio dei dati, mentre i ChannelManager richiedono i Driver per la traduzione. L'isolamento dei due strati è in ogni caso buono poiché il malfunzionamento di un ChannelManager X utilizzato dal Driver Y non compromette l'esecuzione del Driver Y (non lo blocca), infatti esso può continuare a servire altri ChannelManager. Occorre, però, attendere il ripristino del ChannelManager X per poter riprendere l'acquisizione/invio dei dati sul canale, di cui è competente. L'interazione tra ChannelManager e Driver definisce il pattern del flusso d'esecuzione all'interno dell'ECD, per cui la scalabilità risulta essere un pò limitata. La presenza di un processo master all'interno di ogni ChannelManager garantisce il bilanciamento del carico di lavoro, permettendo di condividerlo in modo flessibile fra più thread slave. L'architettura può essere, inoltre, facilmente estesa aggiungendo nuovi moduli Driver o ChannelManager, sfruttando le interfacce comuni Driver2Core e Driver2Channel. La procedura di eliminazione o sostituzione di moduli di entrambi gli strati gode del medesimo vantaggio offerto da tali interfacce comuni. L' "eCentralCore" tramite i metodi pubblicati dall'interfaccia "IChannelManager":

1. inizializza i thread in ascolto con il metodo di firma: *bool InitListeningThread()*;
2. controlla lo stato di salute dei thread in ascolto
   con il metodo di firma: *bool CheckListeningThreadVitality()*;
3. inizializza i thread "postini", Message Dispatch Thread
   con il metodo di firma: *bool InitMessageDispatchThread()*;
4. controlla lo stato di salute del thread "postino", Message Dispatch Thread,

con il metodo di firma: *bool CheckMessageDispatchThreadVitality()*;

5. arresta l'esecuzione del thread "postino", Message Dispatch Thread
   con il metodo di firma: *bool StopMessageDispatchThread()*;
6. arresta l'esecuzione del thread principale
   con il metodo di firma: *bool StopChannelManager()*.

I Driver espongono:

- l'interfaccia *Driver2Channel* ai ChannelManager,
- l'interfaccia *Driver2Core* all' "eCentralCore".

L'architettura è mostrata nella figura 2.20:

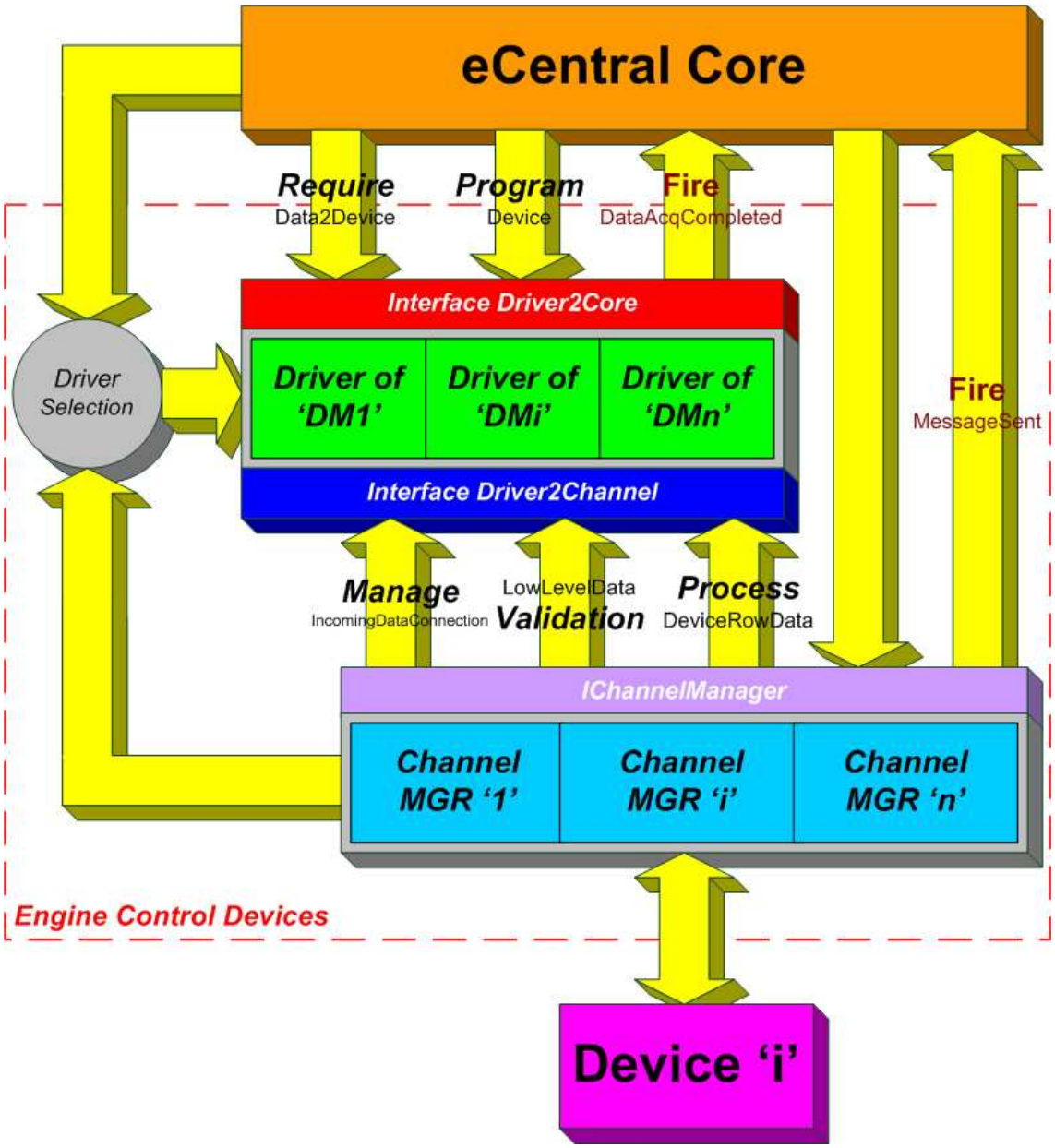


Figura 2.20: ECD

Si procede ora alla descrizione nel dettaglio dell'ECD nelle due differenti tipologie di comunicazione: *device2eCentral* e *eCentral2device*.

Nota

Nel seguito verrà utilizzato l'avverbio "concettualmente" per meglio comprendere la dinamica di interazione dei moduli dell'ECD. Nel modello di esecuzione proposto in figura si potrebbe pensare erroneamente ad un trasferimento inefficiente della medesima struttura dati ***pLowlevelData*** (della classe ***LowLevelData***) dal ChannelManager al Driver per i servizi di "Validation" e "Process", in quanto il Driver potrebbe conservarla. In realtà tale inefficienza non sussiste, in quanto l'interazione tra oggetti avviene per scambio di messaggi, con il vincolo per un oggetto di fornire i propri servizi (pubblicati nell'interfaccia dell'oggetto e, quindi, visibili al chiamante) a qualunque altro oggetto di cui ne conosce l'indirizzo. In questo modo si garantisce un accesso e un'elaborazione controllata dei propri attributi interni. Poiché ***pLowlevelData*** è un oggetto (della classe ***LowLevelData***), esso viene passato per riferimento all'invocazione dei servizi "Validation" e "Process". Essendo tale riferimento (dell'oggetto pLowLevelData) una copia di quello originale, ogni modifica al riferimento copia non ha effetto su quello originale, ma ogni modifica agli attributi dell'oggetto ***pLowlevelData*** risulta visibile al ChannelManager chiamante, perché i servizi sono richiesti all'oggetto puntato dal riferimento originale.

2.1.4.1.1 Comunicazione device2eCentral

Un processo in esecuzione su un device ad un certo istante di tempo stabilito dal firmware (o su sollecitazione) procede a spedire i dati acquisiti attraverso l'opportuno endpoint (identificato ad esempio da: SIM del modem GSM utilizzato, porta + indirizzo IP, etc.) del canale di comunicazione scelto. Il mittente deve conoscere a priori l'endpoint di ricezione di eCentral, che rappresenta il punto di ingresso dei dati comunicati dal device. Se la trasmissione, ad esempio, avviene per mezzo del protocollo TCP/IP, il processo client sul device deve conoscere a priori la porta su cui il processo server presta il servizio di "acquisizione dati TCP" per poter usare tale porta per comunicare. Nel caso, invece, di una comunicazione GSM, il processo ricevente (lato eCentral) deve eseguire il polling sulla batteria di modem per individuare quello su cui il device ha fatto la chiamata dati o a cui ha inviato l'SMS. In tal caso il device mittente è configurato per spedire i dati ad un endpoint di destinazione che è una precisa SIM di un certo modem gestito da eCentral. Gli endpoint hanno una natura differente a seconda del tipo di canale utilizzato per effettuare la comunicazione (GSM, TCP/IP, etc.) e, pertanto, sono sotto la giurisdizione del ChannelManager competente. Nonostante la gestione dei dati in ingresso da parte di eCentral sia affidata a differenti ChannelManager in funzione del tipo di canale di comunicazione, tale gestione viene svolta secondo una struttura standard, per garantire la medesima efficienza. Un thread principale rimane in ascolto delle richieste e, fungendo da dispatcher, ne assegna la gestione ad un thread slave specifico. Il ChannelManager si limita, quindi, ad acquisire i dati in strutture concettuali (SMS, file XML, etc.) senza doversi preoccupare di come trattarle. Per poter acquisire correttamente i dati (cioè eventi, misure), occorre interpretare la loro struttura (protocollo Edor, etc.) avvalendosi di una precisa semantica. Tale

servizio viene svolto dall'entità Driver, di cui ne esiste una per ogni DeviceModel. Tramite un meccanismo di selezione, indicato nell'architettura con **Driver Selection**, si individua e si recupera il Driver corrispondente. Il thread slave del ChannelManager, invocando "**ManageIncomingDataConnection**", trasferisce "concettualmente" (vedi nota) al Driver la struttura dati acquisita per essere prima identificata e successivamente analizzata. Il Driver provvede a riconoscere la struttura dati utilizzata nella comunicazione e la tipizza in quello che effettivamente è di basso livello, ad es. un SMS non è altro che una stringa di testo, mentre una socket sono dati binari, etc. Successivamente restituisce "concettualmente" (vedi nota) i dati "campionati" in una struttura standard denominata "LowLevelData". Per maggiore chiarezza viene mostrata la classe "LowLevelData" serializzabile:

```
[Serializable]
public class LowLevelData
{
    public byte[] BinaryData;
    public string TextData;
    public FileInfo FileInfo;
}
```

Il thread slave del ChannelManager può ora rilasciare le risorse impegnate, ad esempio il modem, poiché i dati grezzi sono stati riconosciuti dal Driver con successo. È bene rilasciare il lock del modem il prima possibile, a causa della sua limitata disponibilità. Il thread slave, successivamente, può richiedere al Driver con "**LowLevelDataValidation**" (perché dipende dal device model) di eseguire un servizio di validazione dei dati ricevuti per poter individuare la presenza di eventuali errori di trasmissione (ad es. con un controllo ciclico di ridondanza). Il Driver, poi, restituisce l'esito del controllo al ChannelManager, che attraverso la chiamata al servizio "**ProcessDeviceRowData**", restituisce "concettualmente" al Driver la struttura dati di tipo "LowLevelData", per essere interpretata e tradotta in qualcosa di comprensibile da parte di "eCentralCore". Al termine del servizio di analisi sarà restituito un valore booleano per indicarne l'esito. Nel caso di successo le singole misure o eventi con i rispettivi valori saranno state identificate. I dati forniti all'eCore hanno una struttura comune, più precisamente sono oggetti della classe "DataAcqCompleteArgs" che viene mostrata di seguito:

```
[Serializable]
public class DataAcqCompletedEventArgs : EventArgs
{
    public Communication Communication;
    public bool CommunicationSuccessful;
}
```

La comunicazione tra ChannelManager e Driver è di tipo procedurale con passaggio "concettuale" (vedi nota) della struttura dati ad ogni fase di elaborazione (acquisizione, validazione, processo). Tale approccio è molto utile, poiché si evitano corse critiche se la stessa ed unica istanza di un dato Driver deve servire due richieste di acquisizione per lo stesso Driver contemporaneamente. Per *sezione critica* s'intende una porzione di codice che può essere eseguita da un solo processo o thread alla volta. Per

regolare l'accesso alla sezione critica è necessario un meccanismo di sincronizzazione (ad esempio un semaforo) all'entrata ed all'uscita del codice *critico*. Entrando nella sezione critica questo meccanismo verifica che non vi sia alcuna esecuzione del codice critico da parte di un processo. In caso affermativo l'esecuzione prosegue e il processo corrente prende possesso della sezione critica, per rilasciarlo all'uscita. In caso contrario il processo può attendere che la sezione critica si liberi oppure eseguire nel frattempo un altro compito. L'accesso di più processi ad una sezione critica ha, normalmente, una priorità regolata da una coda FIFO [http://it.wikipedia.org/wiki/Sezione_critica]. Poiché le richieste non accedono alle variabili del Driver, non si hanno problemi di accesso concorrente, per cui il Driver è *thread-safe*. Inoltre le due esecuzioni (in quanto due sono le richieste) avvengono in contesti differenti, nonostante le richieste condividano lo stesso codice per i tre metodi (acquisizione, validazione, elaborazione).

Quando il Driver ha terminato l'elaborazione deve "risvegliare" l' "eCentralCore" con un evento scatenato da **"FireDataAcqCompleted"**. Poiché gli eventi non possono essere creati in un'interfaccia, **"FireDataAcqCompleted"** non è presente nell'"Interface Driver2Core". Ogni Driver per poter invocare **"FireDataAcqCompleted(**DataAcqCompletedEventArgs evento**)"** eredita la classe "DriverBase" del namespace *XDataNet.CPL.eCentral.DataModel.Driver*. Tale classe è mostrata nel seguito:

```
public class DriverBase
{
    #region Events
    /// <summary>
    /// Data Acquisition Completed event
    /// </summary>
    /// L'evento eDataAcqCompleted
    public event dataAcqCompletedHandler eDataAcqCompleted;
    /// deve essere gestito dal metodo "handler"
    public delegate void dataAcqCompletedHandler(object sender, DataAcqCompletedEventArgs e);
        /// Quando un Driver invoca "FireDataAcqCompleted" passando come
        /// parametro un oggetto della classe Communication...
    protected void FireDataAcqCompleted(DataAcqCompletedEventArgs e)
    {
       ///se l'evento eDataAcqCompleted è diverso da null, cioè è stato scatenato (e deve ///essere gestito
       da dataAcqCompletedHandler) significa che il metodo ///FireDataAcqCompleted è stato invocato
       da un Driver. Il problema è quindi ///gestire tramite un metodo (dataAcqCompletedHandler) un
       altro metodo ///(FireDataAcqCompleted, che rappresenta l'evento quando invocato).
        if (eDataAcqCompleted != null)
       ///se l'evento è stato scatenato, si passa a chi gestisce, cioè al metodo
       ///dataAcqCompletedHandler, colui che ha scatenato l'evento (Driver) e l'oggetto
       ///Communication
            ///Con "this" (nell'oggetto creato da questa classe) si fa riferimento ad una ///proprietà di sè,
            che è il chiamante.
```

```
            eDataAcqCompleted(this, e);
    }

    ///analogamente per il FireMessageSent
    /// <summary>
    /// Enqueued message has been sent
    /// </summary>
    public event EventHandler eMessageSent;
    public delegate void messageSentHandler(object sender, MessageSentEventArgs e);
    protected void FireMessageSent(MessageSentEventArgs e)
    {
        if (eMessageSent != null)
            eMessageSent(this, e);
    }

    #endregion Events
}
```

L'evento "eDataAcqCompleted" è associato al metodo "**eCentralServer_eDataAcqCompleted**" tramite il delegato "dataAcqCompleteHandler". Quando si scatena l'evento "eDataAcqCompleted" (ovvero viene invocato il metodo "FireDataAcqCompleted"), il metodo "**eCentralServer_eDataAcqCompleted**" attende i parametri spediti con "eDataAcqCompleted(this, e)".
La firma del metodo "**eCentralServer_eDataAcqCompleted**":
*void **eCentralServer_eDataAcqCompleted** (object sender,*
XDataNet.CPL.eCentral.DataModel.Device.DataAcqCompletedEventArgs e)
deve esattamente essere uguale a quella del delegato "dataAcqCompletedHandler":
public delegate void dataAcqCompletedHandler(object sender, DataAcqCompletedEventArgs e);
in quanto il delegato (puntatore a funzione) gestisce le funzioni di cui è competente, cioè che hanno la sua stessa firma.
Con il metodo "**eCentralServer_eDataAcqCompleted**" l' "eCentralCore" gestisce l'evento "eDataAcqCompleted", con cui un Driver segnala il completamento della traduzione. Tale metodo ha i seguenti parametri:

1. "object sender": riferimento del Driver;
2. "XDataNet.CPL.eCentral.DataModel.Device.DataAcqCompletedEventArgs ***e***": oggetto "Communication" (per ulteriori informazioni vedi capitolo 3).

e controlla che il campo "CommunicationSuccessful" di '***e***' sia settato a true per verificare che la comunicazione sia stata formattata correttamente dal Driver.
Se l'esito del controllo è positivo, si richiede al "CommunicationManager" di rendere persistente la comunicazione. Il "CommunicationManager" è il responsabile della gestione della tabella

"Communication" nella base di dati; in tale tabella sono memorizzate tutte le comunicazioni avvenute da e verso "eCentralCore".
L'esito dell'operazione viene mostrata in una console creata appositamente per il debug, utilizzando il metodo "Invoke":
*Invoke(new UpdateConsoleLogDelegate(**ConsoleLog**), sb.ToString());*
Con il metodo *ConsoleLog* si scrive su console.

Breve descrizione di Invoke

Al delegato "UpdateConsoleLogDelegate" viene passato il metodo "ConsoleLog", che viene eseguito all'interno del thread a cui appartiene un determinato user control della console di debug (GUI). Quando si lavora in multithread può essere necessario l'aggiornamento di un user control da parte di un thread, anche se c'è un thread principale che crea tutti i controlli grafici. Per consentire l'accesso concorrente di un thread diverso da quello principale ad un determinato user control si utilizza il metodo "Invoke" nel Framework .NET 2.0. Come si può notare, il metodo "Invoke" permette di aggiornare l'interfaccia della console di debug, in quanto sarebbe molto limitativo non consentire ad un thread diverso da quello principale di poter aggiornare la GUI.

L' "eCentralCore" è il solo ad essere autorizzato a memorizzare la comunicazione ed i dati in essa contenuti, per cui l'inserimento di un record nella tabella "Communication" avviene contemporaneamente all'inserimento dei record corrispondenti ai dati ricevuti da "eCentralCore" nella tabella "CommunicationData".
L' "eCentralCore" può svolgere altri compiti in virtù della ricezione di altri specifici eventi, oltre a compiere il salvataggio della comunicazione associata all'invocazione di "FireDataAcqCompleted". Occorre implementare l'evento di interesse (e il metodo che lo rappresenta) ed l'event handler corrispondente nel Driver.
Occorre disabilitare tutti gli handler collegati al Driver, se lo si vuole deregistrare, come descritto nel seguente esempio di codice:
DriverPool.GetDriverInstancebyDRID(6, eCentralConstants.BuiltinPrincipals.Admin.PID, null).DriverBase.eDataAcqCompleted -= eCentralServer_eDataAcqCompleted;

2.1.4.1.1.1 Scenario di identificazione

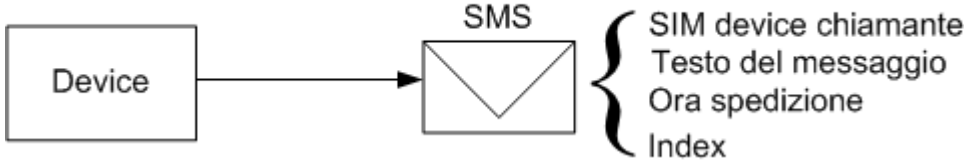


Figura 2.21: invio SMS

Come mostrato in figura 2.21, si suppone che un device invii un SMS ad un certo modem di ricezione di "eCentral". Recuperato l'SMS, si esegue la query sulla basi di dati per determinare il device mittente. Poiché nell'ER (vedi capitolo 3) si modella il legame tra il device e il GSMEndPoint come in figura 2.22:

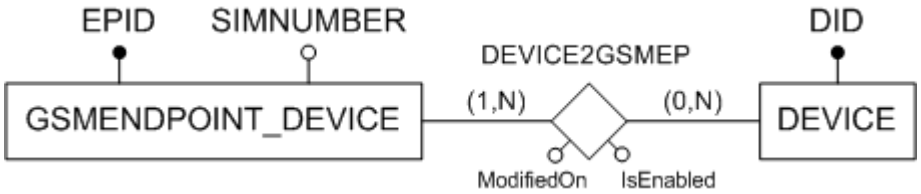


Figura 2.22: porzione di ER

la seguente query:

SELECT D2G.DID
FROM DEVICE2GSMEP as D2G
WHERE D2G.EPID = @EPIDMittente
AND IsEnabled=1 -- per individuare quella utilizzata dal device (da non confondere con isEnabled
-- dell'EndPoint che indica se la SIM è attiva)

potrebbe potenzialmente restituire molti device e non uno solo (poiché max-card(GSMENDPOINT_DEVICE, DEVICE2GSMEP=N)). In tal caso dal numero della SIM chiamante non si potrebbe risalire all'unico device che l'ha utilizzata per effettuare la chiamata. In realtà a livello fisico sussiste il vincolo che un device utilizzi un'unica SIM per comunicare come mostrato in figura 2.23.

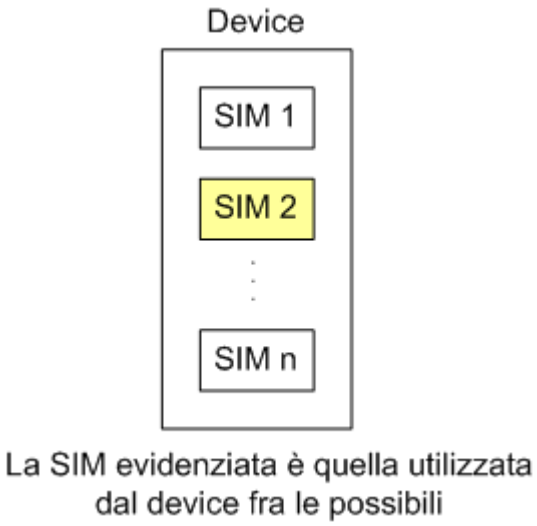


Figura 2.23: device-SIM

Il RDBMS, però, permette di inserire per uno stesso device più SIM, per consentire una maggiore libertà d'azione nella gestione dei legami tra SIM e device. Nell'inserimento di tali legami occorre prestare molta attenzione al settaggio dell'attributo "IsEnabled", affinché un device non sia più univocamente identificabile. Se si esprime tale vincolo nello schema logico con una relazione del tipo:

Device (DID, EPID, isEnableDevice, isEnabledEP, ...)
AK: EPID
il RDBMS garantisce mediante la SIM l'identificazione del device, ma non si potrebbe rappresentare il caso di un device che possiede due o più SIM, perché non si può replicare il *DID* (essendo chiave primaria). Una volta individuato l'unico device (DeviceID) associato alla SIM-chiamante contenuta nell'SMS, seguendo il percorso definito dalle relazioni della base di dati si raggiunge il Driver corretto, deputato ad interpretare l'SMS.

2.1.4.1.2 Comunicazione eCentral2Device

L'attività di programmazione è resa più semplice dal fatto che "eCentralCore" conosce il Driver del device, per cui non è necessaria una fase preliminare di identificazione del Driver. Un Driver, a sua volta, sa quale ChannelManager utilizzare in funzione del device da programmare ed essere competente di più canali, in quanto conosce quello da utilizzare, potendo risalire all'endPoint corrispondente nella base di dati, attraverso l'utilizzo dell'identificatore del device. Se, ad esempio, l'Edor Driver autoritativo per il device model Edor supporta due tipi di comunicazione (GSM e FTP), è possibile risalire al canale di comunicazione utilizzato dal device e, quindi, al ChannelManager competente, ricercando il tipo di endPoint utilizzato per la comunicazione. Se il device comunica con un endPoint GSM (come staticamente registrato nella tabella DEVICE2GSMEP nella base di dati), il Driver, al termine dell'incapsulamento dei dati, ne verifica la corretta esecuzione (checkMessageProof) rispettando la semantica del protocollo ed inserisce il pacchetto risultante nel "buffer persistente" LogSMS (tabella della base di dati). Poiché i record di tale tabella corrispondono a degli SMS, il thread "postino" del GSMChannelManager (competente per la tabella) invoca il metodo "SendSMS" (per ulteriori informazioni vedi capitolo 4) per spedire correttamente il pacchetto in coda. Se un device utilizza più canali di comunicazione, si può indicare quello che usa di default attraverso un apposito attributo (ad es. creando una RepositoryProperty specifica. Vedi capitolo 3 per ulteriori informazioni) a cui fa riferimento il Driver per scegliere in quale struttura incapsulare i dati. Il thread "postino" genera un evento con "FireMessageSent" per segnalare all' "eCentralCore" la conclusione dell'attività di spedizione. Si ricorda che la programmazione richiede il lock del device.

2.1.4.2 Middleware ADO.NET

2.1.4.2.1 Struttura del componente ADO.NET

ADO.NET è un insieme di oggetti che interagiscono fra loro per svolgere la funzione di accesso ai dati. Le classi per la gestione dei dati si trovano in una serie di *namespaces* appartenenti al namespace

principale *System.Data* all'interno del *Framework.NET*. La figura sottostante mostra una vista semplificata degli oggetti che compongono ADO.NET ed i loro legami.

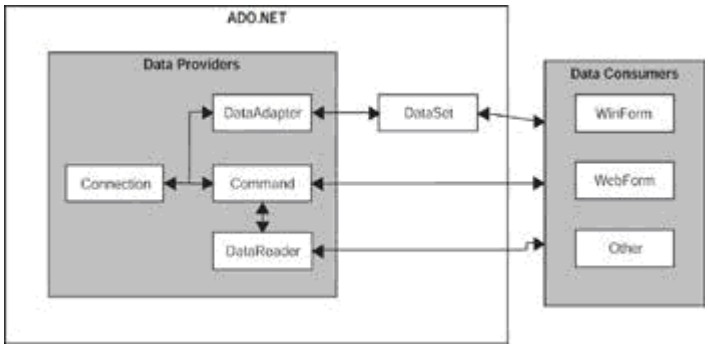


Figura 2.24: Classi ADO.NET

ADO.NET è composto da due componenti fondamentali:

1. il **Data Provider,** denominato anche **Managed Provider**, che mantiene la connessione con la sorgente dati fisica;
2. il **DataSet**, che rappresenta i dati attuali, ovvero i dati su cui si sta lavorando.

Entrambi i componenti possono comunicare con i **Data Consumers**, ovvero i fruitori dei loro servizi, come, ad esempio una *WinForm* o una *WebForm*.

2.1.4.2.2 ADO.NET Data Providers

È disponibile un **Data Provider** specifico per ogni fonte dati (data source). Un Data Provider è un insieme di classi ADO.NET che permette di accedere ad uno specifico database, di eseguire comandi SQL e di recuperare i dati. L'architettura ADO.NET 2.0 prevede quattro provider:

1) **SQL Server provider:** fornisce l'accesso ottimizzato a SQL Server;
2) **OLE DB Provider:** fornisce l'accesso a qualunque data source che ha un driver OLE DB (OLE = Object Linking Exchange);
3) **Oracle Provider:** fornisce l'accesso ottimizzato a SQL Server;
4) **ODBC Provider:** fornisce l'accesso a qualunque data source che ha un driver ODBC.

Quasi tutti i provider elencati contengono le stesse classi, che si differenziano per il nome (ad esempio: **SQLConnection**, **OleDbConnection**, **OracleConnection**, ecc.) e per alcune proprietà e metodi. In figura 2.25 di pag. 91 vengono mostrati tali Provider [23].

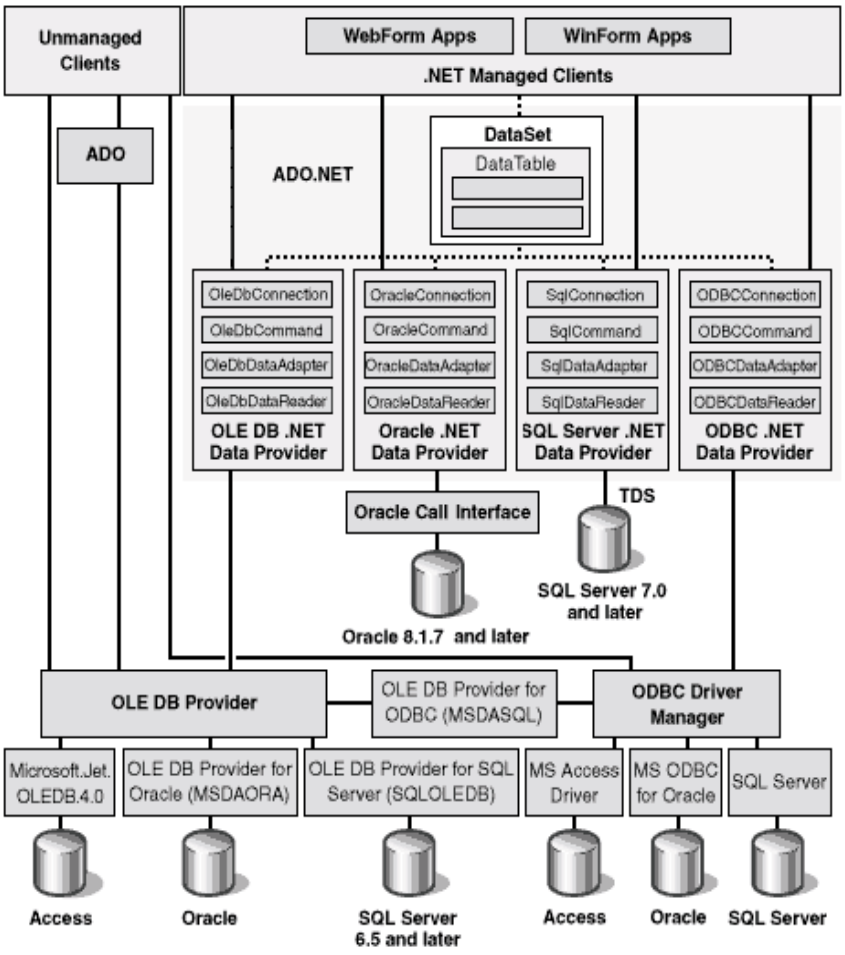


Figura 2.25: Provider

3 PROGETTO DI DETTAGLIO

In tale capitolo vengono descritte la struttura della solution eCentral sviluppata nel linguaggio di programmazione C# e il modello della base di dati.

3.1 Descrizione della solution eCentral

Per lo sviluppo di eCentral è stato adottato un approccio like BCE (Boundary-Control-Entity: frontiera, controllo, entità), cioè rivolto alla modellazione ad oggetti basata su una ripartizione delle classi in tre categorie concettuali distinte:

1. **classi di presentazione (Boundary):** descrivono gli oggetti che rappresentano l'interfaccia tra un attore ed il sistema. Esse rappresentano una parte dello stato del sistema che è presentata all'utente in forma visiva. Gli oggetti Boundary sono dipendenti dagli oggetti entità, tramite alcuni oggetti Control. Gli oggetti Boundary spesso persistono al di là della singola esecuzione del programma;
2. **classi di controllo (Control):** descrivono gli oggetti che percepiscono gli eventi generati dall'utente e controllano l'esecuzione di un processo di business. Essa rappresenta un supporto per le azioni e le attività di un caso d'uso. Gli oggetti di controllo spesso non persistono oltre l'esecuzione del programma. Gli oggetti Control trattano le interazioni tra gli eventi generati dall'utente, gli oggetti Entity e Boundary interessati. In particolare, ciascun oggetto Boundary, che rende possibile un'interazione, deve avere un oggetto Control associato;
3. **classi di sistema (Entity):** descrivono gli oggetti che rappresentano la semantica delle entità nel dominio applicativo, definendolo. Essa corrisponde ad una struttura dati nel database del sistema. Gli oggetti Entity rendono note le variazioni del loro stato per permettere agli oggetti della classe Boundary di aggiornarne la visualizzazione. Gli oggetti Entity sono sempre persistenti nella base dati e possono partecipare a molti casi d'uso.

In una gerarchia di package ben progettata un attore può interagire solo con gli oggetti del package Boundary; gli oggetti del package Entity possono interagire solo con gli oggetti nel package Control i quali, a loro volta, possono interagire con qualsiasi oggetto.

Si è scelto di adottare questo approccio di raggruppamento delle classi in strati gerarchizzati per migliorare la comprensibilità del modello, riducendone la complessità. Diminuisce, inoltre, il rischio di creare "classi mostro", cioè con troppi servizi, che hanno il controllo dell'intera funzionalità del sistema. Per ottenere buon progetto object-oriented l'intelligenza del sistema eCentral è stata uniformemente distribuita tra molte classi.

Nel rispetto dell'applicazione dell'approccio BCE-like ad eCentral, la Solution è stata suddivisa nei seguenti package:

1. ***DataModel (Entity):*** contiene gli oggetti applicativi classificati in sotto-package logici;
2. ***BusinessLogic (Control + Database):*** contiene la logica applicativa del programma ("cosa fa"), i moduli dell'ECD e lo strato di legame tra eCentral (nel ruolo di client) e il DBMS (nel ruolo di

server). In tale package si annoverano i sottopackage <*CanaleDiComunicazione*>*Manager* (a <CanaleDiComunicazione> si sostituisce GSM, FTP, Mail, TCP) e *Driver*, le classi *Manager* responsabili del controllo delle tabelle del DB (metodi di estrazione, ricerca, lettura e scrittura) e le classi specifiche di elaborazione dei dati applicativi classificate in sotto-package logici;

3. ***Web-site (http://localhost/ecentral/):*** contiene tutte le risorse per la gestione di eCentral a livello web (inerente le pagine aspx), ovvero per la visualizzazione delle informazioni su browser compatibilmente con la funzione applicativa;
4. ***eCentralCore:*** contiene informazioni utili a tutti gli altri progetti che costituiscono la solution eCentral;
5. ***eCentralService:*** il servizio MS Windows da attivare corrispondente all'ECD;
6. ***eCentralWinForm (debug):*** rappresenta la simulazione del comportamento del servizio eCentralService, tramite un'apposita interfaccia grafica, al fine di poter svolgere facilmente l'attività di debug.

È bene precisare che il package **DataModel** descrive le classi Entity nella memoria di programma dell'applicazione e non nel BDMS (quest'ultimo non è object-oriented ma relazionale, perciò gli oggetti persistenti del business sono memorizzati come record di tabelle). Per trattare la corrispondenza tra le classi Entity e le tabelle relazionali nel DB e per gestire altre rilevanti informazioni sulla struttura della basi di dati, è necessario uno strato distinto di classi chiamato **BusinessLogic**. Il package **DataModel** fornisce "un modello d'esecuzione del business", memorizza i dati che il programma ottiene dal database invocando servizi (metodi) della **BusinessLogic**, implementati nelle classi Manager. I nomi delle classi del package DataModel, che gestiscono gli "oggetti di business" nella memoria del programma, corrispondono ai nomi di tabelle nel database. Ogni classe del package DataModel ha la sua classe Manager per interagire con il database. Le classi Manager sono responsabili:

1. della memorizzazione persistente degli oggetti di una classe del package DataModel in record dell'omonima tabella del database (insert, update);
2. dell'estrazione dei record da mappare su istanze della corrispondente classe del package DataModel (select);
3. dell'eliminazione dei dati dal database (delete).

Le classi Manager definiscono, quindi, un'interfaccia che isola l'applicazione eCentral, intesa come client, dal meccanismo di memorizzazione persistente (R-DBMS). Tale interfaccia può fornire un accesso indipendente dal R-DBMS (SQL Server, Oracle, MySQL, DB2) agli "oggetti" persistenti, incapsulando il linguaggio di programmazione del database, ovvero SQL. Il package **BusinessLogic** fornisce, pertanto, un livello di indirezione tra l'applicazione ed il database, per cui la logica applicativa è indipendente dalle modifiche ai dati. La **logica d'integrità**, responsabile per le regole di business a livello aziendale, alle quali eCentral deve essere conforme, viene implementata mediante trigger sotto la responsabilità del DBMS.

Un ulteriore vantaggio dell'approccio BCE è la sua corrispondenza con l'architettura fisica descritta nel capitolo 2, in cui si distingue la gestione dei dati (entity) dalla presentazione (boundary), attraverso un livello intermedio dedicato alla logica applicativa (control).

3.1.1 BusinessLogic e ADO.NET

Poiché il linguaggio SQL è ospitato (embedded) dal linguaggio di programmazione, in cui sono scritte (C#) le classi Manager, è necessario un pre-compilatore (pre-processore) per tradurre le istruzioni SQL in chiamate a funzioni nella libreria del DBMS SQL Server 2005. Si è scelto di far comunicare eCentral (che assume il ruolo di client) con il database attraverso le librerie standard (Provider .NET) di ADO.NET (ActiveX Data Objects), che è il componente per l'accesso ai dati del Framework .Net di Microsoft. ADO.NET fornisce un linguaggio standard per il database, che viene tradotto in SQL nativo dal Provider competente per SQL Server 2005, disaccoppiando in questo modo eCentral dal DBMS. Nel caso di migrazione di eCentral, in futuro, su un diverso DBMS (Oracle, DB2, MySQL), per garantirne l'interoperabilità sarà sufficiente sostituire il Provider in uso con il nuovo. eCentral in questo modo risulta estremamente flessibile per quanto concerne l'interoperabilità con DBMS diversi.

3.1.2 Connection Pooling

Per non limitare la scalabilità di eCentral, attraverso l'utilizzo del costrutto "using" si permette ad eCentral (tramite i Manager) di usufruire in modo trasparente del Connection Pooling, che mantiene un set permanente di connessioni aperte condivise da sessioni che usano lo stesso data source. Quando un Manager richiede una connessione, essa viene fornita direttamente dal pool disponibile e non si deve ricrearla. Nel momento del rilascio da parte del Manager la connessione torna nel pool per servire la prossima richiesta. Quanto descritto è mostrato nelle seguenti righe di codice C#:

Classe di un Manager

```
…
string dbConnectionString =
ConfigurationManager.ConnectionStrings[eCentralConstants.ConnectionStringName].ToString();
using (SqlConnection dbConn = new SqlConnection(dbConnectionString))
{
    …
}
```

3.1.3 ConfigurationManager

Per ogni query di ogni Manager, la classe **ConfigurationManager** fornisce l'accesso al file di configurazione per l'applicazione. Il metodo **ConnectionStrings** recupera il "System.Configuration.ConnectionStringSection" per la configurazione di default della dell'applicazione eCentral. Il **ConfigurationManager** con il metodo "ConnectionStrings" recupera l'informazione contenuta nella sezione <connectionStrings>:

- dal file **web.config**, se il contesto corrente di esecuzione è il code behind di una pagina .aspx, cioè se il contesto è il sito web;
- oppure dall'**app.config** se il contesto corrente di esecuzione è il servizio MS Windows (ECD).

Nel seguito è mostrato un esempio di procedura di recupero:

```
string dbConnectionString =
ConfigurationManager.ConnectionStrings[eCentralConstants.ConnectionStringName].ToString();
public const string ConnectionStringName = "eCentralConnectionString";

//per recuperare la stringa di connessione dalla sezione <connectionStrings>
public static ConnectionStringSettingsCollection ConnectionStrings { get; }
```

In eCentral, attraverso la procedura descritta, uno stesso Manager può essere utilizzato in modo dinamico sia per il sito Web che per il servizio MS Windows (ECD), in quanto si appoggia sul file di configurazione corretto a seconda del contesto.

3.2 Progettazione della base di dati

Per progettare una base di dati occorre definirne la struttura, le caratteristiche ed il contenuto. Si tratta di un processo delicato che richiede particolare rigore ed attenzione; l'utilizzo di opportune metodologie è indispensabile per la realizzazione di un prodotto di alta qualità.

Per la modellazione dei dati di eCentral si è seguita la metodologia articolata nelle seguenti tre fasi principali effettuate in cascata:

1. **progettazione concettuale (cosa rappresentare?):** il suo scopo è quello di rappresentare le specifiche informali della realtà di interesse attraverso una loro descrizione formale e completa, ma indipendente dai criteri di rappresentazione utilizzati nei sistemi di gestione delle basi di dati. Il prodotto di questa fase viene chiamato *schema concettuale* e fa riferimento ad un *modello concettuale* dei dati, che permette di descrivere l'organizzazione dei dati ad un alto livello di astrazione, senza tenere conto degli aspetti implementativi, cioè delle modalità con le quali queste informazioni sono codificate in eCentral e dell'efficienza di tale sistema.
2. **progettazione logica (come rappresentarle?):**consiste nella traduzione dello schema concettuale, definito nella fase precedente, nel modello di rappresentazione dei dati adottato dal DBMS SQL Server 2005. Il prodotto di questa fase viene denominato *schema logico* della basi di dati e fa riferimento ad un *modello logico* dei dati. Come noto, un modello logico consente di descrivere i dati secondo una rappresentazione ancora indipendente da dettagli fisici, ma concreta, perché disponibile nel DBMS SQL Server 2005. In questa fase le scelte progettuali si basano, tra l'altro, su criteri di ottimizzazione delle operazioni da effettuare sui dati. Per la verifica della qualità dello schema logico ottenuto è stata utilizzata la tecnica della normalizzazione e, precisamente, lo schema logico è in terza forma normale;
3. **progettazione fisica:** in questa fase lo schema logico viene completato con la specifica dei parametri fisici di memorizzazione dei dati (organizzazione dei file e degli indici). Il prodotto di questa fase viene denominato *schema fisico* e fa riferimento ad un *modello fisico* dei dati. Tale modello dipende da SQL Server 2005 e si basa sui suoi criteri di organizzazione fisica dei dati.

3.2.1 Lo schema concettuale (ER)

La costruzione dello schema concettuale è stata compiuta con l'intento di garantire le proprietà che uno schema concettuale di buona qualità deve possedere e precisamente:

- **correttezza:** uno schema concettuale è corretto quando si utilizzano in modo appropriato i costrutti messi a disposizione dal modello concettuale di riferimento;
- **completezza:** uno schema concettuale è completo quando rappresenta tutti i dati di interesse e quando tutte le operazioni possono essere eseguite a partire dai concetti descritti nello schema. La completezza dello schema ER è stata verificata controllando che tutte le specifiche sui dati siano rappresentate da qualche concetto presente nello schema e che tutti i concetti coinvolti in un'operazione presente nelle specifiche siano raggiungibili "navigando" attraverso lo schema;

- **leggibilità:** uno schema concettuale è leggibile quando rappresenta i requisiti in maniera naturale e facilmente comprensibile. Per rendere lo schema concettuale più leggibile si è proceduto a:
  - collocare al centro le entità più importanti e con più legami;
  - minimizzare le intersezioni;
  - disporre le entità che sono genitori di generalizzazioni sopra le relative entità figlie;
- **minimalità:** uno schema concettuale è minimale quando tutte le specifiche sui dati sono rappresentate una sola volta nello schema. Uno schema non è minimale, quindi, quando esistono delle ridondanze, ovvero concetti che possono essere derivati da altri. Una possibile fonte di ridondanza in uno schema ER è la presenza di cicli nelle relazioni e/o generalizzazioni. A differenza delle altre proprietà, non sempre una ridondanza è indesiderata, ma può nascere da precise scelte progettuali. Queste situazioni sono state documentate in modo rigoroso.

Nello schema ER non sono stati indicati tutti gli attributi di ogni entità per maggior chiarezza; l'elenco completo degli attributi è nel Dizionario dei dati riportato di seguito. Le informazioni di tutte le entità e delle associazioni sono rese significative e complete attraverso il riferimento all'entità *Person*, utilizzando l'attributo *ModifiedBy*, che contiene numeri identificativi di persone corrispondenti alla chiave primaria dell'entità *Person*. I riferimenti sono significativi in quanto i valori nelle entità ed associazioni sono uguali a valori effettivamente presenti nell'entità *Person*. Nel caso in cui valore di *ModifiedBy* non comparisse nelle entità ed associazioni corrispondenti come valore della chiave di *Person*, allora il riferimento non sarebbe efficace. Se sul *ModifiedBy* non fosse specificato un vincolo di integrità referenziale (tra tabelle), per il quale il valore di *ModifiedBy* deve essere il valore della chiave primaria di un'istanza presente nell'anagrafica *Person*, non si potrebbe stabilire se è una persona censita in anagrafica chi modifica l'entità o l'associazione. Per semplificare ulteriormente l'aspetto grafico dell'ER è stata introdotta una tecnica adottata nel disegno elettronico e di seguito spiegata. Se un'entità è referenziata da un alto numero di altre entità, con il simbolo sotto riportato:

si indica l'altro estremo dell'associazione. Sul simbolo è specificata la cardinalità di partecipazione e:

- la foreign key e l'entità destinazione per quella sorgente;
- l'entità sorgente per quella destinazione.

EffectiveSince con ModifiedOn rappresenta: l'istante a partire dal quale si deve ritenere EFFETTIVO ("VALIDO") il cambio di stato del sistema, rappresentato dall'inserimento del nuovo record o dalla modifica del record già esistente.

EvaluateRecorsively: supponiamo di considerare una relazione di inclusione fra gruppi: 'A' contiene 'B', 'C', 'D'. Se il record GID='A' MemberGID='B' ha EvaluateRecorsively=1 dovrò andare ad esplorare i figli di 'B'.

I)

IntegrationTag: per identificare univocamente una RepositoryProperty indipendentemente dalla sua descrizione in lingua. Per l'importazione dei dati.

Es.

Considero un oggetto device che ha un proprietà di nome SERIAL NUMBER (mappata come RepositoryProperty nel modello dati).

Un software esterno, inviando la struttura dati che modella un device (file XML, CSV) con tale RepositoryProperty, la identifica con SR_NM. Se mettessi SR_NM nel campo ShortDescr associato a quella RepositoryProperty per il parsing, non troverei mai la corrispondenza. Aggiungo, quindi, alla RepositoryProperty di device su DB il campo IntegrationTag, in cui inserisco SR_NM, lasciando SERIAL NUMBER nel campo di nome ShortDescr. In questo modo il match è assicurato sull'IntegrationTag. La RepositoryProperty (SERIAL NUMBER) può essere chiamata potenzialmente nel modo in cui si desidera, mantenendo sempre il match sui dati grezzi (la struttura dati può specificare la RepositoryProperty con un nome variabile SR_NM, SN), poichè la RepositoryProperty ha il "criterio di parsing" in un campo differente di nome IntegrationTag. Visualizzo su GUI la ShortDescr anzichè l'IntegrationTag (discorso analogo dell'hostname e indirizzo IP); scelgo un nome concettualmente più facile da ricordare e da utilizzare per la GUI ed analogamente scelgo una sigla più concisa per il parsing.

IsClassifiedProperty: indica una proprietà privata che può vedere solo una persona interna di CPL, non un cliente.

IsDeleted: stesso significato di IsRevoked, ma per le proprietà (si preferisce dire "cancello una proprietà")

IsEnabled: per rendere disponibile o meno l'entità.

Es. Suppongo di dover creare un gruppo di cui devo specificare determinate proprietà; tali proprietà, che posso associare al gruppo, hanno IsEnabled a true.

Se, IsEnabled è false, l'entità non viene visualizzata in GUI.

Nel caso degli EndPoint: supponendo di considerare una trasmissione GSM, le SIM di GSMEndPoint_MGR sono quelle possedute da eCentral, quindi contattate dai device. Qualora la Telecom bloccasse una SIM lato eCentral, occorre sostituirla tempestivamente con un'altra, affinchè tutti i device, che trasmettono dati verso quel GSMEndPoint, non siano impossibilitati a comunicare i propri dati acquisiti. Con l'attributo IsEnabled si specifica se un certo GSMEndPoint è attivo o meno.

IsKPI: ogni dato elaborato non è un KPI, ma lo sono quelli con KPI=true.

IsMandatory: indica se la proprietà è obbligatoria; ciò è utile per il controllo dell'input da GUI;

IsRevoked: indica la revoca di una relazione;

IsSystem: indica se la RepostioryProperty è scolpita nel codice.

IstanceN: qualora l'entità abbia valori multipli, ne specifica il numero. Supponiamo di considerare la RepositoryProperty di esempio "BATTERIA" per un device. Se un device può operare con 3 diversi modelli di batterie:

IstanceN=1 ValueI=1.5 V per la RepositoryProperty "BATTERIA" per un dato device.

IstanceN=2 ValueI=3 V per la RepositoryProperty "BATTERIA" per un dato device.

IstanceN=3 ValueI=5 V per la RepositoryProperty "BATTERIA" per un dato device.

L)

LDAPattName: nome dell'attributo LDAP corrispondente alla RepositoryProperty mappata nell'active Directory LDAP.

M)

ManageSilently: se vale true, indica che l'utente non può intervenire in alcun modo sull'entità (non è permessa l'interazione con l'utente) a cui si riferisce tale attributo, poichè è gestita in modo automatico dall'applicazione (importazione automatica da altro SW, etc.). ManageSilently ha lo stesso effetto video di IsEnabled, ma sono semanticamente molto diversi.

MaxValuesN: indica il numero massimo della RepositoryProperty in esame relativa ad un certo QualifiableObject. Es. il QualifiableObject Group può avere al più 3 RepositoryProperty GroupType, dunque il record Property2QO che lega i due record avrà l'attributo MaxValuesN=3. Se MaxValueN è null, non si hanno limiti, mentre con 'uno' specifico "a singolo valore". Questo attributo permette di sapere se una data RepositoryProperty è multivalore.

ModifiedOn senza EffectiveSince rappresenta:

1. l'istante di tempo ottenuto dalla lettura attuale del RTC locale al DBMS in cui l'informazione che rappresenta il cambio di stato del sistema viene effettivamente "inserted o updated nel DB";
2. l'istante da cui si deve ritenere EFFETTIVO ("VALIDO") il cambio di stato del sistema, rappresentato dall'inserimento del nuovo record o la modifica del record già esistente.

ModifiedOn con EffectiveSince rappresenta: l'istante al tepo di lettura attuale del RTC locale al DBMS in cui l'informazione che rappresenta il cambio di stato del sistema viene effettivamente "inserted o updated nel DB".

O)

OverrideFormatString: coincide con il parametro *IFormatProvide* che deve essere passato al metodo *toString(IFormatProvider)* per ottenere la desiderata formattazione dell'entità nella stringa in funzione del tipo di dato (es. per avere un float con 5 cifre decimali, si invoca sull'oggetto float toString(".5"). Il campo OverrideFormatString vale .5.

R)

RelevanceFactor: criterio di raggruppamento, cioè permette di specificare il criterio di raggruppamento (per rendere dinamico il GROUP BY).

S)

ShowInGrid: rende disponibile la visualizzazione in GUI dell'entità in esame nella colonna in una griglia personalizzabile

Status: attributo a soli 3 valori che descrive lo stato di un pacchetto: spedito, accodato, error.

ShowInWidget: flag che indica se si vuole mostrare la proprietà in widget.

ShowInWidgetDetail: flag che indica se si vuole mostrare la proprietà in "widget detail" cioè cliccando, ad esempio, su "more" (dettaglio) di un widget.

T)

Type (per HL/LLSignal): attributo a soli 4 valori: misura, evento, oggetto di programmazione.

U)

Uom: indica l'unità di misura

V)

ValueSource: indica il suffisso corrispondente al valueA, valueF, valueT, etc. a cui si fa riferimento; per esempio, è A se si riferisce a valueA. Utile per query di concatenazione stringhe.

3.2.1.4 Scelte di design

3.2.1.4.1 Generali

Sapendo che lo schema ER con associazione è più semplice e sintetico di quello equivalente con reificazione, nell'ER la reificazione è stata adottata solo quando necessario, cioè per esprimere regole di vincolo.

Poichè nello sviluppo di un progetto software complesso si deve tenene conto del problema dell'estensibilità, occorre valutare se sia meglio modellare un concetto di interesse come attributo di un'entità oppure creare un'entità ad hoc in vista di un suo ampliamento futuro. Un aspetto importante nella gestione dei dati è certamente la flessibilità e la manutenzione, affinchè la modifica di dati sia facile da eseguire e non avvenga fuori controllo, cioè rimanga sempre confinata a quei dati, senza ripercussioni su altri. Quando non si conosce a priori e si vuole controllare il numero di istanze di un concetto collegato ad un'entità conviene creare, per maggiore "pulizia di progettazione", un'entità ad hoc per quel concetto e non modellarlo come attributo. In questo modo la creazione di una nuova istanza si traduce semplicemente nell'inserimento di un nuovo record e non nella creazione di una nuova colonna della tabella che rappresenta nel modello logico l'entità in esame.

Supponendo di considerare il caso dell'entità HighLevelSignal, si potrebbe rappresentare correttamente l'attributo IALID (lingua: italiano, inglese, etc.) come in figura 3.1:

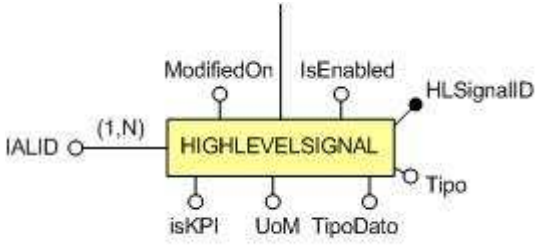


Figura 3.1: HighLevelSignal

con N=numero elevato non noto a priori (es. 20).

Se si vuole garantire una certa flessibilità nella gestione del numero di lingue per l'internazionalizzazione (non conoscendolo a priori), specificare un'integrità referenziale, monitorare il numero di lingue disponibili, conviene rendere visibile questa esigenza anche a livello ER. Una traduzione di questo tipo a livello logico darebbe luogo, inoltre, ad una tabella HighLevelSignal con un numero di colonne soggetto ad una dilatazione o contrazione "fuori controllo".

HLSignalID	Tipo	Uom	...	IALID1	IALID2	IALID3	...	IALID N

Un HLSignal può non sfruttare tutte le N colonne di IALID e potrebbe essere tradotto, per esempio, in 3 lingue su N, con N-3 colonne null.

Nel modello proposto nella tesi le modifiche, le aggiunte e le eliminazioni di IALID sono gestite in modo separato dall'entità HighLevelSignal e si traducono nella modifica, aggiunta ed eliminzione di record di una tabella ad hoc, evitando di modificare ogni volta lo schema dell'entità HighLevelSignal. Lo schema di HighLevelSignal rimane immutato con una sola colonna che referenzia I18NAbstractionLayer, in cui avvengono i cambiamenti, consentendo "una maggiore pulizia" e di evitare ridondanze.

Se nel caso dell'entità DeviceModel si rappresentasse l'attributo Brand come in figura 3.2:

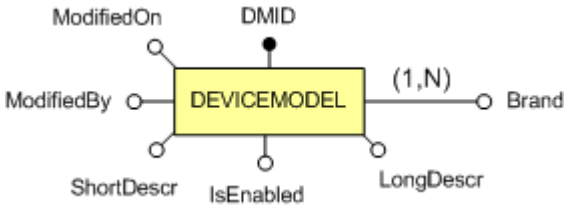


Figura 3.2: DeviceModel

si commetterebbe un errore, in quanto Brand non è un attributo multiplo; un DeviceModel, infatti, non può avere più Brand e, pertanto, una rappresentazione corretta è quella di figura 3.3:

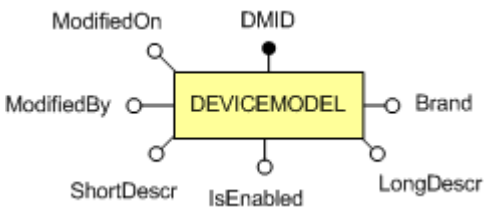


Figura 3.3: DeviceModel

con l'attributo Brand che può assumere 1,2, 3, … N valori. Poiché DMID è la chiave primaria di

DeviceModel, uno stesso device model non si può ovviamente ripetere e, pertanto, ha un solo valore di brand fra gli N del suo dominio.
Poichè Brand non è descritto solo dal nome, ma anche da altri attributi, si deve necessariamente costruire un'entità ad hoc.

3.2.1.4.2 Scelta degli identificatori principali

La scelta degli identificatori principali è essenziale nella traduzione dall'ER al modello relazionale in quanto in questo modello le chiavi vengono usate per stabilire i legami tra dati in relazioni diverse.
Dovendo scegliere una chiave primaria, è preferibile farla coincidere con l'identity per evitare di dover imporre che ogni attributo, facente parte della chiave primaria, sia non null (il valore null può avere uno dei due seguenti significati: "valore al momento sconosciuto" oppure "valore non applicabile"). Anche una chiave alternativa non accetta valori null, perché la sua definizione richiede l'applicazione dei vincoli UNIQUE e NOT NULL. Poiché UNIQUE non implica automaticamente NOT NULL, l'utilizzo del vincolo UNIQUE è la soluzione da adottare per risolvere il problema. Gli attributi di tipo datetime (come EffectiveSince, ModifiedOn, CreatedOn, dataTime, LogTime) sono sempre null all'inserimento/update del record corrispondente ad una certa tabella. Non si passa, infatti, mai come parametro l'indicazione del valore dell'istante di tempo, in quanto agli attributi di tipo datetime viene assegnato il valore del RTC del DBMS con un trigger (gli attributi datetime hanno sempre il valore dell'istante di tempo attuale del DBMS). Quando un tale attributo deve essere incluso nella chiave primaria, di cui non può far parte, è specificato il vincolo UNIQUE. Quando detto può essere spiegato con il seguente esempio. Si suppone di inserire un record nella relazione Driver2DeviceModel. Chi inserisce il record può specificare o meno (NULL) un valore per EffectiveSince, ma nel momento in cui si dà esecuzione all'istruzione il DBMS esegue il trigger si seguito:

```
ALTER TRIGGER [dbo].[Driver2DeviceModel_InserUpdate]
   ON [dbo].[Driver2DeviceModel]
   AFTER INSERT, UPDATE
AS
BEGIN
        SET NOCOUNT ON;

        declare @date as datetime
        set @date = getdate()
        update [dbo].[Driver2DeviceModel] set EffectiveSince = @date
        where DR2DMID in (select DR2DMID from INSERTED)
END
```

con cui sovrescrive il valore di EffectiveSince con il suo RTC.
Si è scelto di utilizzare l'identity come chiave primaria anche per non dover usare un elenco troppo lungo di attributi nella tabella interna (versus tabella esterna per concetto di foreign key). Un identificatore, infine, composto da uno o, comunque, da pochi attributi è sempre da preferire ad identificatori costituiti da molti attributi. Questo approccio, infatti, garantisce che le strutture ausiliarie create per accedere ai dati (gli indici) siano di dimensioni ridotte, permettendo un risparmio di memoria nella realizzazione dei legami logici tra le varie relazioni e facilitando le operazioni di join. Si propone il seguente esempio:

RPropertyEnabling(GPEID, GPID, EvaluetedOnQO, GPAVCode, DCID, ControlledGPID, ControlledQO, ForceMandatory, OverrideCanBeUpdate)
AK1: GPID, GPAVCode, EvaluetedOnQO, ControlledGPID, ControlledQO
AK2: DCID, ControlledGPID, ControlledQO

La relazione, che la referenzia, è:
RPropertyValueEnabling(ControllerGPEID, ControlledGPID, ControlledGPAVCode)
FK: ControllerGPEID references RPropertyEnabling

Se, invece, si eliminasse l'identity come primary key, la relazione sarebbe:
RPropertyEnabling(GPID, EvaluetedOnQO, GPAVCode, DCID, ControlledGPID, ControlledQO, ForceMandatory, OverrideCanBeUpdate)

La relazione interna, invece, sarebbe:
RPropertyValueEnabling(GPID, EvaluetedOnQO, GPAVCode, ControlledGPID, ControlledQO, ControlledGPID, ControlledGPAVCode)
FK: GPID, EvaluetedOnQO, GPAVCode, ControlledGPID, ControlledQO references RPropertyEnabling

Conclusione: si avrebbero 5 colonne al posto di una sola.

3.2.1.4.3 Internazionalizzazione

Alcune entità sono localizzate, cioè l'oggetto del mondo reale che rappresentano richiede di essere tradotto nella lingua del Paese che utilizza eCentral, ad esempio: "sistema di odorizzazione" in lingua italiana diventa "Odorizing System" in lingua inglese. Il marchio, invece, non ha la foreign key IALID, perchè il marchio è conosciuto con il suo nome in tutto il mondo, cioè non si traduce ad esempio Microsoft. Poichè sono diverse le entità soggette ad internazionalizzazione, ciascuna di queste deve essere messa in relazione con CULTURE, attraverso un'associazione binaria. Seguendo questo primo approccio, la traduzione standard prevede, poi, che ogni associazione sia tradotta con una relazione con gli stessi attributi, cui si aggiungono le chiavi primarie di tutte le entità che essa collega. In partcolare essendo la max-card(Ei, R)=N la chiave della relazione che rappresenta l'associazione è composta dall'insieme di tutte le chiavi primarie delle entità collegate. Le chiavi primarie delle entità collegate saranno, poi, chiavi straniere riferite alle corrispondenti entità. Considerando l'esempio di figura, si avrebbe:
APPLICATIONOBJECTI18N_1 (ID1, CUID, ShortDescr, LongDescr, IsEnabled, ModifiedOn, ModifiedBy)
FK: ID1 references ENT_I18N_1
FK: CUID references CULTURE
Questa scelta di progettazione, però, sfocerebbe nella creazione di un numero di tabelle di legame tra entità da tradurre e lingua, che potrebbe crescere in modo non controllato, perchè non si conosce a priori il numero di entità soggette ad internazionalizzazione (avrei N APPLICATIONOBJECTI18N). Un

approccio più efficiente è introdurre un livello di indirezione, denominato I18NabstractionLayer, affinchè il legame con CULTURE sia effettuato attraverso di esso, con la creazione di sole 2 tabelle (una I18NABSTRACTIONLAYER e un'unica APPLICATIONOBJECTI18N), come proposto nell'approccio 2. Ogni entità da tradurre ha una foreign key IALID che referenzia l'I18NABTRACTIONLAYER, per mapparsi su CULTURE. Quanto descritto è mostrato in figura 3.4.

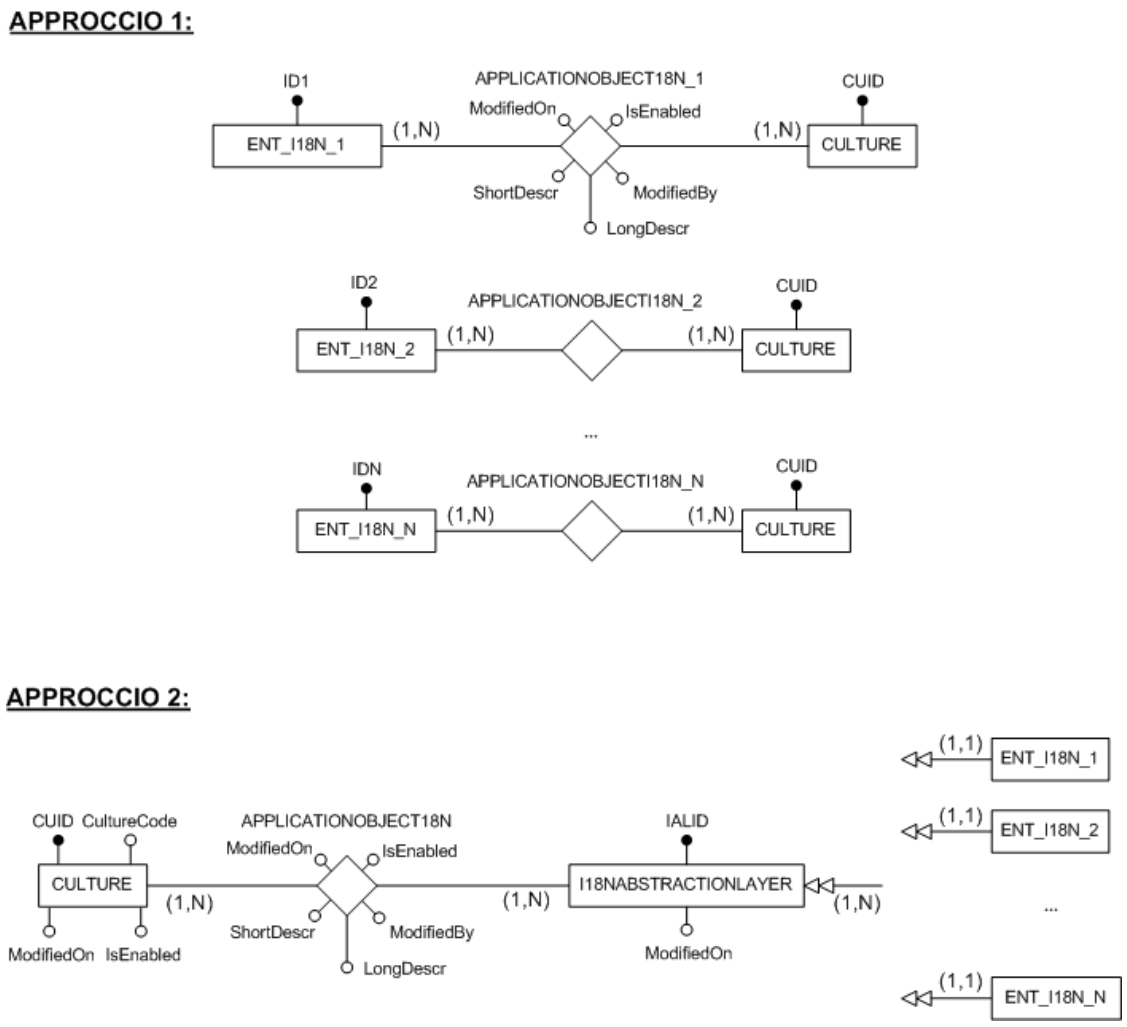


Figura 3.4: internazionalizzazione

3.2.1.4.4 Storicizzazione

Poiché i record di alcune tabelle del database cambiano nel tempo, occorre registrare la loro evoluzione temporale per conoscere il loro stato passato. Nello svolgimento della tesi sono state utilizzate le tabelle mostrate nell'ER, nelle quali sono registrati lo stato attuale e quello passato, ma attraverso un'attività di

ricerca si è ottenuto un modello più efficiente di gestione della storicizzazione. Si considera una tabella storicizzata, come D2DRShipElem, che viene scomposta nelle due tabelle:

1. D2DRShipElement, che contiene solo lo stato attuale;
2. D2DRShipElementHistory, che contiene lo storico.

mostrate in figura 3.5 per maggiore chiarezza:

Figura 3.5: storicizzazione

Tutti i record contenuti in D2DRShipElement sono validi, pertanto è stato possibile eliminare l'attributo "isRevoked". L'attributo EffectiveSince non fa più parte della primary key di D2DRShipElement che, essendo lo snapshot attuale di un dato tipo di relazione, ne contiene l'ultimo stato valido. L'attributo EffectiveSince deve, invece, far parte della chiave della tabella D2DRShipElementHistory per identificarne univocamente i record. Su D2DRShipElement si utilizza:

1. un trigger
   a. in Update per aggiornare un dato record e inserirlo prima della modifica nella tabella storica D2DRShipElementHistory;
   b. in Delete per impedire la rimozione dei record;
2. una stored procedure, a cui si passa la PK e il PID di chi effettua l'operazione, per rimuovere i record dalla tabella attuale e inserirli in quella storica con i campi DeletedBy e DeletedOn valorizzati.

Nella tabella D2DRShipElement la chiave impedisce l'inserimento di informazioni duplicate.

3.2.1.4.5 ER1

3.2.1.4.5.1 QualifiableObject

La promozione (elezione) di un'entità (concetto della realtà da modellare) a QualifiableObject permette di ottenere enormi vantaggi in termini di potenzialità della rappresentazione di tale entità, che sono nel seguito elencati:

1. con poco codice si realizza la logica di INSERT, UPDATE, DELETE su DB;
2. l'entità, che è stata promossa a QualifiableObject, acquisisce un sistema di qualificazione (caratterizzazione/descrizione a livello di schema), che va a potenziare quello ottenuto dalla struttura statica della sua tabella (classica/tradizionale modellazione di un'entità a livello logico-relazionale). I vantaggi, che ne derivano, consistono nella possibilità di associare al QualifiableObject una/più

RepositoryProperty (modularità) con già “scolpiti” nel codice tutti i metodi di interazione con il DB, nel disporre di RepositoryProperty multivalore, i cui valori od esse stesse possono essere eventualmente abilitate rispettivamente dalle RPropertyValueEnabling o dalle RPropertyEnabling;

3. si possono applicare filtri molto generici con i quali è possibile determinare le RepositoryProperty applicate ad un certo QualifiableObject.

Quelle entità che hanno particolari esigenze di flessibilità di gestione a livello di schema saranno promosse a QualifiableObject.

Attualmente i QualifiableObject sono:

1. Person;
2. Group;
3. Device;
4. Device Model;
5. Device Schema;
6. Device Class;
7. Activity;
8. Brand;
9. High Level Signal.

Per conoscere la/e RepositoryProperty associata/e ad un QualifiableObject si utilizza la tabella Property2QO. Per conoscere il valore della/e RepositoryProperty si utilizza la tabella <QualifiableObject>PropertyValue (a <QualifiableObject> si deve sostituire il QualifiableObject di interesse).

3.2.1.4.5.2 RepositoryProperty

Le RepositoryProperty sono associate solo ai QualifiableObject.

Si motiva l’esigenza di creare l’entità RepositoryProperty nei seguenti 3 punti:

1. rimuovere da un’entità QualifiableObject quegli attributi che non sono fondamentali alla sua caratterizzazione per evitare che il numero dei suo attributi e, quindi, delle sue colonne nel modello logico diventi eccessivamente elevato;
2. per gestire in modo centralizzato attributi comuni a più entità che attingono da questo repository;
3. qualora non si disponesse di un repository di proprietà, una modifica apportata allo schema di una o più entità attuata con l’aggiunta di uno o più attributi (equivalente all’aggiunta di uno o più colonne alla tabella corrispondente) si ripercuoterebbe sul codice. Si dorvrebbero modificare tutte le porzioni di codice che operano (direttamente o indirettamente) sulla/e tabella/e modificata/e. Con il repository, invece, si aggiunge semplicemente il record (nuovo attributo) alla tabella RepositoryProperty senza intaccare lo schema di una/più entità e di conseguenza una/più porzioni di codice.

Gli attributi che, invece, contraddistinguono un’entità (la differenziano dalle altre) sono racchiusi nell’entità stessa.

3.2.1.4.5.3 RepositoryPropertyAllowedVal

È bene ricordare che l'integrità referenziale (vincolo di foreign key) crea un legame tra i valori di un attributo della tabella (denominata interna) in cui è definito ed i valori di un attributo di un'altra tabella (denominata esterna). Il vincolo impone che per ogni riga della tabella interna il valore dell'attributo specificato (se diverso dal valore null) sia presente nelle righe della tabella esterna tra i valori del corrispondente attributo. L'unico requisito che la sintassi impone è che l'attributo cui si fa riferimento nella tabella esterna sia soggetto ad un vincolo *unique*, cioè sia un identificatore della tabella stessa. Tipicamente l'attributo della tabella esterna, cui si fa riferimento, rappresenta in effetti la chiave primaria di tale tabella. Il vincolo può essere definito in due modi, come i vincoli unique e primary key. Le chiavi primaria ed esterna in un'integrità referenziale devono essere definite sullo stesso dominio, ma non devono necessariamente avere gli stessi nomi. La tabella esterna sul piano applicativo rappresenta la tabella principale (master) alle cui variazioni la tabella interna (o slave) deve adeguarsi.

Nel caso in esame la tabella interna è una *<QualifiableObject>PropertyValue* (al termine tra parentesi angolari si sostituisce il nome di un QualifiableObject) e quella esterna è la *RepositoryPropertyAllowedVal*. La tabella *RepositoryPropertyAllowedVal* è creata proprio per gestire le proprietà che assumono valori standard, fissati (da normative, enti, designer), cioè proprietà a cui l'utente non può assegnare un valore arbitrario. Si suppone che GPID=1 sia una certa proprietà i cui valori ammessi (fissati da un ente di standardizzazione) sono solo 3: V1, V2 e V3. In *RepositoryPropertyAllowedVal* si hanno i seguenti "record sintesi", come ad esempio:

GPID	**GPAVCODE**	**ShortDescr**
1	1	V1
1	2	V2
1	3	V3
1	4	null

Per avere la sicurezza, per esempio, che un dato device per la GPID=1 assuma uno o più valori fissati dall'ente di standardizzazione (pubblicati in RepositoryPropertyAllowedVal) in DevicePropertyValue si specifica il link alla tabella esterna:

foreign key (GPID, ValueA) references GenerelPropertyValue(GPID, GPAVCODE)

con GPID=1 e ValueA=2 si garantisce al device di assumere un valore ammesso dalla proprietà GPID=1. Ricordo che la coppia (GPID, GPAVCODE) nella relazione RepositoryPropertyValue è chiave primaria. Ci sono quattro possibili vincoli dichiarativi d'integrità referenziale associati alle operazioni di cancellazione e aggiornamento sulle tabelle. Occorre stabilire come procedere con le righe di <QualifiableObject>PropertyValue, se una riga di RepositoryPropertyAllowedVal è cancellata o aggiornata (cioè quando la chiave primaria GPID, GPAVCODE sono cancellate o aggiornate). Vi sono quattro possibilità:

1. NO ACTION: si effettua un'operazione di abort se ci sono ancora righe di <QualifiableObject>PropertyValue legate ad un RepositoryPropertyAllowedVal che si vuole cancellare o modificare. Non viene eseguita, pertanto, alcuna azione;

2. CASCADE: la cancellazione o la modifica della chiave primaria GPID, GPAVCODE di RepositoryPropertyAllowedVal è propagata a tutte le righe di <QualifiableObject>PropertyValue legate/referenti;
3. SET NULL: nelle righe di <QualifiableObject>PropertyValue legate al RepositoryPropertyAllowedVal cancellato/modificato, si assegna NULL alle colonne GPID, ValueA che fanno da chiave esterna nella riga referente;
4. SET DEFAULT: nelle righe di <QualifiableObject>PropertyValue legate al RepositoryPropertyAllowedVal cancellato/modificato, alle colonne GPID, ValueA vengono assegnati valori predefiniti, specificati nella definizione della chiave esterna.

3.2.1.4.5.4 RPropertyEnabling

Esistono delle RepositoryProperty che dipendono concettualmente da altre. Una RepositoryProperty (GPID) valutata su un QualifiableObject (EvaluateOnQO) ed eventualmente valorizzata (se valorizzata il valore deve essere specificato in GPAVCode, altrimenti in GPAVCode si mette null) abilita la RepositoryProperty specificata da ControlledGPID su un certo QualifiableObject. Un controllo più fine si ottiene specificando un valore in GPAVCode, poichè solo quel valore della RepositoryProperty specificata da GPID abilita la ControlledGPID. Se viene attribuito null a GPAVCode, si ritiene che quella RepositoryProperty indicata da GPID, indipendentemente dal valore che assume, abilita la ControlledGPID. La RepositoryProperty indicata da ControlledGPID è quella dipendente concettualmente (abilitata) da GPID. Per l'abilitazione si possono usare in mutua esclusione:

- (GPID, EvaluateOnQO, GPAVCode);
- oppure DCID.

La mutua esclusione è garantita dal seguente trigger:

```
ALTER TRIGGER [dbo].[GPropertyEnabling_InsertUpdate]
  ON [dbo].[GPropertyEnabling]
  AFTER INSERT, UPDATE
AS
BEGIN
        SET NOCOUNT ON;
        if (select count(*) from INSERTED where (DCID is not null) and (GPID is not null or
EvaluateOnQO is not null or GPAVCode is not null)) > 0
        begin
                raiserror('Non è possibile valorizzare la colonna DCID contemporaneamente alle
colonne GPID, EvaluateOnQO e GPAVCode', 16, 1)
                rollback transaction
        end
END
```

Si consideri l'esempio concreto di seguito.

La RepositoryProperty di GPID=1 è “colore” e ha un GPAVCode=22 che corrisponde a “rosso”, quindi:

RepositoryProperty: **Colore**=rosso

Tale RepositoryProperty è valutata (perchè la possiede) su un certo QualifiableObject, quindi EvaluateOnQO=234 è “macchina”, da cui

RepositoryProperty: **Colore**=rosso valutata sul **EvaluateOnQO**=macchina

deve abilitare la RepositoryProperty di GPID=45 che è “tappezzeria” rappresentata da ControlledGPID, quindi:

ControlledGPID=Tappezzeria

relativa al QualifiableObject=478 che è “interni” rappresentato da ControlledQO, quindi:

ControlledQO=interni

Si suppone, infine, che il record di RPropertyEnabling che definisce tale regola abbia GPEID=1.

Nel seguito viene mostrato la figura 3.6 per maggiore chiarezza:

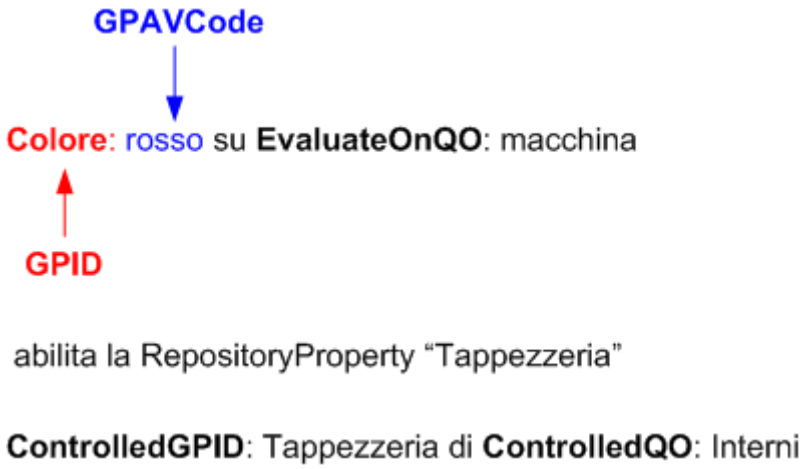


Figura 3.6: RPropertyEnabling

Nella traduzione dal modello ER al modello logico relazionale, l’entità in esame dovrebbe avere un chiave primaria composta di 6 elementi:

- GPID, GPAVCode di RepositoryPropertyAllowedVal;
- QOID, GPID di Property2QO;
- QOID, GPID di Property2QO;

Si rinominano i 6 elementi in modo significativo:

- GPID_RPAV, GPAVCode di RepositoryPropertyAllowedVal;
- QOID_1, GPID_RP di Property2QO;
- QOID_2, GPID_RP2 di Property2QO.

Si confronta la chiave primaria ottenuta (prima riga della tabella) con quella del modello logico desiderato (seconda riga della tabella):

GPID_RPAV	GPAVCode	QOID_1	GPID_RP	QOID_2	GPID_RP2
GPID	GPAVCode	EvaluateOnQO	ControlledGPID	ControlledQO	

Si può notare come i campi della prima e della seconda riga delle colonne evidenziate abbiano una corrispondenza 1 a 1. GPID, invece, viene utilizzato una volta sola, pur riferendosi a GPID diversi:

1. GPID di RepositoryPropertyAllowedVal;
2. GPID di Property2QO.

La fusione di GPID_RPAV e GPID_RP2 in un'unica colonna GPID:

- è possibile, perchè il vincolo di foreign key verso RepositoryPropertyAllowedVal impone che per ogni riga della tabella RPropertyEnabling i valori degli attributi (GPID, GPAVCode) solo se diversi dal valore null debbano essere presenti nelle righe della tabella RepositoryPropertyAllowedVal tra i valori del corrispondente attributo. Quando si vuole esprimere il GPID_RP2, con il solo GPID "comune" a disposizione, occorre specificare GPACCode=null (poichè nella tabella Property2QO non c'è il GPAVCode), ma ciò non dà violazione di integrità referenziale;
- agevola la progettazione, perché non occorre implementare un trigger che controlli eventuali inconsistenze, cioè che GPID_RPAV e GPID_RP2 abbiano valori diversi. La GPID di RepositoryPropertyAllowedVal deve essere comunque associata ad un QualifiableObject, cioè deve essere presente come valore della colonna GPID in un'istanza di Property2QO. GPID, pertanto, partecipa contemporaneamente, se GPAVCode è valorizzato, a:
  - (GPID, GPAVCode) to RepositoryPropertyAllowedVal;
  - (GPID, QOID) to Property2QO.

Gli elementi GPID, GPAVCode, EvaluateOnQO, ControlledGPID, ControlledQO non possono però costituire la chiave primaria, perchè GPID, GPAVCode, EvaluateOnQO possono essere null.

3.2.1.4.5.5 RPropertyValueEnabling

Riprendendo l'esempio di RPropertyEnabling e supponendo che ControlledGPID, cioè Tappezzeria, abbia 10 possibili colori ma, con "Colore: rosso su EvaluateOnQO: macchina, si vuole che ControlledGPID: tappezzeria su ControlledQO: interni assuma solo 3 colori dei 10 ammessi: nero, verde, bianco". Questi 3 colori si specificano come 3 record in RPropertyEnablingValue. Il nome GPpropertyValueEnabling deriva appunto dal fatto di arrivare perfino ad abilitare i singoli valori delle proprietà, quindi ancora più nello specifico rispetto a RPropertyEnabling dove ci si limitavo ad abilitare le proprietà. Se si abilita un sottoinsieme della totalità di valori si può sempre incrementarlo, ma non ulteriormente restringerlo, per questioni di complessità di gestione e comunque perché non richiesto. Esempio: se ho una RepositoryProperty con 10 valori è possibile limitare il dominio a 4 valori, quindi incrementarlo a 7, ma non decrementarlo a 2.

ControlledGPEID	ControlledGPID	ControlledGPAVCode
1	Tappezzeria	nero
1	Tappezzeria	verde
1	Tappezzeria	bianco

3.2.1.4.5.6 <QualifiableObject>PropertyValue

Tale tabella contiene i valori delle RepositoryProperty associate ad un certo QualifiableObject.

Siccome il repository di proprietà raccoglie proprietà non note a priori di qualunque tipo, se si creasse una tabella con unica colonna Value si commetterrebbe un errore, perchè per la colonna Value si deve specificare un unico dominio, ma Value è multidominio. Si consideri il seguente esempio:

RepositoryProperty

GPID	DESCR	VALUE
1	INDIRIZZO	www.dfk1.it
2	ETA	20
3	SPOSATO	true

Si suppone di avere specificato per Value il dominio VARCHAR(255) e si esegue:

1. INSERT INTO RepositoryProperty (GPID, DESCR, VALUE) VALUES (1, 'Indirizzo', 'www.dfk1.it')
2. INSERT INTO RepositoryProperty (GPID, DESCR, VALUE) VALUES (2, 'eta',25)
3. INSERT INTO RepositoryProperty (GPID, DESCR, VALUE) VALUES (3, 'SPOSATO', 'true')

Le ultime due istruzioni sono sbagliate, perchè non sono compatibili con il dominio specificato per la memorizzazione dei valori di Value.

Si può pensare allora di "spalmare" Value su più colonne:

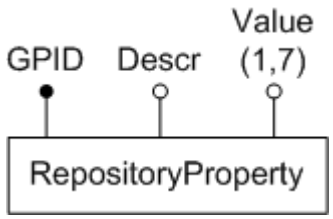


Figura 3.7: RepositoryProperty

La tabella corrispondente all'entita nell'ER sopra, ha i dati non organizzati in modo efficiente, perchè una proprietà è valorizzata per un solo dominio, per cui i rimanenti tipi sono tutti valorizzati con null.

RepositoryProperty

GPID	DESCR	VALUEB	VALUEI	VALUEF	VALUED	VALUET	VALUEL	VALUEA
1	INDIRIZZO	null	null	null	null	www.dfk1.it	null	null
2	ETA	null	20	null	null	null	null	null
3	SPOSATO	true	null	null	null	null	null	null

Un approccio elegante dal punto di vista teorico richiede la specializzazione della tabella in funzione del tipo di dato creando una generalizzazione totale esclusiva di 7 sottotabelle per ogni QualifiableObject. Poiché il numero di record di <QualifiableObject>PropertyValue non è elevatissimo (stimato pari al più a 24000 record) si sceglie di tenere campi null per maggiore praticità di impiego. Per sapere il tipo di dato valorizzato in <QualifiableObject>PropertyValue per una data RepositoryProperty è sufficiente andare a vedere il DataType di quella RepositoryProperty. Se a GPID=1 corrisponde ShortDescr=INDIRIZZO e DataType=Text, significa che tutti i <QualifiableObject>PropertyValue che la referenziano avranno solo VALUET valorizzato, quindi prendendo PPropertyValue che lega ad esempio una data Person (PID=1 che equivale all'admin) a tale RepositoryProperty e supponendo che abbia 3 indirizzi si ha:

PPVID	PID	GPID	IstanceN	VALUEB	VALUEI	VALUEF	VALUED	VALUET	VALUEL	VALUEA
1	1	1	1	null	null	null	null	Admin1@dominio.it	null	null

2	1	2	2	null	null	null	null	Admin2@dominio.it	null	null
3	1	3	3	null	null	null	null	Admin3@dominio.it	null	null

Da un controllo rapido risulta che l'inefficienza dal punto di vista dello storage dovuto ad una tabella parzialmente piena è irrisoria rispetto alle capacità di memorizzazione.

Prove di inserimenti nella "Tabella classica" versus inserimenti in una "Tabellina ad hoc"

boolean

```
DECLARE @CONT int;
SET @CONT = 0;
WHILE (@CONT < 12000)
BEGIN
	INSERT INTO Property(ID, ValueB, ValueI, ValueF, ValueD, ValueT, ValueL, ValueA)
VALUES (@CONT,'true',Null,Null,Null,Null,Null,Null)
	SET @CONT = @CONT+1;
END
```

La tabella occupa 0,445 MB, cioè 455,68 KB

```
DECLARE @CONT int;
SET @CONT = 0;
WHILE (@CONT < 12000)
BEGIN
	INSERT INTO ValueB(ID, ValueB) VALUES (@CONT,'true')
	SET @CONT = @CONT+1;
END
```

La tabella occupa 0,164 MB, cioè 167,94 KB

Conclusione:

Ad ogni inserimento in Property si è penalizzati rispetto ad uno in ValueB di un fattore: **2,713 > 0**

Con lo stesso tipo di dato boolean (stesso valore: true) e 24000 righe il fattore diviene: **2,692 > 0** (0,883 [MB] / 0,328[MB]), ma minore del primo. All'aumentare del numero di record il divario diminuisce, con uno scarto di -0,021.

nvarchar(MAX)

Cambiando il tipo di dato, con nvarchar(MAX) l'inserimento di PAOLO per 12000 righe ha un fattore di penalizzazione di: **1,903 > 0** (0,609 [MB] / 0,320 [MB]), ma minore del booleano di -0,81 a parità di righe inserite.

Aumentando le righe a 24000, il fattore diviene: **1,913 > 0** (1,211 [MB] / 0,633 [MB]). All'aumentare del numero di record il divario aumenta, con uno scarto di +0,010.

Raddoppiando la lunghezza della stringa 'PAOLOPAOLO' con 12000 righe il fattore è: **1,66 > 0** (0,727 [MB] / 0,438 [MB]), ma minore del fattore della stringa 'PAOLO' di uno scarto pari a -0,243.

Raddoppio la lunghezza della stringa 'PAOLOPAOLO' con 24000 righe il fattore è: **1,67 > 0** (1,445 [MB] /0,867 [MB]) maggiore di +0,01 rispetto al precedente, ma comunque minore della stringa singola 'PAOLO'.

Con 30 caratteri (supponiamo sia la lunghezza max) e 24000 righe il fattore è di: **1,332** > 0 (2,383 [MB] / 1,789 [MB]).

int

Cambiando il tipo di dato, con int l'inserimento di 1000 per 12000 righe ha un fattore di penalizzazione di: **2,192 > 0** (0,445 [MB] / 0,203 [MB]).
Aumentando le righe a 24000, il fattore diviene: **2,219 > 0** (0,883 [MB] / 0,398 [MB]). All'aumentare del numero di record il divario aumenta, con uno scarto di +0,027.

datetime

Cambiando il tipo di dato, con datetime l'inserimento di '20090428 18:34:31' per 24000 righe ha un fattore di penalizzazione di: **1,795 > 0** (0,883 [MB] / 0,492 [MB]).

float

Cambiando il tipo di dato, con float l'inserimento di 1,567 per 24000 righe ha un fattore di penalizzazione di: **1,795 > 0** (0,883 [MB] / 0,492 [MB]).

Si suppone, ora, di considerare il legame logico tra un dato QualifiableObject come Device e l'entità RepositoryProperty. Poichè si parla di legame logico, la rappresentazione ER prevede l'utilizzo dell'associazione tra le due entità appena citate, come si può vedere dalla figura 3.8:

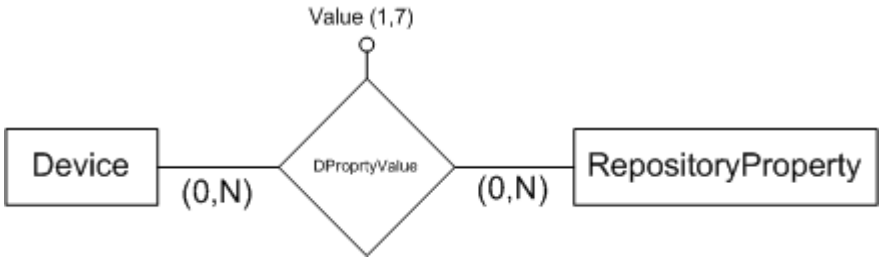


Figura 3.8: Device-RepositoryProperty

Dove min-card(Device, DPropertyValue)=0, perché un device può (non deve) avere valorizzata una proprietà di RepositoryProperty (perchè si tratta di proprietà generali), cioè il device esiste anche se non ha una proprietà di RepositoryProperty valorizzata, così come una proprietà generale esiste senza che debba avere un valore per un certo device. Si è discusso in precedenza che i tipi di valore ammessi sono al più 7, da cui max-card(Value, DPropertyValue)=7 ed almeno un valore deve essere specificato, per cui min-card(Value, DPropertyValue)=1. Con tale rappresentazione una data proprietà è valorizzata una sola volta per un dato device, come si propone nel seguente esempio:
Istanza1 di DPropertyValue = {d1 rP1 valueB1}
~~Istanza2 di DPropertyValue = {d1 rP1 valueT1}~~
Dalle regole di vincolo si ha che una data proprietà è valorizzata più volte per un dato device, ad esempio: il device d1 ha INDIRIZZO che vale: www.1.it, www.2.it, vale www.3.it. Occorre procedere alla reificazione, introducendo il nuovo identificatore IstanceN, come mostrato in figura 3.9:

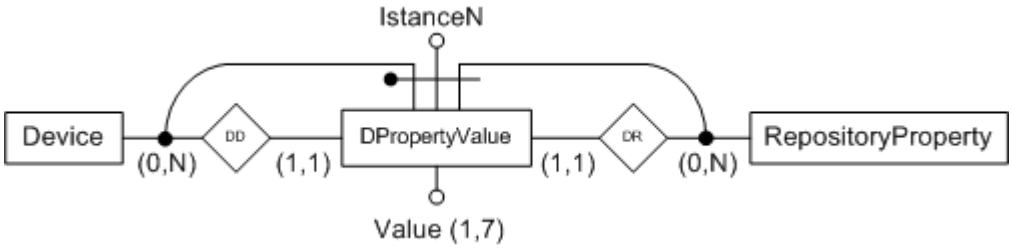


Figura 3.9: DPropertyValue

Un'ulteriore regola di vincolo stabilisce che il legame tra un dato device ed una stessa proprietà può essere valido o meno in istanti di tempo differenti. Per esprimere ciò occorre inserire il nuovo identificatore EffectiveSince e l'attributo IsDeleted (=0 legame valido, =1 legame cancellato) sull'entità DPropertyValue come mostrato in figura 3.10.

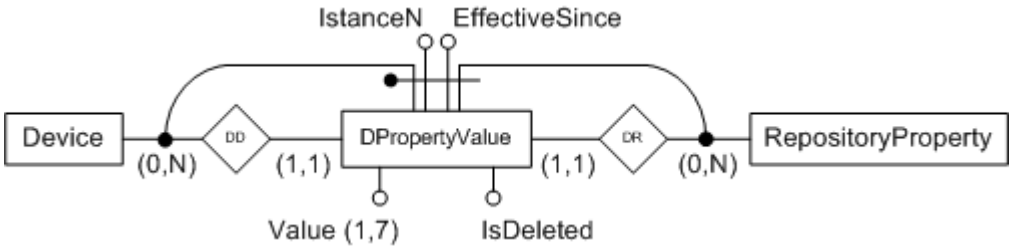


Figura 3.10: DPropertyValue

d1 rp1 istance1 EffectiveSince1 isDeleted=0 (rappresenta: **d1** ha INDIRIZZO che vale **rp1**=www.1.it valido, perchè isDeleted=0, dall'**EffectiveSince1**)

d1 rp1 istance1 EffectiveSince2 isDeleted=1 (rappresenta: **d1** ha INDIRIZZO che vale **rp1**=www.1.it non valido, perchè isDeleted=1, dall'**EffectiveSince2**)

Le entità <QualifiableObject>PropertyValue (a <QualifiableObject> si sostituisce Device, Person, etc.) sono soggette a storicizzazione. La storicizzazione nell'ER, in accordo con la prof.ssa, si rappresentata attraverso un'entità per lo stato attuale legata da una gerarchia ISA o subset a quella storica, come mostrato in figura 3.11.

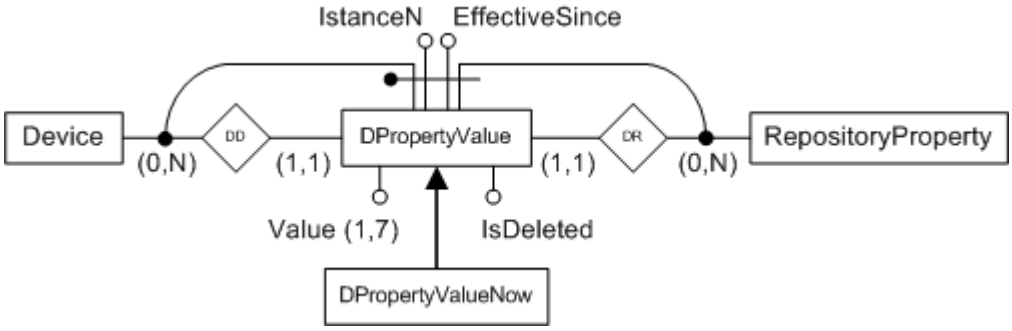


Figura 3.11: DPropertyValue

3.2.1.4.5.7 MostFrequentlyView

È una tabella, in cui il QOID specifica il QualifiableObject (es. Device, Group, etc.); l'ObjectID è

<QualifiableObject>ID, cioè definito in fuzione del QOID, ad esempio:
QOID: Device → ObjectID: DID
QOID: Group → ObjectID: GID
Il ViewerID è, infine, la Person loggata.

3.2.1.4.6 ER2

3.2.1.4.6.1 Group

Tale entità giustifica il fatto che nel modello ER (e conseguentemente nella base di dati) non si trovano entità del tipo “cabina”, “sistema di odorizzazione”, etc.
È un’entità i cui oggetti sono caratterizzati dalla proprietà comune di “raggruppare oggetti”. Nell’entità Group vi sono oggetti che possono apparire disomogenei, potendo essere gruppi di device, gruppi di impianti, gruppi di persone, gruppi di device e di persone, etc. Il tipo di un oggetto Group è specificato dalla RepositoryProperty “GroupType”, mentre la composizione di tale oggetto Group è specificata in GroupMShip<QualifiableObject> (a QualifiableObject si sostituisce Device, Person, etc.). Tali oggetti sembrano non avere caratteristiche comuni per poter essere classificati (far parte) nella medesima entità Group, perchè da un punto di vista logico un gruppo di device non ha alcuna affinità (caratteristiche comuni) come, ad esempio, con un gruppo di persone. In realtà con questo modo di ragionare si focalizza erroneamente l’attenzione sui componenti di un gruppo, che sono di natura differente, mentre invece attraverso un’astrazione concettuale è possibile comprendere la correttezza di una tale classificazione comune a gruppi diversi. Gli oggetti Group devono avere la proprietà comune di rappruppare, indipendentemente dalla natura dei loro componenti. Ad esempio, un oggetto Group può essere:

- un impianto: gruppo di device (un EasyDor e un Edor);
- un sistema di odorizzazione: gruppo di impianti;
- etc.

Si possono considerare oggetti Group che hanno lo stesso schema (es. i componenti del gruppo sono degli oggetti Person) ma sono diversi (per mansione, etc.) come ad esempio un gruppo di tecnici (composta di Person) ed un un gruppo di reperibili (composta sempre di Person). Nel modello è la descrizione, assieme al GroupType, che specificano il criterio di classificazione, ottenendo ciò che è mostrato in figura 3.12:

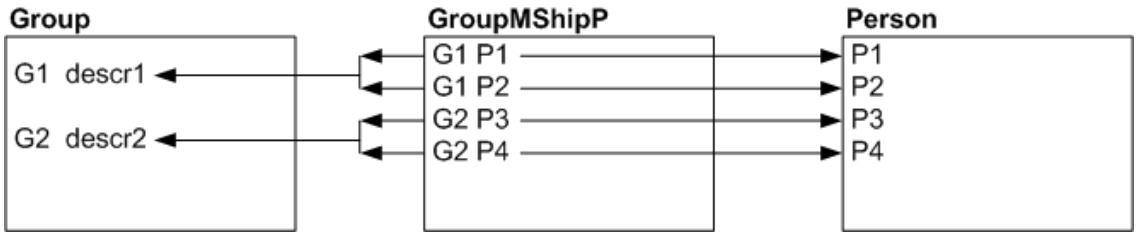


Figura 3.12: Group

Si considera ora il caso di un gruppo costituito da gruppi, come il sistema di odorizzazione, che è un gruppo di impianti che sono a loro volta gruppi di device, come visualizzato in figura 3.13:

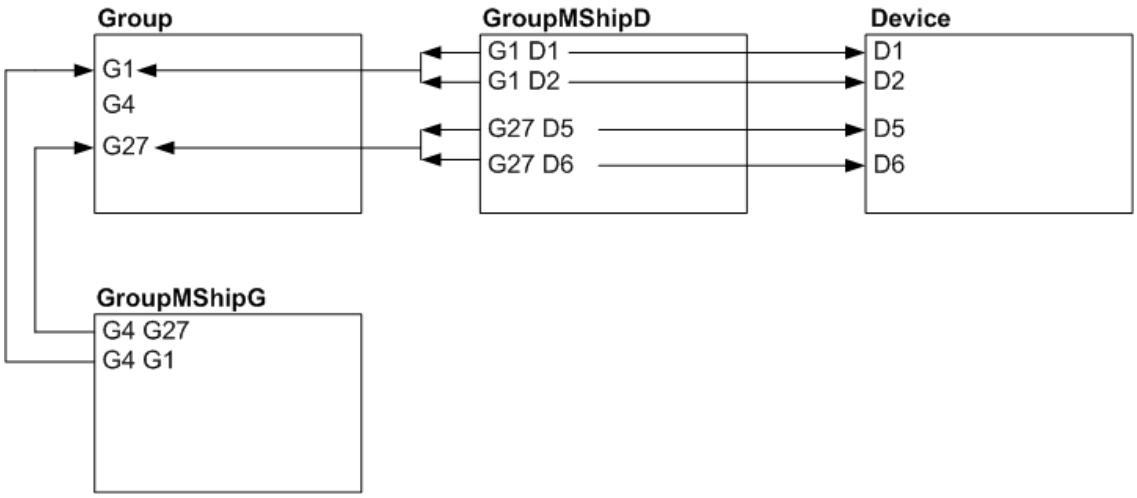


Figura 3.13: Group

Un altro caso è rappresentato da un gruppo costituito da persone e da device (ad esempio per definire dei controlli di accesso) come visualizzato in figura 3.14:

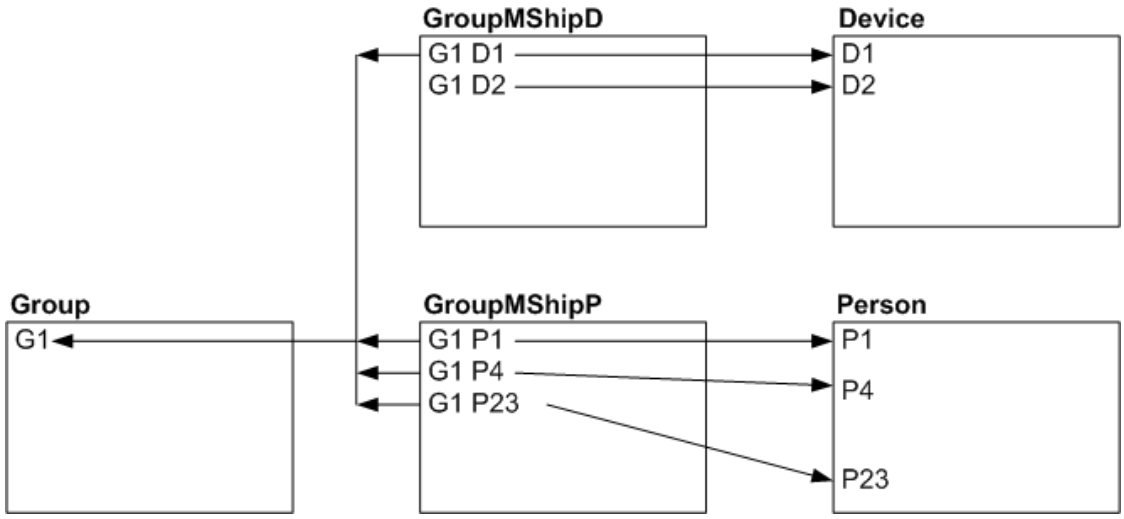


Figura 3.14: Group

Le Person P1, P4, P23 possono accedere ai Device D1, D2 e, quindi, logicamente li inserisco nel Group G1.

Se si creasse un'entità ad hoc per ogni tipologia (criterio di raggruppamento) come "cabina", "sistema odorizzazione", etc. oppure in funzione del numero di partecipanti (Device+Person, Device+Person+Other QualifiableObject, Device+Person+Other QualifiableObject+Other QualifiableObject, etc.) in modo tale che le sue istanze raggruppino oggetti caratterizzate dalle stesse proprietà (ad esempio ogni istanza di "cabina" è composta solo da device e, pertanto, tutti gli elementi appartenenti all'entità "cabina" sono accumunati dalla caratteristiche di un device) la dimensione del DB crescerebbe a macchia d'olio e rapidamente. Tale modellazione, quindi, risulta essere quella minima e garantisce la massima flessibilità, per soddisfare in tempi rapidi le richieste di modifiche repentine da parte del committente del software.

3.2.1.4.6.2 Group2HLSignal

È una tabella statica, che contiene i raggruppamenti di KPI (HLSignal). Determinati user control devono caricare gruppi fissi di HLSignal, che sono recuperati da tale tabella. Nel Current Status Summery (del sistema di odorizzazione) si definisce, ad esempio, un gruppo contenente determinate KPI da mostrare; in tale tabella, pertanto, sono registrati i seguenti record:

GID1-HLSignalID1

GID1-HLSignalID2

GID1-HLSignalID3

GID1-HLSignalID4

Il gruppo GID1 contiene quattro KPI.

3.2.1.4.6.3 ElaboratedData

Un ElaboratedData specifica il valore di un certo HighLevelSignal, calcolato o su un gruppo o su un device (in mutua eslusione). Poichè un HighLevelSignal non sempre è mappato su unico LowLevelSignal, non è possibile risalire dal HighLevelSignal al device, al quale si riferisce. Inoltre, un HighLevelSignal può essere riferito a un device, come risultato di una certa elaborazione, ovvero tale HighLevelSignal non è pervenuto dal device tramite una comunicazione. La navigazione attraverso Communication, pertanto, non permette di risalire al device. I motivi appena esposti giustificano la presenza del ciclo tra ElaboratedData e Device (o Group). Non è possibile identificare univocamente un ElaboratedData con la coppia (GID, EffectiveOn) come primary key e con la coppia (DID, EffectiveOn) come alternative key, in quanto GID e DID possono assumere valori null. Se fosse corretto identificare ElaboratedData con una delle coppie precedenti, l'EffectiveOn dovrebbe essere presente in esse, perchè il calcolo di un ElaboratedData per uno stesso gruppo o per uno stesso device si ripete ad intervalli di tempo prefissati, ad es. ogni 24 ore. La partecipazione di Group o Device verso ElaboratedData è (0, N) perchè su uno stesso Group (o Device) si possono calcolare zero o 'n' ElaboratedData. Per impedire l'inserimento di un record, con DID e GID valorizzati contemporaneamente, si è implementato il seguente trigger:

```
ALTER TRIGGER [dbo].[ElaboratedData_InsertUpdate]
   ON [dbo].[ElaboratedData]
   AFTER INSERT, UPDATE
AS
BEGIN
        SET NOCOUNT ON;

        if (select count(*) from INSERTED where GID is not null AND DID is not null) > 0
        begin
                raiserror('Non è possibile valorizzare contemporaneament le colonne GID e DID', 16,
1)
```

```
            rollback transaction
        end

        declare @date as datetime
        set @date = getdate()
        update [dbo].[ElaboratedData] set ModifiedOn = @date
        where EDID in (select EDID from INSERTED)
END
```

La max-card(HighLevelSignal, HLSignal2ED) è N poiché, ad esempio, uno stesso HLSignal viene aggiornato una volta al giorno. Dopo un mese si avranno, pertanto, 30 istanze di ElaboratedData associate ad uno stesso HighLevelSignal.

3.2.1.4.6.4 Activity

L'informazione della persona che ha compiuto l'attività può essere rappresentata da una RepositoryProperty.

3.2.1.4.6.5 Activity2Object

Tale tabella consente di legare un'attività ad un Device o ad un Group. Per impedire l'inserimento di un record, con DID e GID valorizzati contemporaneamente, si è implementato il seguente trigger:

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER TRIGGER [dbo].[Activity2Object_InsertUpdate]
   ON  [dbo].[Activity2Object]
   AFTER INSERT, UPDATE
AS
BEGIN
        SET NOCOUNT ON;

        declare @pACT2OID as int
        declare @pDID as int
        declare @pGID as int
        declare @date as datetime
        set @date = getdate()

        declare RowCursor cursor local static for
        select ACT2OID, DID, GID from INSERTED
        open RowCursor
        fetch next from RowCursor into @pACT2OID, @pDID, @pGID
        while @@fetch_status = 0
        begin
                if ((@pDID is not null) and (@pGID is not null))
                begin
                        raiserror ('E'' possibile valorizzare solo il campo "DID" oppure solo il campo
"GID"', 16, 1)
```

```
            end

            update [dbo].[Activity2Object]
            set EffectiveSince = @date
            where ACT2OID = @pACT2OID

            fetch next from RowCursor into @pACT2OID, @pDID, @pGID
        end

END
```

3.2.1.4.6.6 GenericAlgorithm

È una tabella che contiene, come suoi record, delle DLL C# che implementano degli algoritmi.

3.2.1.4.6.7 GenericAlgorithmExecutionResult

È una tabella, che contiene i risultati di un dato algoritmo, per enumerarli (non in ordine di importanza, ma semplicemente di numero). Se un algortimo ha prodotto dieci risultati e si desidera utilizzare il quarto, che deve essere ragionevolmente recente, lo si può recuperare da tale tabella, evitando di rieseguire l'algoritmo. In questo modo si ottimizzano le prestazioni dell'elaborazione.

3.2.1.4.6.8 Tipo di gruppo

L'entità Group non ha il tipo come attributo interno, perchè può essere multidominio, per cui GroupType è una RepositoryProperty.

3.2.1.4.6.9 LogDataSent

Dovendo, ad esempio, svolgere un'Attività di Programmazione (AP1) di 10 parametri di un Device (D1) ed essendo la dimensione del testo a disposizione di un SMS limitata a 160 caratteri, per quell'atto di comunicazione si può rendere necessario l'invio di più SMS (SMS1, SMS2, SMS3). Si è stabilito di riferire a quell'atto di comunicazione gli SMS inviati, cioè:

(AP1, D1, {SMS1, SMS2, SMS3}**)**

LogDataSent non deve essere legato a Device con un'associazione, per sapere il Device a cui ci si riferisce (tramite una foreign key DID). Tale informazione è ricavata con la foreign key CommID su Communication, che ha la foreign key DID verso Device. LogDataSent non deve avere l'attributo CommDirection, perchè è ricavabile in Communication (che ha l'attributo CommDirection) con la foreign key CommID. Le entità figlie di LogDataSent sono adibite alla memorizzazione della corrispondente struttura dati inviata da eCentral al device.

3.2.1.4.6.10 EndPoint

Per garantire un certo grado di flessibilità nella gestione sia dei dispositivi sia di eCentral dal punto di vista della comunicazione, si è deciso di tenere separato il concetto di "entità comunicante" da quello di

“estremo di comunicazione”. Questo approccio prevede che una comunicazione sia considerata da due punti di vista ben distinti e separati: a basso livello, se si considera stabilita tra endpoint, mentre ad alto livello, se coinvolge (si “realizza”) entità logiche di alto livello quali ChannelManager e device. Questa suddivisione logica su due livelli permette di rappresentare agevolmente e, quindi, di astrarre concetti propriamente legati alla gestione del canale di comunicazione (come il blocco di SIM, disabilitazione di porte, etc.). È possibile in questo modo tenere separati questi concetti da una visione di più alto livello, che non deve essere influenzata dai dettagli di comunicazione inerenti aspetti difformi da quelli che sono gli obiettivi (interessi) della logica applicativa, per la quale occorre focalizzare l’attenzione sui soggetti coivolti nella comunicazione stessa. L’entità endPoint viene specializzata per tipo di canale di comunicazione e lato, secondo una gerarchia di generalizzazione totale esclusiva. Poiché non avrebbe senso parlare di GSMEndPoint_Device come concetto a sè stante (cioè senza correlarlo al device che l’utilizza), la sua partecipazione all’associazione di legame endpoint-device ha min-card fissata a 1. L’attributo “IsEnabled” dell’associazione Device2GSMEndPoint e Device2TCPEndPoint è indipendente dall’“IsEnabled” di device e GSMEndPoint, ovvero un device o un endpoint potrebbero essere o meno attivi, senza compromettere la validità del legame device-endpoint. Se un EndPoint ep1 non è attivo, pur essendo questo disservizio temporaneo, si desidera comunque mantenere valida l’affermazione che il device d1 usa l’EndPoint ep1. Nella fase di progettazione del modello dati conviene sempre cercare di indirizzare la costruzione dell’ER verso una soluzione capace di gestire in modo intelligente i ‘fili’ (legami) a livello concettuale tra le varie entità. È, pertanto, preferibile procedere ad una costruzione del modello dati mirata all’individuazione di inconsistenze dei dati e concepita per la risoluzione di un problema già in fase di progettazione, sfruttando la capacità di visione d’insieme del modello stesso. Si suppone di soddisfare un certo vincolo a livello di modello ER, ad esempio, per garantire che i record di Communication si riferiscano a comunicazioni avvenute correttamente ad opera di device (device2eCentral). Avendo a disposizione i dati di una comunicazione GSM, dai quali è possibile risalire all’EndPoint utilizzato dal device per comunicare, si vuole controllare che ciò che è avvenuto nella realtà abbia effettivo riscontro con i dati tenuti nel DB, cioè che l’endPoint esista in anagrafica e sia attivo (consistenza dei dati ottenuti dalla comunicazione con quelli di anagrafica). Si desidera, pertanto, impedire l’inserimento di un record di Communication, che rappresenta una comunicazione impossibile da effettuare nella realtà, nel caso in cui dalla lettura della seriale si ricava che il device utilizza una SIM, che da anagrafica risulta disattivata. Il modello proposto in figura 3.15 di pag. 134 sarebbe inappropriato a tale scopo:

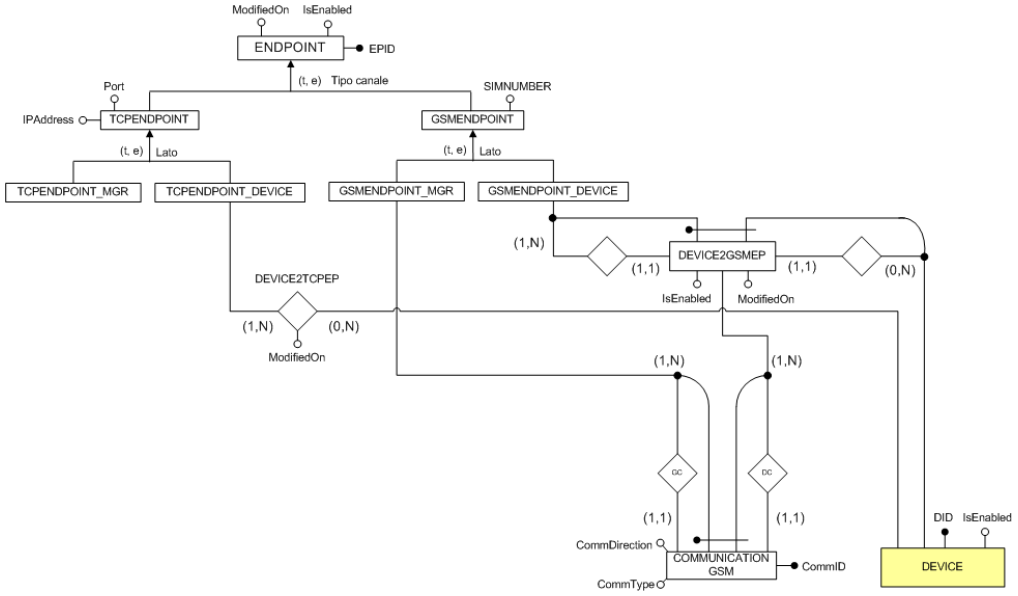


Figura 3.15: LLSignal

Con tale rappresentazione si soddisferebbe la regola di vincolo: “per un dato Device e un dato GSMEndPoint_Device esiste un unico legame che può essere valido o meno a seconda del valore dell’attributo *isEnabled* ad un certo istante di tempo specificato da *ModifiedOn*. Un gsmendPoint_Manager e un gsmendPoint_device partecipano ad una precisa comunicazione”. L’espressività dell’ER non mi permette di specificare un vincolo sul valore di “isEnabled” dell’associazione Device2GSMEndPoint, GSMEndPoint_Device e GSMEndPoint_MGR sul quale si basa il mio vincolo per il check di validità delle SIM (lato Channel Manager e Device) e del legame che sincera che un device utilizza una certa SIM per comunicare (Device2GSMEndPoint). Con tale rappresentazione, inoltre, per ogni record di Communication si dovrebbero specificare dettagli necessari per una certa regola di vincolo (che non è quella richiesta), ma non utili ai fini della caratterizzazione di una comunicazione secondo le specifiche (di più alto livello). Nello schema di Communication si devono mantenere memorizzati in modo permanente solo gli attributi significativi ad alto livello, cioè non interessa conoscere per ogni record di comunicazione entrambi gli endpoint o solo l’endpoint sorgente, cioè dettagli di basso livello. Di una comunicazione interessa conoscere il device (o soggetto della comunicazione) che ha effettuato o a cui è rivolta una comunicazione (come se un programma di instant messaging mostrasse l’interlocutore con l’indirizzo IP del computer anzichè lo username/nickname). Con il modello ER sarebbe inutile soddisfare una certa regola di vincolo che si “avvicina a quella desiderata”, per cui conviene semplificare l’ER, demandando ad un trigger l’eventuale controllo citato in precedenza:

CREATE TRIGGER CheckCommunication ON [dbo].[Communication]

FOR INSERT

BEGIN

Declare @cont as int

SELECT @cont=COUNT(G.GSMEPID)

FROM GSMENDPOINT_DEVICE AS G

JOIN DEVICE2GSMEP AS D2G ON G.DID = D2G.DID

JOIN INSERTED I ON D2G.DID = I.DID

where G.IsEnabled = 1

and D2G.IsEnabled = 1

IF (@CONT) == 0

Raiserror('Inconsistenza di dati. Il device non può comunicare con una SIM che non può usare (IsEnabled=0 sull'associazione DEVICE2GSMEP) oppure usa una SIM che non è attiva (IsEnabled=0 sull'entità GSMENDPOINT.')

Rollback transaction

END

END

3.2.1.4.6.11 Communication e CommunicationData

Le tabelle **Communication** e **CommunicationData** sono le uniche ad essere aggiornate in real-time con i dati delle comunicazioni con i device, mentre LLSignal è statica. Quando un device di 2° generazione invia un HLSignal, che in realtà viene acquisito dal sistema come LLSignal in una Communication, viene riconosciuto come HLSignal (e mappato su di esso) successivamente. I suoi valori sono mantenuti in CommunicationData e non vengono replicati in ElavoratedData. Gli ElaboratedData sono i valori dei soli HLSignal prodotti da eCentral tramite apposite elaborazioni e riferiti o ad un device o ad un gruppo e non sono, quindi, acquisiti attraverso comunicazioni. Quanto descritto viene mostrato nella figura 3.16 di pag 140.

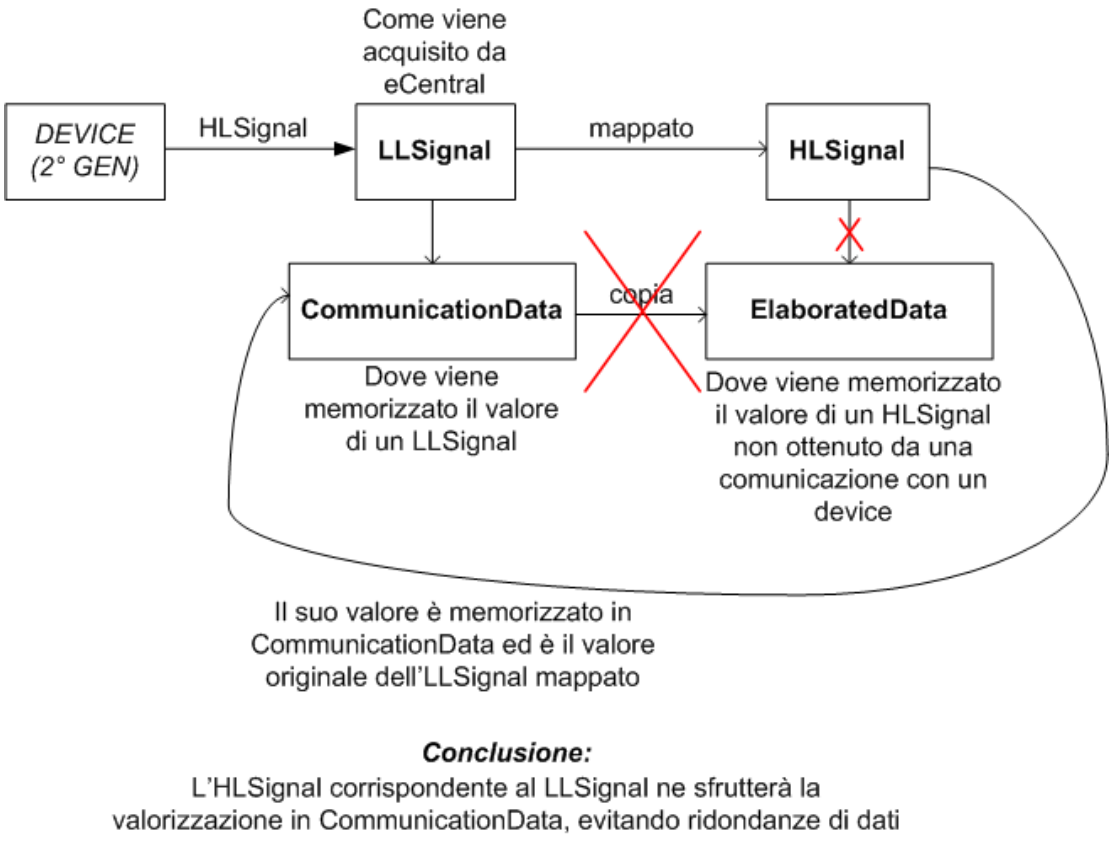


Figura 3.16: LLSignal

3.2.1.4.6.12 LowLevelSignal VS HighLevelSignal

La distinzione tra questi due tipi di Signal è sottile, ma ha un notevole impatto dal punto di vista architetturale, poiché garantisce una trattazione uniforme dei dati ad alto livello. Si conoscono già i possibili raggruppamenti per modello o classe caratterizzanti i device ma, finora, non si è tenuto conto anche del tempo come criterio distintivo. Tra i device contemplati in anagrafica ve ne sono di più "datati" ed altri più recenti. I primi utilizzano driver semplici, che generano Signal di basso livello, misure grezze etichettate, ad esempio, Signal 200 e 201. Questi due Signal possono essere forniti da un sensore, ad esempio un misuratore di flusso. Come detto in precedenza si tratta di dati grezzi che presi singolarmente non danno alcun contributo informativo significativo, ma che acquistano rilievo se opportunamente trattati/correlati insieme. Tali Signal, pertanto, costituiranno l'input per uno specifico algoritmo di eCentral che, attraverso un'adeguata elaborazione, produrrà in output un altro Signal etichettato Signal 400. Con l'evoluzione tecnologica i driver dei device di "seconda generazione" sono più sofisticati, capaci di elaborazioni complesse e, quindi, capaci di sintetizzare in un unico Signal quello che un device di "prima generazione" scomponeva in più Signal. Tali device più evoluti sono in grado di fornire loro stessi un Signal (con diverso ID, ad esempio Signal 500), ma che semanticamente coincide con Signal 400, sollevando eCentral dall'onere di processare dati più grezzi tramite un algoritmo specifico. Occorre, però, procedere ad uniformare i vari Signal che rappresentano esattamente la stessa cosa, quindi i LowLevelSignal 400 e 500 saranno attribuiti ad unico HighLevelSignal con ID, pari ad es. a 600. I Driver

e una parte della logica di eCore ragionano in termini di LowLevelSignal, mentre la presentazione e l'elaborazione di alto livello richiede l'uso degli HighLevelSignal, per poter dimenticare che quelle informazioni sono pervenute da quei device direttamente oppure che sono state ottenute in modo indiretto come output di un'elaborazione preventiva da parte di eCentral/device più "intelligenti". Operare ad alto livello comporta un'astrazione maggiore e, quindi, non ci si deve più preoccupare di gestire informazioni inerenti dettagli di più basso livello, come la gestione delle differenze nell'anno di produzione dei device, che si traduce in una differente capacità di elaborazione degli stessi. Queste differenze devono essere gestite immediatamente durante l'acquisizione, in modo da garantire l'indipendenza da esse prima possibile. Viene mostrata la figura 3.17 per maggior chiarezza:

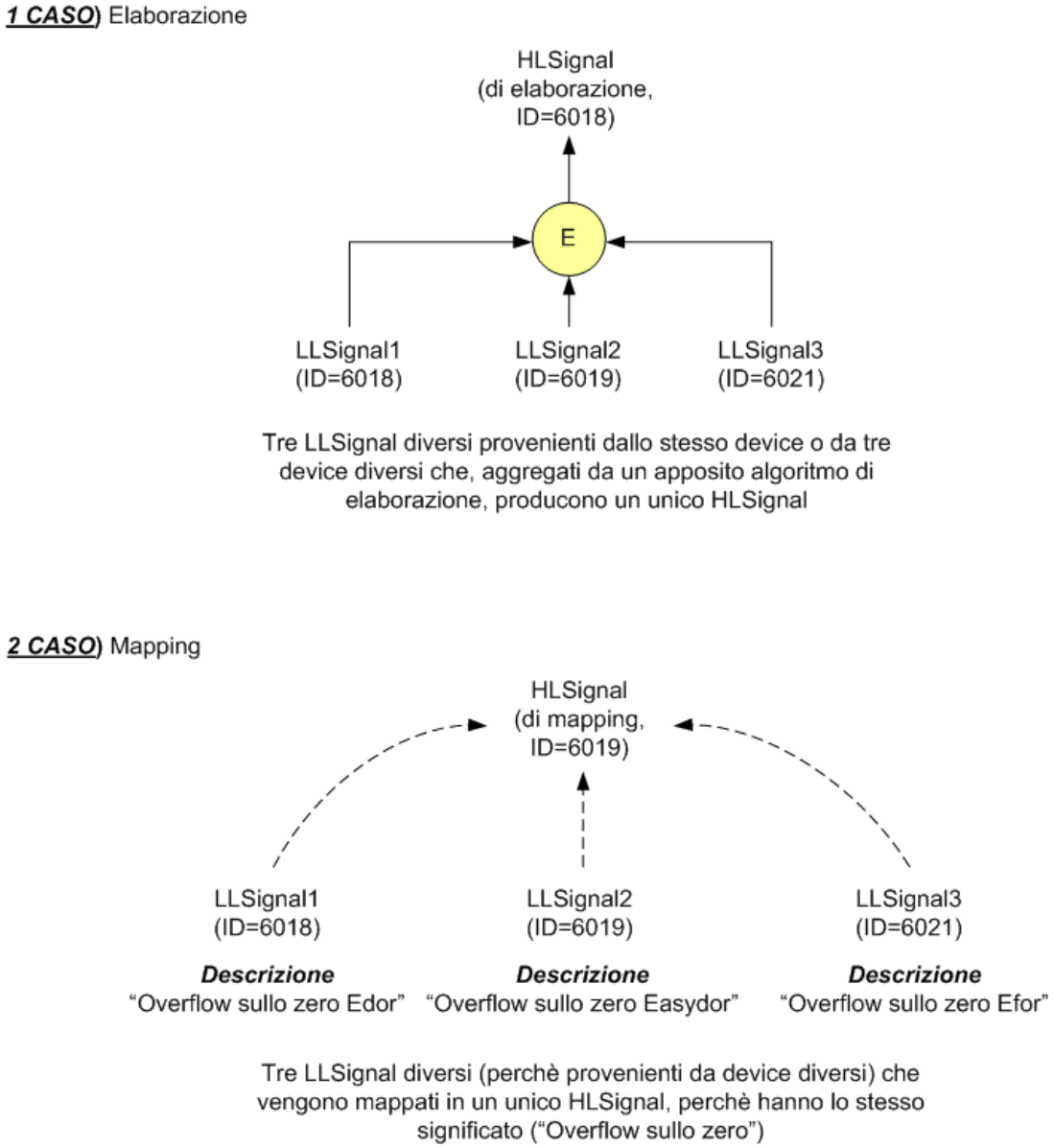


Figura 3.17: Mapping

Tutti i dati elaborati vengono memorizzati come HighLevelSignal.

Min-card(LowLevelSignal, LH) = 1, perchè un LLevelSignal da solo o aggregato ad altri LowLevelSignal viene sempre mappato in un HighLevelSignal.
Min-card(HighLevelSignal, LH) = 1, caso di device di 2 generazione, per cui un HighLevelSignal è mappato su un preciso LowLevelSignal.
Max-card(HighLevelSignal, LH) = N, caso di aggregazione di LLSignal, per cui su uno stesso HighLevelSignal confluiscono fino a N LowLevelSignal.

3.2.1.4.6.13 G2GRShip

Al momento si segnalano le seguenti relazioni fra impianti (gruppi di device):

1. *a-son-of*: rappresenta il legame di discendenza di un livello come mostrato in figura 3.18.

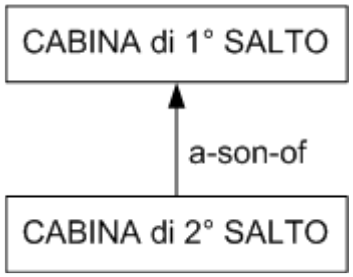


Figura 3.18: a-son-of

3.2.1.4.6.14 H2HRShip

Al momento si segnalano le seguenti relazioni fra HighLevelSignal:

1. attivazione-rientro allarme (ID=288 in ApplicationObjectI18N);
2. dettaglio evento (ID=289 in ApplicationObjectI18N).

3.2.1.4.6.15 Nomi delle relazioni

I nomi delle relazioni tra gruppi (G2GRShip), device (D2DRShip) e HighLevelSignal (H2HRShip) non possono essere attributi delle rispettive relazioni “Elem”, cioè G2GRShipElem, D2DRShipElem, H2HRShipElem, perchè il nome è soggetto ad internazionalizzazione. Occorre, pertanto, costruire una tabella ad hoc per i nomi delle relazioni con la foreign key IALID per l’internazionalizzazione.

3.2.1.4.6.16 Location

Un’esigenza dell’utente è quella di conoscere la disposizione geografica degli impianti. A livello fisico un impianto può essere collocato in un unico luogo fisico, ma vi è l’interesse da parte dell’utente di correlare da un punto di vista di prossimità geografica più impianti, associandoli ad un’unica locazione geografica (*location*) che è concettuale, ma non fisica per i motivi appena esposti (un impianto risiede sempre in unico luogo). Si considera il caso mostrato in figura 3.19 di pag. 139:

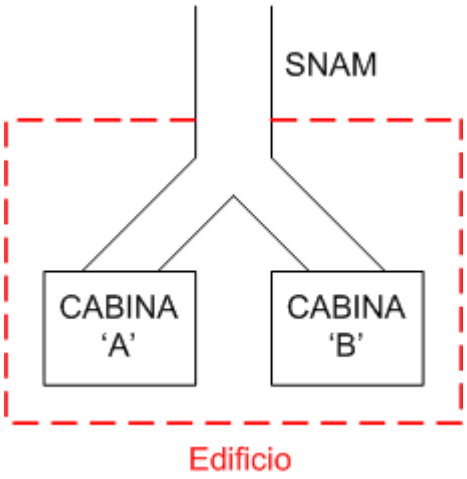


Figura 3.19: Location

Come mostrato in figura il tubo SNAM può giungere ad un edificio, in cui si trovano due diverse cabine (la cabina 'A' potrebbe non avere alcun gruppo di riduzione). I due impianti si trovano fisicamente in due posizioni diverse (ad esempio ad una distanza di 10 m l'una dall'altra), ma essendo situate nello stesso edificio, a livello logico appartengono alla medesima *location*. Si deve astrarre il concetto di location rendendolo indipendente dalla necessità di farlo corrispondere sempre ad un edificio. Riassumendo:

- una location può coincidere con un impianto;
- una location può ospitare più impianti differenti.

3.2.1.4.6.17 Attributo "Period" di HighLevelSignal (HL) e LowLevelSignal (LL)

Alcuni HL/LL Signal sono misure non associate ad un unico istante, ma si riferiscono ad un intero intervallo di tempo come, ad esempio, la portata: numero di m^3/h di gas che sono transitati in un dato tubo. È importante creare un modello di dati il più possibile omogeneo, in modo da svincolare la fase di recupero ed elaborazione dati dalla necessità di dover gestire differenze grossolane o fini fra gli stessi. Per evitare di avere taluni HL/LL Signal associati a due istanti di tempo (t1=istate di tempo iniziale e t2=istante di tempo finale) ed altri associati ad un unico istante di tempo (quello di misura), si decide di rendere persistenti su DB qualunque tipo di HL/LL Signal con un unico timestamp associato, garantendo una gestione uniforme di tutti gli HL/LL Signal. La tabella HL/LL Signal prevede una colonna "Period" che sarà valorizzata solo per gli "HL/LL Signal di durata" (valore medio calcolato per forza di cose su un periodo di tempo), mentre sarà null per le gli "HL/LL Signal di istante" (misura istantanea). Il codice, conoscendo la semantica degli HL/LL Signal, sarà di conseguenza in grado di gestirli opportunamente. Saranno sviluppati metodi specifici di elaborazione degli "HL/LLSignal di durata" in funzione dell'intervallo di competenza. Una modifica dell'intervallo di competenza si ripercuote ovviamente sul codice. Gli intervalli di competenza hanno durata variabile, si possono avere:

HLSignalID: 1507 Δt=1h

HLSignalID: 1703 Δt=30min

Es.

Da telettura perviene l'HLSignal con timestamp riferito a t2, che si rende persistente con quel timestamp su DB. Il sistema saprà ricavare l'istante di tempo da cui è partita la misurazione facendo (t2 (timestamp associato al HLSignal)) – (Δt di competenza).

La colonna Period permette anche di ricavare il numero di HL/LL Signal mancanti dato un certo periodo di "silenzio" del device, durante il quale non si ricevono più dati. Si consideri il caso semplice di un device avanzato che spedisce HLSignal calcolati su un Δt di 20 min. All'istante di tempo t (generico), trascorso l'intervallo di tempo di 20 min, si riceve l'HLSignal. Se per un intervallo di tempo di 2 ore non si ricevono più dati (assenza campo GSM, assenza di alimentazione, etc.), significa che mancano **(**2 [h] x 60 [min]**)**=120[min]/20[min/HLSignal]=**6 [HLSignal]**. L'attributo "Period" è, pertanto, molto importante perché permette di valutare la possibilità di applicare, ad esempio, un algoritmo di interpolazione senza commettere errori che superano una certa tollerenza consentita. Nel caso in cui il numero di misure mancanti superi il limite di campioni consecutivi mancanti, le misure prodotto non possono essere considerate valide.

3.2.1.4.6.18 HLSignal con isKPI=true è un KPI. Problema di aggiornamento del KPI

Si suppone di considerare il Signal consumo giornaliero. Il dato viene calcolato alla fine di ogni giorno. Se si vuole visualizzare il consumo giornaliero di ieri è facile, perchè il giorno ieri è concluso e non mi giungono più dati dai device. Si suppone di voler, invece, visualizzare il consumo giornaliero, quando la giornata non è ancora finita. La definizione di Signal elaborato, cioè di HighLevelSignal, prevede che ad esso sia associata la scadenza temporale di calcolo (alla fine del giorno, settimanale, mensile, etc.). Se il dato eleborato ha scadenza giornaliera:

(00:00 -> 00:00) → dato elaborato "completo"

Se il RTC di riferimento indica, ad es., le 17:00 si dovrebbero attendere 7 ore pe ricevere il Signal dalla periferica. Per ottenere il valore di consumo giornaliero si interroga con una ronda estemporanea il device. Il Signal acquisito deve, comunque, essere memorizzato su DB, in quanto tutti i Signal devono essere resi persistenti, ma non è un dato completo, perchè la giornata non è ancora finita. Se il consumo giornaliero non è completo, non può essere confrontato con i dati giornalieri memorizzati come completi, cioè:

RTC: 28 giugno 2009-05-15 ore 17.00

Esempi di Signal:

misura1: 28 giugno 2009-05-15 (sottinteso 00:00)

misura2: 27 giugno 2009-05-15 (sottinteso 00:00)

misura3: 28 giugno 2009-05-15 ore 17.00

Le misure 1 e 2 sono complete, mentre la misura 2 no.

L'aggiornamento del valore dell'HLSignal "a tempo d'esecuzione" può essere effettuato anche in automatico per mezzo di uno schedulatore ogni Δt < (Δt associato per quel HLSignal, ad es. 1 gg) con

tolleranza concordata con l'utente. Se il Δt di aggiornamento dell'HLSignal è ogni 10 min nell'arco di una giornata, l'utente ritiene comunque valido il dato visualizzato, anche se l'HLSignal è cambiato di valore durante tale intervallo Δt (in quanto l'aggiornamento avviene solo alla fine del Δt).

Esistono diversi soluzioni:

1. PUSH: il device fornisce il Signal e il sistema riaggiorna tutti gli HLSignal dal cui calcolo dipende la nuova misura acquisita;
2. PULL: il sistema aggiorna i dati quando l'utente accede alla finestra di visualizzazione "dati acquisiti":
    a. se l'utente accede di frequente alla finestra, il sistema deve eseguire ripetutamente gli stessi calcoli (tanti quanti gli accessi, quindi al crescere degli accessi cresce il numero di calcoli da eseguire) con un impegno non efficiente delle risorse;
    b. se l'utente apre poco la finestra, ha dati poco aggiornati;
c. se l'utente apre la finestra dopo molto tempo, il sistema impiega molto tempo per eseguire i calcoli necessari all'aggiornamento, con ripercussioni sul tempo di risposta per la visualizzazione della finestra. Se tali dati servono alla sezione di report, non si ha la garanzia di completezza dati (poichè i dati vengono acquisiti saltuariamente).
3. aggiornamento sincrono per mezzo di schedulatore ed eventuale interrogazione estemporanea (now) per aggiornare il valore dell'HLSignal di interesse al momento. In questo modo è possibile stabilire in real-time se il sistema è ben dimensionato.

Come "per il buon studente lo studio giornaliero lo tiene al passo garantendogli un buon profitto", analogamente la soluzione 3 risulta essere la migliore e, quindi, quella da adottare.

Si utilizza il seguente esempio per indicare quali dati utilizzare nel calcolo del KPI.

RTC di riferimento: martedì 12/9/09 00:00

Supponendo sempre un Δt giornaliero, i dati inviati dal device Edor il giorno 11/09 sono stati utilizzati per il calcolo del KPI di quel giorno, contrassegnandolo come completato con una bandieria "virtuale" che indica anche il punto di partenza per il calcolo del KPI del giorno seguente. Al termine della giornata corrente (martedì 12/09) si procede ad un nuovo calcolo del KPI.

3.2.1.4.6.19 Differenza tra GroupMemberShip<QualifiableObject> e <QualifiableObject>2<QualifiableObject>RShipElem

La prima tipologia di relazione rappresenta quella di inclusione di un <QualifiableObject> in un gruppo, mentre la seconda tipologia di relazione rappresenta quelle fra elementi di uno stesso <QualifiableObject>.

3.2.1.4.6.20 SyncroAlgorithmGAID

I Signal (le misure) sono etichettate con un datetime, es: <istante di tempo della misurazione> <Valore>. Nel database memorizzo l'istante di tempo comunicato dal device rispetto al suo RTC (RTC del device)

per l'istante di tempo in cui è avvenuta quella misura. Si pensi all'HLSignal "totale consumo giornaliero" che, per standard, dovrebbe essere comunicato alle fine di ogni giornata, cioè alle ore 00:00. Si suppone che il RTC di riferimento sia 00:00 11/03/09, cioè l'istante di tempo in cui dovrebbe arrivare l'HLSignal. Se tale misura arriva dal dispositivo alle 00:01, oppure alle 00:03, oppure 00:05, etc. perchè il clock del device è soggetto ad una deriva temporale in avanti rispetto al RTC di riferimento, si afferma che tale misura è un consumo riferito al giorno precedente, cioè al 10/03/09. L'HlSignal è registrato nel DB con l'istante di tempo comunicato secondo il RTC del device, cioè 00:01, oppure 00:03, oppure 00:05, ma in una "tabella dati" o grafico tale misura verrà approssimata alle ore 00:00 con una certa tolleranza. Per ogni Signal, dunque, bisogna calcolare la corretta tollerenza in funzione della periodicità con cui viene compiuta la misura. Se il device campiona le misure ad una certa frequenza f, la tollerenza deve essere > 2f.

Possono presentarsi situazioni nelle quali i timestamp di misure logicamente "affini" tra loro (e.s.: valori relativi allo stesso signal ma provenienti da device differenti) sono "leggermente" differenti a causa della non perfetta sincronizzazione degli orologi locali ai device.

Si verifica, quindi, la situazione in cui alcuni di questi timestamp differiscono di un intervallo di tempo tale (entro un certo margine/tollerenza temporale) da renderli logicamente correlati dal punto di vista dell'operatore e dal punto di vista logico, pur essendo differenti dal punto di vista del software (che tratta i dati con la massima sensibilità e, quindi, ritiene i due timestamp diversi). Occorre, pertanto, definire una politica di comportamento sufficientemente elastica per mascherare, dove necessario, queste piccole differenze pur mantenendo la corretta persistenza dei dati nel database. Nella presentazione a video dei dati (ad esempio sotto forma di tabella) potrebbe, quindi, rendersi necessario uniformare queste misure. Quanto fin qui descritto può essere più chiaramente compreso con un esempio concreto. Si suppone di presentare i dati nella vista "analisi dati". Si considerano due device che forniscono i valori dello stesso HLSignal di ID = 100

Timestamp	**Device1**	**Device2**
7.00	valore	
7.03		Valore
8.00	valore	
8.03		Valore

Se la frequenza di invio dati è 1[misura/h], ad ogni ora i device spediranno la misura rilevata. Se il RTC di Device2 ha una deriva temporale di 3 minuti rispetto al RTC di Device1 (che si suppone in punto, cioè di valore uguale al nostro RTC) la visualizzazione dei dati produrrebbe la seguente fastidiosa visualizzazione a pattern periodici in griglia. Con la normalizzazione isocrona di dati non sincroni si ottiene, invece, il seguente risultato migliore sotto l'aspetto visivo:

Timestamp	**Device1**	**Device2**
7.00	Valore	Valore
8.00	Valore	valore

Si potrebbe presentare anche il problema di frequenze diverse d'invio delle misure che aggraverebbe ulteriormente la situazione. Si suppone che il Device1 fornisca un HLSignal ogni 10 min, mentre il Device2 fornisce un altro HLSignal ogni ora, come visualizzato nella tabella 3.1. Questo situazione non rappresenta un problema, perchè è corretto che esistano misure con frequenze d'invio diverse. Tale effetto, poi, si può sommare all'asincronia di invio dati, come visualizzato nella tabella 3.2 (shift temporale di 7 min del device 2 evidenziato in grassetto)

Timestamp	**Device1**	**Device2**
7.00	valore	valore
7.10	valore	
7.20	valore	
7.30	valore	
7.40	valore	
7.50	valore	
8.00	valore	valore
8.10	valore	
8.20	valore	
8.30	valore	
8.40	valore	
8.50	valore	
9.00	valore	valore

Tabella 3.1

Timestamp	**Device1**	**Device2**
7.00	valore	valore
7.10	valore	
7.20	valore	
7.30	valore	
7.40	valore	
7.50	valore	
8.00	valore	
8.07		valore
8.10	valore	
8.20	valore	
8.30	valore	
8.40	valore	
8.50	valore	
9.00	valore	valore

Tabella 3.2

La mancanza di sincronizzazione temporale fra device può essere risolta con i seguenti algoritmi:

2. **tolleranza**: è possibile ignorare le differenze di tempo "trascurabili" secondo determinati criteri di decisione considerando tutte le misure i cui timestamp appartengono ad un certo intervallo di tolleranza come se fossero relative allo stesso timestamp, es:

   Δt=10min

   a. HLSignalID: 1204 TimeStamp: 12/08/2009 12:03 device1
   b. HLSignalID: 1204 TimeStamp: 12/08/2009 12:09 device2

   Entrambe le misure sono riportate in tabella a video con TimeStamp: 12/08/2009 12:00.

   Può anche accadere che si debba calcolare un KPI ad orari prefissati, attraverso l'elaborazione di HL/LL Signal diversi. Si utilizza questo approccio "a tolleranza" per pubblicare il KPIx alle ore 00:00 avendo il RTC di riferimento pari a 00:00; si definisce:

   Misura1: device 1 ha l'orologio indietro di 2 ore e, pertanto, non invia nulla essendo per tale device le 22:00 (si riceve alle ore 00:00+2 ore);

   Misura2: device2 ha l'orologio avanti di 2 ore e, pertanto, ha già inviato i dati 2 ore fa, essendo per tale device sono le 2:00;

   Misura3: device 3 ha l'orologio sincronizzato ed invia i dati (ricevuti alle ore 00:00);

   Misura4: device 4 ha l'orologio indietro di 5 min e, pertanto, deve attendere 5 min per spedire i dati (saranno ricevuti alle ore 00:05);

   Misura5: device 4 ha l'orologio avanti di 4 min e, pertanto, ha già inviato i dati 4 min fa.

Avendo una tollerenza di + 10 min e – 10 min sul calcolo del KPIx alle ore 00:00, nel KPI si tiene conto solo delle misure 3, 4 e 5.

3. **Interpolazione-like**: applicare un algoritmo di calcolo della misura al timestamp selezionato in base alla misura effettivamente acquisita, alla periodicità della stessa, alle misure precedentemente acquisite, etc.

Si può, quindi, procedere come segue:

1. ad ogni Signal viene assegnata una periodicità 'P';
2. per ogni DataView (che viene definita) viene specificato un flag "periodic" che se true indica che "ad un uguale numero di righe della tabella di output corrisponde un uguale intervallo di tempo". Questo significa che in presenza di questo flag attivo occorrerà attuare il meccanismo per uniformare i tempi o i numeri secondo uno dei due algoritmi sopra indicati:
   a. se viene flaggato "periodic" si controlla un secondo flag "ignorare piccole differenze di tempo (e di numero)?":
      i. se 'SI', si applica la correzione di tempo senza modificare il valore della misura, non la si interpola; se si riceve, ad esempio, 400 m^3/h di gas alle ore 6.07 quando si doveva ricevere la misura alle ore 6.00 (shift temporale di 7 min) si mostra sempre 400 m^3/h alle ore 6.00, senza calcolare l'esatto valore (che non sarebbe 400) delle ore 6.00;
      ii. se 'NO', si applica un algoritmo di interpolazione-like i cui range di tolleranza sono resi parametrici, tramite una tabella su DB, in funzione di alcune variabili quali: SignalID, AppSubdomain, DeviceModel, DeviceClass.

      In questo modo il calcolo della misura alle ore 6.00 (istante esatto di tempo di ricezione) è una una funzione complessa che tiene conto di vari parametri, ad esempio, del valore della misura alle 6.07 (fornita dal device con clock con deriva temporale), della periodicità, dei valori di misure acquisite nel passato, del valore max e min che la misura può assumere, della rapidità di variazione della misura, come:

      $x|_{6.00} = f(x|_{6.07}, P, \{\text{valori di LL/HL LevelSignal old}\})$

      Questi limiti possono essere, ad esempio, il limite di campioni consecutivi mancanti oltre i quali le misure prodotto non possono essere considerate valide, etc.

3.2.1.4.6.21 Allarme versus evento

Un allarme ed un evento sono sempre Signal ma, mentre un allarme è uno stato, un evento determina la transizione del sistema da uno stato ad un altro (es. ad uno stato di allarme). Si richiedono solitamente gli eventi che si sono verificati in un certo intervallo di tempo, mentre si vuole conoscere quando il sistema è in uno stato di allarme o meno. L'allarme è riferito ad un istante di tempo, mentre un evento ad un intervallo di tempo. Tipiche richieste:

- quali sono gli eventi accaduti nell'intervallo di tempo (t1, t2);
- qual è lo stato del sistema all'istante di tempo t (allarme o meno).

3.2.1.4.6.22 Eventi “Come&Go”

Gli eventi si contraddistinguono nell’essere significativi se considerati in coppia oppure individualmente e, pertanto, si classificano in due famiglie di eventi “Come&Go” e “non Come&Go”. I primi non hanno senso se considerati individualmente, ovvero, ad un evento di attivazione, se ne associa sempre uno di rientro (da cui la coppia).

Si è preferito rendere gli eventi “Come&Go” persistenti nel database, piuttosto che scolpirli in una classe specifica del codice (con la necessità di rendere definitive le modifiche ricompilando il codice), perchè se si dovesse, ad esempio, modificare il firmware di un device, l’aggiunta di una ventina di eventi si tradurrebbe nel semplice inserimento di 20 record nella corrispondente tabella. Una partizione degli eventi “Come&Go” è rappresentata dagli eventi “Attivazione-Rientro”, che vengono valutati per conoscere lo stato del sistema, cioè l’identificazione dello stato del sistema viene attuata con una ricerca su eventi di tipo “Attivazione-Rientro”. In particolare, quando ad un evento “Attivazione” ne corrisponde uno “Rientro”, significa che l’allarme è rientrato. Lo stato di allarme del sistema è determinato da un sottoinsieme degli eventi “Come&Go”, poiché essi non si riferiscono esclusivamente a stati di allarme.

Per la valorizzazione dei dettagli di un evento si propone un’opportuna correlazione fra HLSignal, ovvero, ad un HLSignal che rappresenta un evento importante, si associa un HLSignal qualificatore dell’evento per associare a quel HLSignal dei valori semanticamente rilevanti. Un’altra partizione degli eventi “Come&Go” è, quindi, rappresentata dagli eventi “dettaglio evento”.

Si suppone di considerare l’evento descritto da un HLSignalID=23 di tipo evento (cioè, non Measure, Programming Item) che segnala “Campo GSM basso”. Se la periferica volesse correlare a questa segnalazione un’informazione di dettaglio, ad es. il “valore di campo GSM attuale” che è a sua volta un HLSignal di ID, per esempio 56, ci si avvale della relazione “dettaglio evento” per associare l’HLSignalID=23 con l’HLSignalID=56. In questo modo con il qualificatore di HLSignal di ID=56 si dettaglia l’informazione sul valore del campo rispetto al HLSignal di ID=23 di segnalazione “campo GSM basso”. La correlazione tra i due HLsignal avviene in questo modo:

1. *HLSignal*: tipo=evento → “basso campo GSM” (ruolo: segnalazione)
2. *H2HRShip*: dettaglio evento
3. *HLSignal*: tipo=evento → “campo GSM” (ruolo: qualificatore di 1)

la relazione “dettaglio evento” è una relazione con verso, poichè l’informazione di dettaglio (2) dipende dall’informazione di evento (1) e, pertanto, in H2HRShipElem HLSignalID1 è l’evento, mentre HLSignalID2 è il qualificatore.

Il raggruppamento statico di HLSignal in “HLSignal di evento” ed un “HLSignal qualificatore” richiede l’individuazione di un algoritmo per correlarli. Si può proporre l’individuazione della corrispondenza sulla base del timestamp. Se si hanno due HLSignal con il medesimo timestamp significa che uno rappresenta l’evento e l’altro il qualificatore dell’evento.

Un altra tipologia di eventi “Come&Go” sono quelli che segnalano l’attivazione e il rientro di un allarme, nel seguente modo:

1. *HLSignal*: tipo=evento → “bassa soglia di odorizzante”

2. *H2HRShip*: attivazione-rientro allarme
3. *HLSignal*: tipo=evento → “soglia di odorizzante corretta”

la relazione che lega i due HLSignal è sempre H2HRShipElem. Tale tabella è statica, cioè in essa troviamo l’anagrafica delle possibili ed ammesse coppie di eventi, in questo caso attivazione-rientro. L’aggettivo statica indica che tale tabella è popolata a priori sulla base di un’opportuna documentazione. Se si desidera sapere se un dato allarme è rientrato si deve controllare la corrispondenza statica in H2HRShipElem, ma occorre controllare se in LowLevelSignal (caso di comunicazione di un singolo device) ci sia l’evento rientro con data di acquisizione maggiore di quella dell’evento di attivazione. È in LowLevelSignal che si ha la dinamicità.

L’eCentralCore esegue la lettura e scrittura di HighLevelSignal, mentre l’ECD quelle di LowLevelSignal.

3.2.1.4.6.23 Driver2DLL istanziata

Per controllare che un Driver abbia sempre una DLL istanziata, si utilizza il seguente trigger:

```
ALTER TRIGGER [dbo].[Driver2DeviceModel_InserUpdate]
   ON [dbo].[Driver2DeviceModel]
   AFTER INSERT, UPDATE
AS
BEGIN
        SET NOCOUNT ON;

        if (select count(*) from INSERTED join Driver on INSERTED.DRID = Driver.DRID where
        Driver.DDLLID is null) > 0
        begin
                raiserror('Non è possibile associare un Driver per cui non è presente la DLL', 16, 1)
                rollback transaction
        end

        declare @date as datetime
        set @date = getdate()
        update [dbo].[Driver2DeviceModel] set EffectiveSince = @date
        where DR2DMID in (select DR2DMID from INSERTED)
END
```

3.2.3 Progettazione fisica

Le definizioni delle tabelle sono state integrate con la specifica degli indici, per velocizzare il calcolo delle interrogazioni necessarie alla pubblicazione di contenuti di entità e di relazioni. Per ogni tabella di entità è stato creato un indice unico sulla colonna che costituisce la chiave primaria (sulla colonna ID) per velocizzare le interrogazioni contenenti clausole WHERE nella forma *ID=<valore>*. Sono stati definiti ulteriori indici sulle colonne, che rappresentano attributi utilizzati in interrogazioni con condizioni di selezione basate su tali attributi. Questi indici possono accelerare l'esecuzione delle interrogazioni necessarie all'estrazione dei risultati. Per una tabella "ponte", che rappresenta una relazione molti-a-molti, è stato definito un indice su ciascuna colonna, che ha il ruolo di chiave esterna per velocizzare:

- le operazioni di join tra la tabella "ponte" e le tabelle coinvolte;
- le interrogazioni che selezionano gli oggetti della tabella "ponte" associati ad uno specifico oggetto di una delle tabelle coinvolte attraverso, tipicamente, l'inclusione di una clausola WHERE nella forma *TabellaPonte.EntX_ID=<valore>*.

Per una tabella di entità con inclusa una colonna, che agisce da chiave esterna e che rappresenta una relazione uno-a-molti, è stata aggiunto un indice su tale colonna per accelerare le interrogazioni, che selezionano gli oggetti associati ad uno specifico oggetto per mezzo di una clausola WHERE nella forma *TabellaEntX.EntY_ID=<valore>*.

3.2.4 Confronto tra modello ER e modello relazionale

Nonostante un'evidente analogia tra i seguenti concetti:

- Classe e Relazione;
- Oggetto ed Ennupla;
- Attributo (ER) e Attributo (Rel);

sussiste in realtà una significativa differenza tra di loro nei due modelli:

1. **non sempre una relazione rappresenta una classe**, perché può rappresentare un'associazione o un attributo multivalore;
2. **non sempre un'ennupla rappresenta un oggetto**, perché può rappresentare una coppia di oggetti in associazione o un possibile valore di un attributo multivalore;
3. **non sempre un attributo Rel rappresenta un attributo ER**, perché può rappresentare un oggetto di un'altra classe (chiave esterna) [24, 25, 26].

4 IMPLEMETAZIONE

In tale capitolo si descrivono gli algoritmi e le strutture dati dei moduli implementati *GSMChanellManager* e *EdorDriver* della *BusinessLogic*. Successivamente vengono descritte le funzioni server di alto livello implementate tramite delle query alla base di dati.

4.1 ECD

4.1.1 GSMChannelManager

4.1.1.1 Scopo

Offre una serie di metodi per gestire la comunicazione su di un canale GSM.

4.1.1.2 Interfaccia

La classe implementa l'interfaccia *IChannelManager* (a cui devono attenersi tutti i ChannelManager) che pubblica i seguenti metodi:

- InitListeningThread();
- CheckListeningThreadVitality();
- InitMessageDispatchThread();
- CheckMessageDispatchThreadVitality();
- StopMessageDispatchThread();
- StopChannelManager();

4.1.1.3 Campi

La classe "GSMChannelManager" ha i seguenti campi:

Visibilità	Tipo primitive/Classe	Nome	Descrizione	Valore di default
private	Thread	mainthread	thread principale di esecuzione del GSMChannelManager	/
private	Bool	listeningThreadsMustContinue	flag di proseguimento per il thread principale	/
private	Thread	dataCallDispatchThread	"postino" delle chiamate dati	/
private	Thread	smsDispatchThread	"postino" degli sms	/
private	bool	dispatchThreadsMustContinue	flag di proseguimento	/

			per i “postini”	

4.1.1.4 Metodi di comunicazione GSM

4.1.1.4.1 Metodo 1: ChannelManager to device

Con tale metodo di firma:

“**public static bool SendSMS(string pSMS, string pDestNumber, GSMModem pGSMModem, int pLoggedPID, out string pReturnMessage)**”

il GSMChannelManager gestisce una richiesta di invio SMS. Esso riceve i seguenti parametri:

1. la stringa del messaggio SMS da spedire (parametro pSMS);
2. la stringa corrispondente al numero del destinatario (parametro pDestNumber coincide con la SIM);
3. l’oggetto che astrae l’entità modem GSM (parametro pGSMModem della classe GSMModem). Poiché per spedire un messaggio SMS su canale GSM si deve utilizzare un modem, occorre settarne le impostazioni di configurazione.

   La classe “GSMModem” ha i seguenti campi per astrarre le caratteristiche di configurazione:

Visibilità	Tipo primitivo/Classe	Nome	Descrizione	Valore di default
private	int	gsmmid	GSM modem ID	-1
private	string	Descr	descrizione	stringa vuota
private	string	SIMNumber	numero della SIM	stringa vuota
private	bool	isEnabled	flag per indicare se il modem attivo	false
private	string	COMName	nome della porta	stringa vuota
private	int	COMBaud	velocità in baud	0
private	int	COMParity	bit di parità	-1
private	int	COMStopBits	bit di stop	0
private	int	COMDataBits	bit di dati	0
private	SerialPort	serial	rappresenta una risorsa di porta seriale	null

4. l’utente loggato (Person ID) per controllo dei diritti d’accesso (parametro pLoggedPID);
5. “out string pReturnMessage” (out: non compila se non gli si assegna un valore) è un messaggo human readable per effettuare il debug e conoscere l’esito del metodo in certi punti critici.

Per procedere all’invio di un SMS vengono svolte in successione le operazioni ‘a’ e ‘b’ nel seguito riportate.

a. Configurazione seriale

Se al modem non è associata una porta seriale opportunamente inizializzata, ovvero al campo “serial” del parametro “pGSMModem” non è associato il riferimento ad un oggetto che astrae l’entità porta seriale, occorre invocare:

1. *pGSMModem.InitSerial()*

Nel metodo “InitSerial()” si inizializza la porta seriale del modem, compiendo uno switch-case sul valore del campo “COMParity” della classe “GSMModem” per impostare il protocollo di controllo di parità scegliendo tra:

a. **None:** non viene eseguito alcun controllo di parità;

b. **Even:** imposta il bit di parità in modo che il numero di bit sia pari;

c. **Odd:** imposta il bit di parità in modo che il numero di bit sia dispari;

d. **Default** equivale a None.

Il valore scelto viene assegnato alla variabile “parity” di tipo Parity, essendo Parity una proprietà di SerialPort con cui impostare il protocollo di controllo della parità.

Si controlla, poi, il valore del campo “COMStopBits” della classe “GSMModem” per impostare il numero standard dei bit di stop per byte. I bit di stop separano ogni unità di dati su una connessione seriale asincrona e vengono, inoltre, inviati continuamente quando non sono disponibili dati per la trasmissione. La scelta ricade su:

a. **None:** non sono utilizzati bit di stop;

b. **One:** viene utilizzato un bit di stop;

c. **Two:** vengono utilizzati due bit di stop;

d. **OnePointFive:** vengono utilizzati 1,5 bit di stop;

e. **Default** equivale a None.

Il valore scelto viene assegnato alla variabile “stopBits” di tipo StopBits.

Si istanzia, quindi, l’oggetto di nome “serial” della classe SerialPort passando al costruttore i seguenti campi riferiti all’istanza corrente della classe: COMName, COMBaud, parity, COMDataBits, stopBits. Si settano, successivamente, i seguenti campi dell’istanza “serial” ereditati dalla classe “SerialPort”:

Visibilità	Tipo primitivo/Classe	Nome	Descrizione	Valore di default
private	String	NewLine	Impostare il valore utilizzato per interpretare la fine di una chiamata ai metodi ReadLine e WriteLine. Con ReadLine() leggo, quindi, fino al valore di NewLine nel buffer di input, mentre con WriteLine() scrivo la stringa specificata ed il valore di NewLine nel buffer di output.	“\r”
private	int	ReadTimeout	Impostare il numero di millisecondi prima del timeout quando un’operazione di lettura non viene completata.	1000
private	int	WriteTimeout	Impostare il numero di millisecondi prima del	3000

			timeout quando un'operazione di scrittura non viene completata.	
private	bool	DtrEnable	Impostare un valore che attiva il segnale DTR (Data Terminal Ready) durante una comunicazione seriale (DTE -> DCE).	false
Private	bool	RtsEnable	Impostare un valore che indica se il segnale RTS (Request to Send) è attivato durante la comunicazione seriale (DTE -> DCE).	true
private	Encoding	Encoding	Impostare la codifica dei byte per la conversione del testo prima e dopo la trasmissione. Oggetto Encoding per il set di caratteri ASCII (7 bit)	Encoding.ASCII

2. pGSMModem.Serial.Open()

   Si apre una connessione sulla porta seriale (il metodo Open è sempre ereditato da SerialPort).

b. Configurazione modem

Con il comando **ATV1** viene concordata con il modem (DCE) la struttura del messaggio inviato in risposta ad un certo comando. La struttura non varia se nel messaggio c'è del testo oppure del codice, comunque nel manuale Siemens si riportano i due casi:

1. risposta con informazione testuale: **<CR><LF><text><CR><LF>**
2. risposta con codice: **<CR><LF><verbose code><CR><LF>**

Es:

si scrive nell'interfaccia il comando ATV1 (intanto si legge da seriale ciò che è stato scritto, cioè ATV1, perchè "echo attivato") e in risposta al comando si ottiene:

OK

---

perché la risposta inizia con <CR>.

Con il metodo "ReadLine()" di SerialPort si legge dal buffer di input della porta seriale fino al raggiungimento del carattere di controllo "NewLine" impostato a "\r" (vedi tabella precedente). Si procede all'elaborazione di quanto letto, sostituitendo il "\n" (perchè il framework .NET converte il carattere di controllo da "\r" a "\n") con una stringa vuota. Per eliminare dalla risposta il ritorno a capo, si controlla se ciò che è stato letto con ReadLine() ha lunghezza zero ed in tal caso non si fa nulla. Se l'informazione testuale non contiene OK si è verificato un errore nell'invio del comando ATV1 al modem.

Con il comando **ATE0** si disabilita la modalità "echo" con la quale il DCE ripete all'indietro (su seriale) i caratteri ricevuti nei comandi spediti con il DTE, per ottenere solo l'esito del comando inviato. Con il metodo "ReadLine()" di SerialPort si legge dal buffer di input della porta seriale fino al raggiungimento del carattere di controllo "NewLine" impostato a "\r" (vedi tabella precedente). Se l'informazione testuale letta dalla porta seriale non è nè OK nè ZERO si è verificato un errore nell'invio del comando ATE0 al modem.

Con il comando **AT+CMGF=1** si specifica che il formato degli SMS di input e di output è testuale. Con il metodo "ReadLine()" di SerialPort si legge dal buffer di input della porta seriale fino al raggiungimento del carattere di controllo "NewLine" impostato a "\r" (vedi tabella precedente). Se l'informazione testuale letta dalla porta seriale non contiene OK si è verificato un errore nell'invio del comando AT+CMGF=1 al modem.

Con il comando **AT+CMGS=pDestNumber** si invia un SMS al destinatario "pDestNumber", che è il campo "simnumber" di un oggetto della classe "GSMEndPoint", ovvero l'astrazione della SIM del destinatario.
I campi della classe "GSMEndPoint":

Visibilità	Tipo primitivo/Classe	Nome	Descrizione	Valore di default
private	int	gsmepid	GSM EndPoint ID	-1
private	string	simnumber	numero della SIM	null
private	bool	isEnabled	flag di abilitazione GSM EndPoint	false
private	DateTime	modifiedOn	Data di modifica	DateTime.MinValue
private	int	modifiedBy	Autore della modifica	-1

DateTime.MinValue: rappresenta il valore minimo dell'oggetto DateTime (che rappresenta un istante di tempo, in genere espresso come data e ora del giorno.). Il campo è di sola lettura.

Si crea una stringa "toSend" ottenuta dalla concatenazione di "pSMS" con il carattere di controllo "CTRL+Z", scelto come fine stringa. Si scrive, poi, tale stringa sulla porta seriale per inviare il contenuto dell'SMS.

4.1.1.4.2 Metodo 2: ChannelManager to device

Nel secondo metodo avente firma:
public static bool SendDataCall(string pDataCall, string pDestNumber, GSMModem pGSMModem, int pLoggedPID, out string pReturnMessage)
si controlla innanzitutto che al modem sia associata una porta seriale opportunamente inizializzata.
Se il campo "serial" ha riferimento null, si compiono due operazioni:

1. pGSMModem.InitSerial();
2. pGSMModem.Serial.Open();

Se il modem è, invece, connesso (seriale opportunamente configurata) si demanda all'unica istanza della classe "GSMModemManager" con il metodo "InitModem(pGSMModem, pLoggedPID)" di inizializzare il modem.

Al termine dell'inizializzazione vengono inviati al modem i seguenti comandi:

1. **ATDT<numero del destinatario>** per stabilire una chiamata dati (nel caso in esame per spedire del testo). Il modem deve rispondere con "CONNECT" se si è stabilita la connessione. In tal caso si passa automaticamente nella modalità dati;
2. si salva il parametro "pDataCall" in "toSend" e lo si scrive sulla porta seriale;
3. **+++**. Con il comando "+++", durante una chiamata dati, si forza il modem locale a cancellare il flusso dati sulla sua interfaccia ed a commutare nella modalità comandi. Questo permette di inserire comandi AT verso il modem locale (che li interpreta come comandi e non come dati da spedire all'altro modem), mentre si mantiene una connessione dati verso l'altro modem remoto. Il comando "+++" deve essere preceduto e seguito da una pausa di almeno 1'000ms per evitare che sia interpretato come dato. I caratteri "+++" devono essere inseriti in veloce successione, tutti all'interno dei 1'000ms;
4. **ATH** per disconnettersi;
5. **AT+CPAS** per conoscere lo stato del modem. Se nel contenuto letto dalla porta seriale non è presente "+CPAS: 0" il modem è ancora connesso. Tale comando serve come controllo per verificare che la chiamata dati intrapresa da "eCentral" è stata chiusa.

Si scrive, infine, sulla base dati l'evento di fine chiamata:

INSERT INTO GSMModemEvent(GSMModemEvent.GSMMID, GSMModemEvent.EventID) VALUES ('pGSMModem.GSMMID', 'pEventID');
SET @GSMMEID = ident_current('GSMModemEvent')

N.B.: ad ogni tentativo di inserimento di una riga in una data tabella, SQLServer incrementa in automatico l'identity, cioè l'enumerazione progressiva di record per quella tabella. Se un inserimento non dovesse andare a buon fine, ad esempio per l'attivazione di un trigger che fa il roolback della transazione, l'identity viene in ogni caso incrementato, per cui si possono verificare dei buchi nell'enumerazione (es. 1,2,3,5). Con "ident_current" si recupera l'ultimo valore identity generato per quella tabella.

4.1.1.4.3 Metodo 3: device to ChannelManager

Il terzo metodo presenta la seguente firma:

"public static bool CheckForIncomingCall(GSMModem pGSMModem, out string pReturnMessage, int pLoggedPID)"

Al suo interno, viene invocato sul parametro "pGSMModem" il metodo "ClearSerialBuffer()" con cui si elimina l'eventuale "sporcizia" presente nel buffer di ricezione.

Inizio descrizione ClearSerialBuffer

Con la proprietà "serial.BytesToRead" (ereditata da SerialPort) si ottiene il numero di byte di dati nel buffer di ricezione (che include il buffer di ricezione del driver seriale e il buffer interno nell'oggetto SerialPort) utilizzato per dimensionare un buffer "buf":

byte[] buf = new byte[serial.BytesToRead];

Si legge un numero di byte pari a “serial.BytesToRead” dal buffer di input della porta seriale che vengono inseriti in “buf” a partire dalla posizione zero.

Fine descrizione ClearSerialBuffer

Nella variabile d’appoggio “readTimeout” viene salvato il valore di default della proprietà della classe SerialPort “pModem.Serial.ReadTimeout”, per poter successivamente modificare il valore di quest’ultima settandola a 500ms. In pratica si riduce il timeout, che scatta quando un’operazione di lettura non viene completata entro 500ms (si riduce il timeout per quei comandi a cui il modem risponde immediatamente, mentre lo si aumenta se l’operazione richiesta richiede un tempo di attesa alto, come l’instaurazione di una connessione GSM dati).

Si legge dal buffer di input della porta seriale fino al raggiungimento del carattere di “NewLine” impostato pari a “\r”. Si elabora la stringa letta sostituendo i caratteri di controllo “\n” e “\0” con la stringa vuota (“\n” per eliminare il ritorno a capo della risposta e “\0” che è il carattere di terminazione in C corrispondente a quello nell’alfabeto GSM). Ogni operazione di lettura viene tentata tre volte, specificando nel metodo SerialReadLine il parametro “eCentralConstants.GSMP2P.ReadWriteRetries”.

I campi della classe GSMP2P sono:

Visibilità	Qualificatore	Tipo primitivo/Classe	Nome	Descrizione	Valore di default
public	const	string	CALLDONE	Stringa inviata prima della chiusura della chiamata GSM	“CALL DONE”
public	const	string	CLIP	Stringa utilizzata per definire il chiamante di una chiamata in ingresso	“CLIP”
public	const	string	CONNECT	Stringa utilizzata per confermare una connessione stabilita in remoto	“CONNECT”
public	const	string	EndOfTransmission	Fine del report trasmesso	“END OF TRASMISSION”
public	const	string	NOCARRIER	Stringa ricevuta quando una connessione è chiusa	“NO CARRIER”
public	const	int	ReadWriteRetries	Numero di tentativi per ciascuna operazione di lettura/scrittura sulla porta seriale	3
public	const	string	RECEIVED	Stringa utilizzata per confermare al device la ricezione di un	“RECEIVED”

				messaggio	
public	const	string	REP	Stringa utilizzata per richiedere un report di stato al device	“REP”
public	const	string	RING	Stringa utilizzata per identificare un chiamata in ingresso sul modem	“RING”
public	const	int	SerialReadTimeOut	Timeout utilizzato in una comunicazione seriale	4000

Se la stringa “line” letta dalla porta seriale contiene RING, viene rilevata una chiamata in ingresso. Si attribuisce a “pGSMModem.Serial.ReadTimeout” il suo valore originale, mediante l’assegnamento del valore della variabile d’appoggio “readTimeout”, preventivamente salvato.

La firma del metodo “SerialReadLine”, che è stato utilizzato, è la seguente:

private static string SerialReadLine(SerialPort pSerial, int pRetries)

Al suo interno il valore di default di ritorno viene settato ad “Empty” (*string retVal = String.Empty*); si inizializza, poi, a zero un contatore per i tentativi di lettura (*int retryCount = 0*) e si setta a *false* un flag booleano di successo lettura (*bool success = false*). L’algoritmo è costituito da un ciclo while in cui si ripete la lettura finché essa non viene compiuta con successo per un massimo di tre tentativi. Se si completa la lettura con successo, viene restituita la stringa letta dalla porta seriale.

4.1.1.4.4 Metodo 4: device to ChannelManager

La firma del quarto metodo è la seguente:

“public static bool AnswerCall(GSMModem pModem, out string pReturnMessage, int pLoggedPID)”

Nella variabile d’appoggio “readTimeout” viene salvato il valore di default (1’000ms) della proprietà della classe SerialPort “pModem.Serial.ReadTimeout”, dovendo essere settato a 6’000ms (1’000ms può essere un tempo d’attesa troppo restrittivo per un’operazione di lettura).

Per evitare di interrompere il flusso dei dati e per rispettare la sequenze logica delle operazioni (prima si identifica la chiamata e poi si acquisiscono i dati), si invia al modem il comando “AT+CLCC”, per ottenere dettagli sulla chiamata.

Il modem risponde con il seguente pacchetto:

+CLCC: <idx>,<dir>,<stat>,<mode>,<mpty>,<number>,<type>,<alpha>

<idx>: tipo integer, numero identificativo di chiamata come descritto in GSM 02.30 sottoclausola 4.5.5.1;

<dir>: 0 per chiamata originata, cioè in uscita, mentre 1 per chiamata ricevuta, cioè in ingresso;

<stat>: stato della chiamata:

0 per attiva;

1 per held;

2 per dialing (MO call);

3 per alerting (MO call);

4 per incoming (MT call);

5 per waiting (MT call);

<mode>: modalità:

0 per voice;

1 per data;

2 per fax;

3 per VOICE_AND_DATA_INVOICEMODE (da specifiche GSM, ma non usato);

4 per ALTERNATING_VOICEDATA_INVOICEMODE (da specifiche GSM, ma non usato);

5 per ALTERNATING_VOICEFAX_INVOICEMODE (da specifiche GSM, ma non usato);

6 per VOICE_AND_DATA_INDATAMODE (da specifiche GSM, ma non usato);

7 per ALTERNATING_VOICEDATA_INDATAMODE (da specifiche GSM, ma non usato);

8 per ALTERNATING_VOICEFAX_INDATAMODE (da specifiche GSM, ma non usato);

9 per unknown.

<mpty>: 0 quando la chiamata non fa parte di una chiamata in conferenza (1 viceversa);

<number>: numero telefonico in formato testo;

<type>: indirizzo di 8 cifre in formato intero; 145 quando la stringa di chiamata include il carattere di prefisso internazionale '+', altrimenti 129;

<alpha>: tipo di rappresentazione alfanumerica di un numero corrispondente all'entry trovata in agenda.

Tale comando viene utilizzato per identificare il chiamante di una GSMDataCall, mentre in un SMS il numero del chiamante è già incluso, per cui la sua identificazione è automatica.

Utilizzando come carattere separatore di campo la virgola, si procede a settare i seguenti campi della classe DataCall:

Visibilità	Tipo primitivo/Classe	Nome	Descrizione	Valore di default
private	int	gsmmid	Identificatore del modem	-1
private	int	id	Identificatore datacall	-1
private	CALLDIRECTION	dir	Direzione	CALLDIRECTION.NODIRECTION
private	CALLSTATE	stat	Stato della datacall	CALLSTATE.NOCALLSTATE
private	CALLMODE	mode	Modo	CALLMODE.NOCALLMODE
private	CALLMULTIPARTY	mpty	Conferenza (MULTIPART	String.Empty

			Y_NO = 0, MULTIPART Y_YES = 1)	

Con **ATA** si comanda di rispondere alla chiamata GSM in ingresso, per poterne recuperare i dati sulla connessione stabilita (si simula l'alzata della cornetta come quando si preme il classico pulsante con l'icona verde di un cellulare). Il numero di tentativi dell'operazione di scrittura del comando ATA sulla porta seriale è tre.

La firma del metodo "SerialWriteLine" per scrivere sulla porta seriale è riportata nel seguito:

private static bool SerialWriteLine(SerialPort pSerial, string pToSend, int pRetries)

Al suo interno si inizializza a zero un contatore per i tentativi di scrittura (*int retryCount = 0*) e si setta a *false* un flag booleano di successo scrittura (*bool success = false*). L'algoritmo è costituito da un ciclo while in cui si ripete la scrittura finché essa non viene compiuta con successo per un massimo di tre tentativi.

La scrittura consiste di tre fasi:

1. si converte la stringa da scrivere in un array di caratteri "chars";
2. si crea un array di caratteri "sendChar" di dimesione pari 1;
3. per ogni carattere 'c' in "chars" si ripete il seguente ciclo:
   a. si mette 'c' in "sendChar[0]";
   b. si scrive il carattere sulla porta seriale (pSerial.Write(sendChar, 0, 1));
   c. prima di scrivere il prossimo carattere si attendono 100ms (Thread.Sleep(100)).

Finché la stringa letta è vuota oppure contiene CLIP oppure RING (vedi tabella GSMP2P) viene ripetuta la lettura dalla porta seriale per un massimo di tre volte fino a raggiungere il valore di "NewLine" (impostato a "\r"). Si interrompe il thread per 100ms ad ogni ciclo. Solo quando la stringa letta contiene CONNECT viene rilevata la risposta, viceversa si gestisce l'errore.

Alla fine si riporta il valore di "PortSerial.ReadTimeout" al valore 1'000ms di default.

4.1.1.4.5 Metodo 5: device to ChannelManager

La firma del quinto metodo è:

"public static string ReadGSMDataCall(GSMModem pGSMModem, out string pReturnMessage, int pLoggedPID)"

Viene salvato il valore di default di "pGSMModem.Serial.ReadTimeout" in una variabile di appoggio "readTimeout". Si istanzia un oggetto "StringBuilder sb" e si setta "pGSMModem.Serial.ReadTimeout" a 60'000ms, quindi si procede alla lettura del messaggio.

Viene letta una linea alla volta dalla porta seriale invocando il metodo "ReadGSMDataCallLine"; ogni singola linea che non contiene l'"EndOfTransmission" viene aggiunta a "sb". Al termine della lettura "sb" contiene l'intera informazione testuale della chiamata dati GSM.

La firma del metodo "ReadGSMDataCallLine" è la seguente:

"private static string ReadGSMDataCallLine(GSMModem pGSMModem, int pRetries)"

Al suo interno il valore di default di ritorno viene settato ad “Empty” (*string retVal = String.Empty*); si inizializza, poi, a zero un contatore per i tentativi di lettura (*int retryCount = 0*) e si setta a *false* un flag booleano di successo lettura (*bool success = false*). L’algoritmo è costituito da un ciclo while in cui si ripete la lettura finché essa non viene compiuta con successo per un massimo di tre tentativi. Se la lettura viene completata con successo viene resituita la stringa letta dalla porta seriale. Per la lettura si utilizza un ciclo *do-while*, in cui:

si legge in modo sincrono un carattere alla volta dal buffer di input di SerialPort (pGSMModem.Serial.ReadChar()), inserendolo nella variabile char ‘c’ ed aggiungendolo alla stringa retVal da restituire

finché

il carattere “c” è diverso dai caratteri di controllo ‘\n’ e ‘\r’, e la variabile “retVal” non contiene l’ “EndOfTransmission”.

Alla fine si riporta il valore di “PortSerial.ReadTimeout” al valore 1’000ms di default.

4.1.1.4.6 Metodo 6: ChannelManager to device

Il sesto metodo ha la seguente firma:

“public static bool CloseCall(GSMModem pModem, out string pReturnMessage, bool pSendReceivedString, int pLoggedPID)”

Viene salvato il valore di “pGSMModem.Serial.ReadTimeout” nella variabile di appoggio “readTimeout”. Si setta a 3’000ms il valore del campo “pGSMModem.Serial.ReadTimeout”.

Se pSendReceivedString è true:

- viene inviata la stringa “RECEIVED” per confermare al device la ricezione di un messaggio (vengono compiuti tre tentativi dell’operazione di scrittura);
- viene ripetuta la lettura dal buffer di input della porta seriale fino al raggiungimento del valore di “NewLine” impostato a “\r”. Si elabora la stringa letta, sostituendo i caratteri di controllo “\n” e “\0” con la stringa vuota. Se la stringa elaborata è vuota non si fa nulla (“\n” per eliminare il ritorno a capo della risposta e “\0” che è il carattere di terminazione del linguaggio di programmazione C corrispondente a quello nell’alfabeto GSM).

Viene inviato il comando +++ per passare nella modalità comandi.

Si invia, poi, il comando **ATH** per disconnettere la chiamata in progresso.

Se la stringa letta dalla porta seriale:

1. è “OK”: la chiusura della comunicazione è avvenuta con successo;
2. contiene “NO CARRIER”: il dispositivo remoto ha effettuato una normale chiusura della comunicazione.

Alla fine si riporta il valore di “PortSerial.ReadTimeout” al valore 1’000ms di default.

4.1.1.5 Metodi di interfaccia del GSMChannelManager

4.1.1.5.1 Metodo 1: inizializzazione thread di ascolto

Nel metodo "InitListeningThread()" si inizializza a 'true' la variabile 'listeningThreadsMustContinue' che viene controllata dal thread principale ("mainthread") per poter proseguire o meno. Il thread principale viene creato con "mainthread = new Thread(**PoolModems**)" e viene mandato successivamente in esecuzione con il metodo "Start". Tale thread gestisce il pool di modem attivi e disponibili. La creazione di un thread con "new Thread(funzione delegata)" si realizza con un delegato.

Nella funzione "public void **PoolModems()**" si setta "mustContinue" a "true". Finché "mustContinue" è "true", il "mainthread", che esegue il codice di tale funzione, può proseguire creando una lista di thread slave denominata "threads" (List<Thread>).

Si demanda all'istanza della classe "GSMModemPool" di restituire al mainthread la lista "modemIDs" di modemID. Infine si rimuovono con il metodo "Clear()" tutti gli elementi della lista "threads" per liberarla.

Si crea un thread slave che si incarica di gestire ciascun modemID della lista "modemIDs". Il thread slave è creato con la funzione delegata "ModemPoolThreadMethod", di cui si riporta il codice nel seguito:

Thread thread = new Thread(**ModemPoolThreadMethod**)

in quanto la funzione "Start" di avvio del thread slave deve avere la stessa firma del delegato:

"private void **ModemPoolThreadMethod(object pGSMMID)**"

Si aggiunge il thread slave appena creato alla lista "threads" e, poi, viene mandato in esecuzione parametrizzato con il GSM modemID in esame:

thread.Start(gsmmid)

perchè la funzione delegata richiede come parametro un "object pGSMMID".

Su ogni elemento della lista "threads" si invoca il metodo Join() per bloccare il "mainthread" in attesa che ogni thread slave abbia termine (il "mainthread", infatti, crea tanti thread slave quanti sono i modem da gestire, quindi più di uno). In questo modo si sincronizza il mainthred con più thread slave.

Per evitare un immediato tentativo di ritestare la condizione del while (mustContinue), il mainthread viene sospeso. Si assegna a "mustContinue" il valore di "listeningThreadsMustContinue"; il thread principale entra nel ciclo while almeno una volta, perchè "mustContinue" è settata di default a true. Con questo assegnamento "mustContinue" dipende da "listeningThreadsMustContinue", cioé quando "listeningThreadsMustContinue" diventa false, anche "mustContinue" lo diventa e il "mainthread" esce dal ciclo. Il valore di "listeningThreadsMustContinue" diventa false invocando il metodo "StopChannelManager()". È immediato comprendere che stoppando il "mainthread" si interrompe la creazione di nuovi thread slave.

Nel metodo "private void ModemPoolThreadMethod(object pGSMMID)" si assegna a "modem" il riferimento all'oggetto della classe "GSMModem" restituito dal metodo

"GetGSMModemByGSMMID((int)pGSMMID)" invocato sull'istanza della classe "GSMModemPool", a cui viene fornito l'ID del modem. Se tale riferimento è diverso da "null", si demanda all'istanza della classe "GSMModemManager" con il metodo "GetSMSList(modem)" il recupero della lista di oggetti della classe "SMS" e la creazione, per ciascun elemento della lista, di un thread parametrizzato che si accolla l'onere di gestirlo. Nel seguito è riportato il codice, specificando la funzione delegata "SMSWorkingThread" in rosso:

Thread newThread = new Thread(SMSWorkingThread) e, di conseguenza, viene mandato in esecuzione passando l'oggetto "sms" della classe "SMS" al metodo "Start".

Inizio descrizione "SMSWorkingThread"

I campi della classe SMS sono:

Visibilità	Tipo primitive/Classe	Nome	Descrizione	Valore di default
private	Int	gsmmid	GSM modem ID	-1
private	Int	messageIndex	posizione del messaggio nel buffer del modem	-1
private	String	fromNumber	Specifica il numero del mittente	Stringa vuota
private	String	message	Messaggio	Stringa vuota
private	DateTime	timeStamp	Istante di tempo di ricezione/invio	DateTime.MinValue

Nella funzione delegata di firma "private void SMSWorkingThread(object pSMS)" ci si rivolge innanzitutto con il metodo "eCentralDriver FindDriverFromSMS((SMS) *pSMS*, out device)" all'istanza di "DriverIdentifier".

Inizio descrizione "FindDriverFromSMS((SMS) pSMS, out device)"

In tale metodo il DriverIdentifier si rivolge all'istanza DeviceManager con il metodo "GetDeviceFromSIMNumber(*pSMS*.FromNumber)" per identificare e, quindi, recuperare il riferimento all'oggetto della classe "Device" nota la sua SIM (FromNumber):

```
SELECT * FROM FindDeviceBySIMNumber(@pSIMnumber)

ALTER FUNCTION [dbo].[FindDeviceBySIMNumber]
(
    @pSIMNumber nvarchar(30)
)
RETURNS
@retTable TABLE
(
    DID int,
```

```
    ShortDescr nvarchar(255),
    LongDescr nvarchar(1024),
    DMID int,
    IsEnabled bit,
    ModifiedOn datetime,
    ModifiedBy int
)
AS
BEGIN
    insert into @retTable (DID, ShortDescr, LongDescr, DMID, IsEnabled, ModifiedOn, ModifiedBy)
    select D.DID, D.ShortDescr, D.LongDescr, D.DMID, D.IsEnabled, D.ModifiedOn, D.ModifiedBy
    from Device D join Device2GSMEndpoint D2E on D.DID = D2E.DID
    join GSMEndpoint DE on D2E.GSMEPID = DE.GSMEPID
    where D2E.IsLocalToServer = 0 and DE.SIMNumber like @pSIMNumber
    RETURN
END
```

Se a tale SIM è associato un device, il DriverIdentifier demanda all'istanza di "DriverManager" con il metodo GetDriverByDRID(**Driver ID**) che svolge la seguente istruzione:

```
SELECT * FROM Driver WHERE DRID = @DRID
```

in cui **DriverID** è l'oggetto restituito invocando sull'istanza di "DriverManager" il metodo "GetDRIDbyDID(pDevice.DID)", cioè recuperando il driver corrente (non revocato) corrispondente al *Device Model* del device in esame:

```
SELECT @DRID = DRID FROM GetCur_Driver2DeviceModel(@pTIME) A join DeviceModel B on A.DMID = B.DMID join Device C on B.DMID = C.DMID where A.IsRevoked = 0 and C.DID = @pDID
```

```
ALTER FUNCTION [dbo].[GetCur_Driver2DeviceModel](@pTime datetime)
RETURNS TABLE
AS RETURN
(
    Select O.*
    From Driver2DeviceModel O
    Where O.EffectiveSince <= @pTime
     and O.EffectiveSince     =          (Select Max(I.EffectiveSince)
                                         From Driver2DeviceModel I
                                         Where I.DRID = O.DRID
                                         And I.DMID = O.DMID
                                         And I.EffectiveSince <= @pTime
```

)

)

Fine descrizione "FindDriverFromSMS((SMS) pSMS, out device)"

Al termine del metodo prima descritto vengono resituiti il riferimento ad un oggetto eCentralDriver, che viene assegnato per riferimento a "eCentralDriver ecd", e il riferimento all'oggetto parametro "out device" assegnato ad un oggetto omonimo della classe "Device".

Il "DriverManager" pubblica i seguenti metodi:

1. gestione Driver lato DB con i metodi:
    a. SaveDriver;
    b. BindDriver2DeviceModel;
    c. DeleteDriver;
2. caricamento Driver con i metodi:
    a. GetDriverByDRID (recupera un eCentralDriver dato il DriverID);
    b. GetDriverDLLByDRID (recupera un eCentralDriverDLL dato il DriverID);
    c. GetDRIDbyDID (recupera il DriverID dato il DeviceModelID del DeviceID);
3. mapping da una data tabella DB alla classe corrispondente con due metodi DataRowToDataModel diversi:
    a. Driver;
    b. DriverDLL;
4. gestione delle istanze di Driver con i metodi:
    a. CreateDriverInstance;
    b. CheckDriverInstance;
5. creazione delle stringhe delle query fisse (per non doverle riscrivere ogni volta) con i metodi:
    1. SQLDeleteDriverRecord;
    2. SQLInsertDriver2DeviceModelRecord;
    3. SQLInsertDriverRecord;
    4. SQLSelectDriver2DeviceModelRecord;
    5. SQLSelectDriverDLLRecord;
    6. SQLSelectDriverRecord;
    7. SQLUpdateDeviceRecord.

I campi della classe "eCentralDriver" sono:

Visibilità	Tipo primitive/Classe	Nome	Descrizione	Valore di default
private	int	drid	Driver ID	-1
private	string	shortDescr	Breve descrizione	null
private	string	longDescr	Descrizione lunga	null
private	string	driverVersion	Versione del driver	null

private	bool	isEnabled	Flag di abilitazione driver	false
private	int	modifiedBy	Autore della modifica	-1
private	DateTime	modifiedOn	Data di modifica	DateTime.MinValue
private	int	ddllid	Driver DLL ID	-1

Solo se i riferimento di "*ecd*" e di "*device*" non sono nulli, con il metodo "GetDriverInstancebyDRID(**ecd.DriverID**)" viene demandato all'istanza di "DriverPool" di restituire il riferimento all'oggetto "eCentralDriverDLL", cioè il riferimento della DLL del driver avente quell'ID, denominato "*newecddll*".

Ad esempio, se il DriverID vale 6:

*select * from DriverDLL where DDLLID = (select DDLLID from Driver where DRID = 6)*

si ottiene il record di figura 4.1:

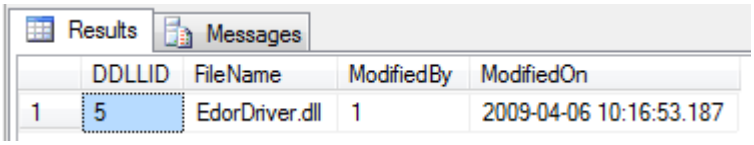


Figura 4.1: risultato

Si controlla, poi, se il driver è già stato istanziato per non doverlo ricaricare ottimizzando l'occupazione di memoria. Lo stato della classe "DriverPool" è fornito da un hashtable di nome "drivers" in cui sono contenute le DLL istanziate fino a quel momento. In "DriverPool":

1. viene demandato all'istanza di "DriverManager" con il metodo "GetDriverDLLByDRID(**ecd.DriverID**)" il compito di recuperare il riferimento all'oggetto "eCentralDriverDLL" corrispondente.

I campi della classe "eCentralDriverDLL" sono:

Visibilità	Tipo primitive/Classe	Nome	Descrizione	Valore di default
Private	int	ddllid	Driver DLL ID	-1
Private	string	filename	Nome del file DLL	Null
Private	int	modifiedBy	Autore della modifica	-1
Private	DateTime	modifiedOn	Data di modifica	DateTime.MinValue
Private	object	dllIstance	Driver DLL ID	Null

2. si ricerca nell'hashtable "drivers" l'elemento DLL di chiave DDLLID (Driver DLL ID), in quanto la sua presenza indica che l'oggetto "eCentralDriverDLL" cercato è già stato istanziato, altrimenti:

a. all'istanza del DriverManager viene demandato il compito di creare un'istanza della DLL del Driver, mentre ad un'istanza della classe "Assembly" (che rappresenta un blocco di compilazione di un'applicazione Common Language Runtime riutilizzabile) di caricare a tempo d'esecuzione l'oggetto della DLL (campo filename dell'oggetto "eCentralDriverDLL"). Si cerca nelle dichiarazioni di tipo definite nell'oggetto della classe Assembly se è presente l'interfaccia "DriverBase", in quanto la DLL deve essere compatibile con la definizione "Driverbase" ereditata da tutti i Driver; in caso affermativo, viene istanziata con il nome "driverInstance" di tipo "object";
b. all'istanza del DriverManager si demanda il controllo della compatibilità dell'oggetto passato, cioè del "driverIstance", con almeno uno di questi tipi:
    i. IDriver2Core;
    ii. IDriver2Channel;
    iii. DriverBase.

    In caso affermativo, si assegna per riferimento "driverIstance" alla proprietà *dllIstance* dell'oggetto "eCentralDriverDLL".
c. viene aggiunta, infine, la DLL all'hashtable.

Si procede, poi, ad indicare all'oggetto Driver che ciò che riceverà (un oggetto della classe "IncomingDataConnectingArgument") ha la struttura di un messaggio SMS, assegnando tale SMS al campo SMS dell'oggetto "idca" della classe "IncomingDataConnectingArgument" (→ idca).

I campi della classe "IncomingDataConnetionArgument" sono:

Visibilità	Tipo primitive/Classe	Nome	Descrizione	Valore di default
private	Socket	TCPSocket	Socket TCP	/
private	GSMModem	GSMModem	Modem GSM	/
private	SMS	SMS	SMS	/
private	FTPWorkingSet	FTPWorkingSet	Pacchetto FTP	/
private	MailWorkingSet	MailWorkingSet	Pacchetto Mail	/

Poiché "newecddll" è l'oggetto *DriverX.dll* caricato a tempo d'esecuzione, occorre invocare su di esso il metodo "Driver2Channel" per poter invocare correttamente il metodo "ManageIncomingDataConnection(idca, versione_del_driver)", con cui gli si passa l'SMS. Si demanda, poi, all'oggetto *DriverX.dll* di processare i dati in base alla semantica che conosce e alla versione richiesta.

Fine descrizione "SMSWorkingThread"

Nel caso, invece, di esito positivo del controllo "CheckForIncomingCall", la gestione di una chiamata in ingresso al modem viene attuata tramite la creazione di un apposito thread parametrizzato non più con il metodo "SMSWorkingThread", ma con il metodo "DataCallWorkingThread". L'avvio del thread ha come parametro un oggetto GSMModem modem. Si riporta nel seguito il codice:

```
Thread newThread = new Thread(DataCallWorkingThread)
newThread.Start(modem)
```

Inizio descrizione "DataCallWorkingThread"

La funzione delegata "private void DataCallWorkingThread(object pGSMModem)" è analoga alla funzione "SMSWorkingThread", ma con due sostanziali differenze:

- a monte dell'identificazione del Driver si attende che il metodo "AnswerCall" restituisca vero (quindi anziché "FindDriverFromSMS((SMS) pSMS" si ha "FindDriverFromDataCall(DataCall pDataCall)");
- l' "IncomingDataConnectionArgument idca" da gestire non è più un SMS, ma un GSMModem, infatti si assegna al campo "GSMModem" di "idca" il parametro della funzione delegata "(GSMModem)pGSMModem" opportunamente castizzata.

Fine descrizione "DataCallWorkingThread"

In caso contrario l'istanza di "GSMModemPool" rilascia la risorsa modem.

4.1.1.5.2 Metodo 2: inizializzazione del thread gestore dei thread "postini" di SMS e di DataCall

Nel metodo di firma "public bool InitMessageDispatchThread()" viene settato a true il flag per il proseguimento del "dispatchThreads" e creato un thread "smsDispatchThread" parametrizzato con funzione delegata "SMSDispatchThreadMethod"; in tale funzione si crea un hashtable "toBeUpdate". Finchè "mustContinue" è true, con il metodo "DeleteCompletedOutgoingSMS" si demanda all'istanza di "CommunicationLogManager" il compito di cancellare dalla tabella "LogSMS" tutti i record escludendo quelli con "Status" pari a 1, cioè quelli "Enqueued" (accodati).

DELETE LogSMS WHERE LogSMS.Status <> 1

Con il metodo "GetOutgoingSMSList" si demanda all'istanza di "CommunicationLogManager" il compito di recuperare la lista "smsList" di LogSMS che, dopo l'operazione di delete precedente, è costituita di soli messaggi in coda da spedire.

*SELECT * FROM LogSMS WHERE LogSMS.Status=1*

I campi della classe "LogSMS" sono:

Visibilità	Tipo primitive/Classe	Nome	Descrizione	Valore di default
private	int	logSMSID	Identificatore del log SMS	-1
private	int	did	Device ID	-1
private	string	messageText	Testo del messaggio	null
private	DateTime	logTime	Istante di tempo del log	DateTime.MinValue
private	int	modifiedBy	Autore della modifica	-1
private	int	simnumber	Numero della	null

			SIM	

Se la lista di “LogSMS” non è vuota, si invoca il metodo “GetModem()” sull’istanza di “GSMModemPool”; si recupera un’istanza di GSMModem (attiva e disponibile) e per ogni elemento “LogSMS” della lista “smsList” si salva nell’elemento dell’hashtable “toBeUpdated” alla posizione specificata dalla chiave “sms.LogSMSID”:

1. lo stato “Sent”, se l’esito della spedizione dell’sms con il metodo “SendSMS” (passando il numero della SIM del destinatario presente nell’oggetto LogSMS) è positivo;
2. lo stato “Error”, in caso contrario.

Terminato l’utilizzo del modem si demanda all’istanza di “GSMModemPool” di rilasciarlo. Se è true il valore di ritorno del metodo “UpdateOutgoingSMSStatus”, invocato sull’istanza del “CommunicationLogManager” passando come parametro principale l’hashtable “toBeUpdated”, si eliminano tutti gli elementi dell’hashtable “toBeUpdated”.

In “UpdateOutgoingSMSStatus” sostanzialmente si scorrono gli elementi di “toBeUpdated” sulla chiave “sms.LogSMSID”, controllandone lo stato. Se lo stato è “Sent”, viene aggiunto in uno StringBuilder “sb” il “LogSMSID” dell’elemento in esame. Lo “sb” conterrà l’elenco di elementi di “toBeUpdate” (separati dalla virgola) che hanno stato pari a “Sent”. Si procede, infine, all’aggiornamento dello stato “accodato” in “spedito” dei record corrispondenti della tabella “LogSMS”.

UPDATE LogSMS SET LogSMS.Status = ‘SENT’ WHERE LogSMS.LogSMSID IN @sb

Viene ripetuta la procedura, prima descritta, per gli elementi con stato “Error”.

Prima di testare nuovamente la condizione “dispatchThreadsMustContinue”, il thread viene fermato per 8ms, corrispondente all’intervallo di tempo tra due sue esecuzioni consecutive.

Si manda in esecuzione l’ “smsDispatchThread”.

Si crea e si manda in esecuzione un thread “dataCallDispatchThread” parametrizzato con funzione delegata “DataCallDispatchThreadMethod”. Nella funzione “DataCallDispatchThreadMethod” si compiono le medesime elaborazioni trasponendole da un oggetto della classe LogSMS ad uno della classe LogGSMCall. I dati vengono spediti, invocando il metodo “SendDataCall” anzichè “SendSMS”.

I campi della classe LogGSMCall sono:

Visibilità	Tipo primitive/Classe	Nome	Descrizione	Valore di default
private	int	logGSMCallID	Identificatore del log della chiamata GSM	-1
private	int	Did	Device ID	-1
private	string	messageText	Testo del messaggio	null
private	string	Simnumber	Numero della SIM	null
private	DateTime	logTime	Log del tempo	DateTime.MinValue
private	Int	modifiedBy	Autore di modifica	-1

4.1.1.5.3 Metodo 3: arresto del thread gestore dei thread “postini” di SMS e di DataCall

Nel metodo “StopMessageDispatchThread()” viene settato “dispatchThreadsMustContinue = false”; pertanto se il riferimento di “smsDispatchThread” non è null e lo stato del thread è “WaitSleepJoin”, lo si abortisce. Se il riferimento di “dataCallDispatchThread” non è null e lo stato del thread è “WaitSleepJoin” lo si abortisce.

4.1.1.5.4 Metodo 4: arresto del GSMChannelManager (mainthread)

Nel metodo “StopChannelManager()” viene settato “listeningThreadsMustContinue = false” e se lo stato di “mainthread” è “WaitSleepJoin” lo si abortisce. Viene settato “dispatchThreadsMustContinue = false”; pertanto se il riferimento di “smsDispatchThread” non è null e lo stato del thread è “WaitSleepJoin”, lo si abortisce. Se il riferimento di “dataCallDispatchThread” non è null e lo stato del thread è “WaitSleepJoin” lo si abortisce.

4.1.1.5.5 Metodo 5: controllo dello stato del servizio del mainthread

Per controllare lo stato del servizio del mainthread è disponibile il metodo CheckListeningThreadVitality() con cui si determina se il thread è in esecuzione, quando il suo riferimento non è null.

4.1.1.5.6 Metodo 6: controllo dello stato dei thread slave di gestione dei log di SMS e DataCall

Nel metodo “CheckMessageDispatchThreadVitality() si controlla l’esecuzione di entrambi i thread postini “smsDispatchThread” e “dataCallDispatchThread”.

4.1.1.6 Classi correlate al GSMChannelManager

4.1.1.6.1 GSMModemManager

Scopo: offre funzioni di gestione dei modem e dei loro eventi sulla base di dati e funge da mediatore tra il GSMModemPool e tale base.

Si specifica che la classe è Singleton, affinchè di tale classe esista una sola istanza una volta invocata la property "Istance".

Inizio metodi per la base di dati

Nel metodo:

"public GSMModem GetModem(int pGSMMID, int pLoggedPID, SqlTransaction pDbTran)"

vengono settati i campi di un oggetto della classe "GSMModem" attraverso il loro recupero dalla tabella omonima nella base dati con una query sul GSMMID.

*SELECT * FROM GSMModem WHERE GSMMID = pGSMMID*

Nel metodo:

"public List<GSMModem> GetAllModems(bool pOnlyEnabledModems, int pLoggedPID, SqlTransaction pDbTran)"

si recupera la lista di tutte e sole le istanze di "GSMModem" attivi.

*SELECT * FROM GSMModem WHERE GSMModem.IsEnabled=1*

Nel metodo:

"public bool SaveModem(GSMModem pGSMModem, int pLoggedPID, SqlTransaction pDbTran)"

è permessa la memorizzazione di un'istanza "GSMModem" in modo perstente nella tabella omonima nella base di dati.

Nel metodo:

"public static bool WriteModemEvent (EC.GSMModemEvent pEventID, GSMModem pGSMModem, int pLoggedPID, SqlTransaction pDbTran)"

viene associato un evento (es. EndCall) riferito ad un dato oggetto "GSMModem" nella tabella "GSMModemEvent" della basi dati. Lo scopo è quello di determinare il costo del traffico GSM durante la fase di sviluppo degli eventi.

INSERT INTO GSMModemEvent (GSMMID, EventID) VALUES (@GSMMID, @EventID);
SET @GSMMEID = ident_current('GSMModemEvent')

Fine metodi per la base di dati

Nel metodo:

“public bool InitModem(GSMModem pGSMModem, int pLoggedPID)”

si invoca “InitSerial()” sul parametro pGSMModem per inizializzare la porta seriale del modem ed aprire una nuova connessione.

Si procede successivamente:

1. ad abilitare la risposta in formato testo con il comando ATV1;
2. a disabilitare l’echo con il comando ATE0;
3. a disabilitare la modalità di risposta automatica per chiamate in ingresso con il comando ATS0=0;
4. ad indicare la priorità di salvataggio degli SMS (memoria interna vs SIM) con il comando Siemens AT^SSMSS=0;
5. a configurare la memoria preferita per lo storage degli SMS con il comando AT+CPMS=MT,MT,MT; in particolare MT rappresenta l’insieme della memoria interna del modem e della SIM. Il primo MT è la memoria utilizzata per ricevere e cancellare i messaggi; il secondo MT è usato per scrivere e spedire i messaggi; il terzo MT è usato per specificare dove verranno memorizzati gli SMS se l’instradamento del modem non è settato;
6. a settare l’RTS (Request To Send) con il comando AT\Q3 per attuare un controllo di flusso hardware (per prevenire la perdita di dati o evitare errori quando, durante una chiamata dati, il dispositivo mittente spedisce i dati ad una velcotà maggiore della velocità di ricezione che il destinatario è in grado di supportate. Quando il buffer di ricezione si satura, il dispositivo in ricezione deve essere in grado di stoppare il mittente finché non si libera spazio nel suo buffer.

Nel metodo:

“public int ModemSignalQuality(GSMModem pGSMModem, int pLoggedPID, out int pBitErrorValue)”

viene controllata la qualità del segnale con il comando AT+CSQ (il modem fornisce l’intensità del segnale ricevuto ed il bit error rate del canale dal mobile equipment) e l’output viene formattato come numero intero.

Nel metodo:

“public string ModemSignalQualityText(GSMModem pGSMModem, int pLoggedPID)”

viene controllata la qualità del segnale (come il metodo “ModemSignalQuality) e l’output è in formato testo.

Nel metodo:

“public List<SMS> GetSMSList(GSMModem pGSMModem, bool pDeleteMessages, int pLoggedPID)”

viene creata una lista di oggetti della classe SMS da restituire al chiamante e denominata ***retVal***. I campi di un messaggio SMS (data e tempo) sono separati dai seguenti caratteri di controllo:

1. *messageItemSeparator* è il carattere ‘,’;

2. *dateItemSeparator* è il carattere ‘/’;
3. *timeItemSeparator* è il carattere ‘:’;

Per il trim delle sottostringhe costituenti il corpo (body) di un SMS si fa riferimento al carattere di controllo ‘\r’. Si eseguono non più di 4 volte le seguenti operazioni:

1. si invia al modem il comando **AT+CMGF=1** per specificare il formato testuale dei messaggi SMS e si determina l’esito di tale invio, attendendo che il modem risponda con OK;
2. si sospende il thread per 1 secondo in modo da gestire correttamente la latenza del modem;
3. si invocano i metodi *DiscardInBuffer()* e *DiscardOutBuffer()* sulla proprietà *Serial* di *pGSMModem*;
3. si invia al modem il comando **AT+CMGL=REC UNREAD** per configurarlo a restituire tutti i messaggi SMS non ancora letti e memorizzati al suo interno;
4. viene memorizzato in una stringa denominata *tempstring* tutto il contenuto letto dalla porta seriale fino al raggiungimento di OK o di ERROR. Se viene letto ERROR si ritorna al punto 1.

Per identificare i singoli SMS all’interno del contenuto letto dalla porta seriale (cioè in *tempstring*), ci si avvale della struttura dati di un SMS, che si compone di due parti: un’intestazione (+CMGL) e un corpo (contenuto informativo). Le due parti sono separate dal carattere di controllo ‘\r’. Quando si incontra **+CMGL** inizia il messaggio successivo.

Finchè *tempstring* è non vuota e non inizia con OK (se *tempstring* coincidesse con il solo OK vorrebbe dire che l’invio del comando **AT+CMGL=REC UNREAD** è avvenuto con successo e non ci sono SMS non letti in memoria):

1. si costruisce un oggetto di nome *sms* della classe SMS;
2. individuata la fine dell’intestazione di un SMS con la prima occorrenza del carattere di controllo ‘\r’, si procede a suddividerla in tante sottostringhe, mediante il separatore “messageItemSeparator”, ottenendo in questo modo un array di stringhe denominato “items”. Si procede, poi, all’analisi dell’intestazione, per settare i campi corrispondenti dell’oggetto *sms*:
    1. il campo ***MessageIndex*** di *sms* (posizione del messaggio SMS nella memoria del modem GSM) viene settato con il valore ottenuto dalla conversione in intero dell’elemento *“items[0]”*, dopo averne sostituito la stringa “+CMGL: ” con “” (stringa vuota);
    2. il campo ***FromNumber*** di *sms* viene settato con il valore dell’elemento “items[2]” dopo averne sostituito la stringa “\”” con “”(stringa vuota);
    3. si crea un array di stringhe (denominato *date*) attraverso lo *split* di “items[4]” su “dateItemSeparator” dopo averne sostituito la stringa “\”” con la stringa vuota;
    4. si crea un array di stringhe (denominato *time*) attraverso lo *split* su “timeItemSeparator” della sottostringa ottenuta a partire dalla posizione zero fino alla posizione del carattere ‘+’ di “items[5]”;
    5. si utilizzano gli array dei punti 3-4 per settare il campo *TimeStamp* di *sms*. Nel seguito è mostrato il codice per maggiore chiarezza:

       TimeStamp = new DateTime(Convert.ToInt32(date[0]) + 2000, Convert.ToInt32(date[1]), Convert.ToInt32(date[2]), Convert.ToInt32(time[0]), Convert.ToInt32(time[1]), Convert.ToInt32(time[2]));

3. l'intestazione analizzata al punto 2 viene tolta dalla *tempstring*, che contiene il rimanente contenuto letto dalla porta seriale. La successiva parte del SMS da analizzare, cioè il corpo, viene identificata con il carattere di controllo '\r'. Si rimuovono, poi, le occorrenze iniziali e finali di '\n' e di '\r' da *tempstring*. Il corpo del SMS può essere composto di più righe di testo separate dal carattere di controllo '\r'. Finchè *tempstring* non inizia con **+CMGL** (intestazione di nuovo SMS) o **OK** (fine messaggio SMS in esame), ogni sottostringa di *tempstring* compresa tra due occorrenze del carattere di controllo '\r' viene aggiunta ad un contenitore denominato *body* di tipo *string*, terminandola con il carattere di controllo '\n'. *tempstring* viene ridotta ad ogni identificazione di una stringa costituente il corpo del SMS in esame. Il campo ***Message*** di *sms* viene infine settato con *body*;
4. il campo ***GSMMID*** di *sms* viene settato con quello del parametro passato "pGSMModem.GSMMID";
5. poiché il metodo restituisce una lista di SMS, al termine del parsing, l'SMS viene aggiunto alla lista retVal da restituire.

I passi da 1 a 5 vengono, pertanto, ripetuti fino a quando la stringa *tempstring* (contenente originariamente tutto il contenuto letto dalla porta seriale, ovvero uno o più SMS), riducendosi di volta in volta, si azzerra o si riduce al solo OK, che indica la fine del contenuto letto dalla porta seriale in risposta al comando **AT+CMGL=REC UNREAD**.

Se il parametro "pDeleteMessages" è 'true', allora per ogni SMS nella lista "retVal" si invoca:

DeleteSMS(pGSMModem, sms.MessageIndex, pLoggedPID);

con cui si spedisce al modem il comando **AT+CMGD=<MessageIndex>** per cancellare dalla sua memoria il messaggio nella posizione specificata da "MessageIndex".

Sono state implementate, infine, le properties per la creazione delle stringhe delle query fisse (per non doverle riscrivere ogni volta):

1. SQLSelectGSMModemRecord;
2. SQLSelectALLGSMModemRecord;
3. SQLInsertGSMModemRecord;
4. SQLUpdateGSMModemRecord;
5. SQLInsertGSMModemEventRecord.

4.1.1.6.2 GSMModemPool

Si può avere una sola istanza della classe in esame. Tale istanza ha un unico campo "modems" di tipo hashtable, i cui elementi sono oggetti della classe "GSMModemPoolItem"; "modems" rappresenta la pila di modem (quelli attivi, ciascuno dei quali può essere disponibile o meno, cioè locked).

Il costruttore "GSMModemPool()" invoca la funzione "BuildModemPool(EC.BuiltinPrincipals.Admin.PID)" di firma:

public void BuildModemPool(int pLoggedPID)

nella quale si acquisisce il lock dell'hashtable "modems", per garantire un accesso mutuamente esclusivo da parte dei thread.

Si procede alla cancellazione di tutti gli elementi nell'hashtable "modems", per poi demandare all'istanza della classe "GSMModemManager" il compito di restituire la lista di oggetti "GSMModem" attivi (chiamata "tmpModems"), avendo specificato come true il parametro "pOnlyEnabledModems".
Per ogni elemento della lista si controlla se il modem è connesso, cioè se il riferimento della sua porta seriale non è null; in caso affermativo viene aperta una nuova connessione, altrimenti essa deve essere inizializzata.
Si crea, poi, un oggetto della classe "GSMModemPoolItem" denominato "poolItem".
La classe GSMModemPoolItem ha due campi:

Visibilità	Tipo primitive/Classe	Nome	Descrizione	Valore di default
private	GSMModem	gsmmodem	Astrazione dell'entità modem del mondo reale	null
private	bool	locked	Flag per verificarne la disponibilità	false

Si settano i campi dell'oggetto della "pila":

- con la property GSMModem (invocabile dall'esterno), che assegna al campo "gsmmodem" il riferimento dell'elemento in esame della lista "tmpModems";
- con la property Locked viene settato a false il campo "locked";

Viene, infine, aggiunto l'oggetto "poolItem" all'hashtable "modems" con chiave pari al GSMMID (identificatore del modem GSM).

Nel metodo "public void RebuildModemPool(int pLoggedPID)" si acquisisce il lock dell'hashtable 'modems', si crea un hashtable "newList" e si demanda all'istanza della classe "GSMModemManager" il compito di restituire la lista di "GSMModem" modem attivi, chiamata "tmpModems", specificando come true il parametro "pOnlyEnabledModems". Per ogni elemento della lista viene creata un'istanza "poolItem" della classe "GSMModemPoolItem" e, poi, se ne settano i campi; in particolare:

- con la property GSMModem (invocabile dall'esterno) si assegna al campo "gsmmodem" il riferimento dell'elemento in esame della lista "tmpModems";
- per il campo "locked" si controlla la presenza dell'oggetto con chiave "gsmmodem.GSMMID" nell'hashtable "modems"; in caso affermativo si setta il campo "locked" al valore del campo "locked" dell'oggetto recuperato con la chiave GSMMID, altrimenti lo si setta a false.

Si aggiunge, infine, l'oggetto "poolItem" con chiave GSMMID all'hashtable "newList".
All'uscita del ciclo, ma ancora all'interno del blocco *lock(modems)*, su ogni elemento della classe "GSMModem" si esegue il seguente assegnamento per riferimento:
modems = newList
affinché il riferimento di "modems" sia quello di "newList", in quanto il metodo è un rebuild della pila di modem attivi.

Nel metodo “public GSMModem GetModem()” si acquisisce il lock su “modems”. Tale metodo permette di recuperare il primo modem disponibile (da “modems”), attivo e non locked. Viene settato “locked” a true con la property Locked per segnalare che tale modem risulta essere ora occupato.

Con il metodo “public bool ReleaseModem(GSMModem pModem)” si effettua la ricerca nell’hashtable “modems” dell’elemento di chiave “pModem” e, una volta individuato, se ne setta a true il campo “locked”, per segnalare che è tornato disponibile. L’accesso a “modems” è sempre mutuamente esclusivo.

Con il metodo “public GSMModem GetGSMModemByGSMMID(int pGSMMID)” si impegna il GSMModemPoolItem (elemento dell’hashtable “modem”) con chiave pGSMMID (specificata nei parametri) e se ne setta a true il campo “locked”, per segnalare che non è più disponibile. L’accesso a “modems” è sempre mutuamente esclusivo.

Con il metodo “public List<int> GetListGSMModemInPool()” viene restituita la lista di GSMModemID dei modem contenuti nel pool di modems, cioè nell’hashtable “modems”.

4.1.2 EdorDriver

4.1.2.1 Scopo

Gestire i dati spediti da e verso i device che appartengono al DeviceModel Edor. Per poter svolgere la tesi si è implementato il protocollo SMS Edor di comunicazione, che richiede l'interazione del driver con il GSMChannelManager. Per garantire in ogni caso l'interpretazione del contenuto inviato da un device del modello Edor su di un diverso canale di comunicazione, i metodi corrispondenti verranno sempre implementati in questa classe. L'Edor Driver in questo modo interagirebbe con il ChannelManager associato, oltre che con il GSMChannelManager. Si ricorda che un Driver è una classe che espone dei metodi.

4.1.2.2 Descrizione

Edor comunica via sms:

1. in uscita per: telelettura, allarmi, reportistica;
2. in ingresso per: programmazione e telecomandi.

La comunicazione avviene normalmente tra eCentral ed Edor, con l'unica eccezione degli SMS di allarme, inviati direttamente dal device al cellulare del reperibile.

L'identificazione del Driver deve essere eseguita a monte del processo di acquisizione dei dati da device per poter contattare il Driver competente della corretta interpretazione semantica dei dati ricevuti da eCentral.

Il canale GSM è punto a punto, per cui si deve eseguire una comunicazione per un device alla volta.

Nel seguito viene proposta una sintesi delle varie modalità di comunicazione:

Da device a eCentral:

Tipo A: SMS con i dati di consumo e di diagnostica, una volta al giorno;

Tipo B: SMS con misura singola, su evento (operatore in campo);

Tipo C: SMS con dati di configurazione, su evento, tipicamente su richiesta;

Tipo D: SMS con dati di taratura, su evento, tipicamente una volta ogni 15-20 giorni;

Tipo E: SMS di report eventi;

Tipo F: SMS di report totalizzatori "vita" cella, segue sempre un tipo C;

Tipo ACK: SMS di conferma configurazione, segue sempre la ricezione di un tipo CFG.

Da device a reperibile:

Tipo ALL: SMS di allarme TO ALTO e TO BASSO, su evento; tale informazione viene inviata solo al reperibile, mentre ad eCentral viene comunicata nel successivo SMS di tipo A (tipicamente il giorno dopo).

Da eCentral a device:

Tipo CFG: SMS di configurazione/telecontrollo; tipicamente viene ricevuto dal device Edor il giorno successivo all'invio ed è utilizzato per modificare la programmazione o richiedere una connessione diretta.

Tabella 4.1: "Tipi di comunicazione"

Tipo	Funzione	I/O	Destinatari	Contenuto
A	Misure giornaliere	OUT	eCentral	*Invio: AUTOMATICO giornaliero* Dati relativi alle quattro misurazioni eseguite nell'ultima giornata. Viene inviato in automatico una volta al giorno, ad un orario programmabile (default 7:00)
B	Misura estemporanea	OUT	eCentral	*Invio: AUTOMATICO ogni <n> ore (opzionale) o SU RICHIESTA* Dati relativi all'ultima misurazione eseguita. Può essere inviato in automatico dopo ogni misura (tranne dopo l'ultima, che è invece seguita da un sms di tipo A), o può essere richiesto tramite un sms di comando (tipo N), oppure può essere comandato dall'operatore in campo
C	Parametri	OUT	eCentral	*Invio: SU RICHIESTA* Report dei parametri di configurazione di Edor; tra essi l'ID, i numeri del centro e del reperibile e le soglie di allarme. Viene inviato solo su richiesta, da effettuarsi tramite un sms di comando (tipo N)
D	Parametri	OUT	eCentral	*Invio: SU RICHIESTA o AUTOMATICO (dopo autotaratura)* Report dei parametri di configurazione di Edor; tra essi l'ID, i numeri del centro e del reperibile e le soglie di allarme. Viene inviato su richiesta, da effettuarsi tramite un sms di comando (tipo N), oppure, sul modello dotato di Sampling System, viene inviato in automatico dopo un'autotaratura del device
E	Eventi	OUT	eCentral	*Invio: AUTOMATICO (dopo tipo A o tipo B, solo se attivo)* Report degli ultimi 10 eventi dell'apparecchiatura. La spedizione è automatica solamente se tale funzionalità è abilitata da firmware, in caso negativo il messaggio non viene mai spedito
F	Totalizzatori	OUT	eCentral	*Invio: AUTOMATICO (dopo tipo C)* Report dei totalizzatori di misura, come numero di misure eseguite dalla cella, media delle temperatura, data di attivazione della cella e altro ancora
ACK	Acknoledgment configurazione	OUT	Edor	*Invio: AUTOMATICO (dopo ricezione sms di programmazione)* Indica che l'Edor ha ricevuto l'sms di programmazione e lo ha elaborato correttamente
ALL	Allarme	OUT	Reperibile	*Invio: AUTOMATICO (dopo allarme)* Sms testuali che riportano l'ID del dispositivo Edor e il tipo di allarme che si è verificato. Vengono inviati dopo ogni campionamento, se la condizione di allarme si verifica due volte consecutivamente. Gli allarmi riconosciuti sono: supero soglia massima e sotto soglia minima
CFG	CFG Program., Comandi	IN	Edor	*Invio: QUANDO NECESSARIO (da eCentral a Edor)* Una serie di istruzioni di programmazione e di comando che vengono inviate di norma da eCentral all'Edor

N.B.

La classe prevede un tipo di dato enumerazione denominato **Command_type**, in cui ad ogni tipo di comando (A, B, C, D, E, F, G, H, I, J, K, L, M, N, T, O, NUM) si associa un numero progressivo a partire da 0 (A→0, B→1, etc.). "NUM" indica il numero totale di comandi. Tale parametro si utilizza per eseguire un'iterazione dinamica sui tipi di comando, per evitare di dover modificare nel codice di tutte le iterazioni il valore statico MAX del numero di comandi (es. 16), su cui si itera come mostrato nel seguente esempio: *for (int i=0; i<16; i++)).*

Con NUM (es. for (int i=0; i<COMMAND_TYPE.NUM; i++)), viceversa, il codice si adatta automaticamente al cambiamento.

4.1.2.3 Interfaccia

La classe implementa i metodi della interfaccia *IDriver2Channel*:

- LowLevelData **ManageIncomingDataConnection**(IncomingDataConnectionArgument pIDCI, string pDriverVersion);
- bool **LowLevelDataValidation**(LowLevelData pLowlevelData, string pDriverVersion);
- bool **ProcessDeviceRowData**(LowLevelData pLowlevelData, string pDriverVersion);

e quelli di *IDriver2Core*:

- bool **RequireDataToDevice**(int pCommID, string pMessage);
- bool **ProgramDevice**(int pCommID, List<CommunicationData> pCdList);

(A tali interfacce devono attenersi tutti i Driver)

La classe EdorDriver estende, infine, la classe *DriverBase* per la gestione degli eventi.

4.1.2.4 Metodi di interfaccia

4.1.2.4.1 Metodo 1: Gestione dati in arrivo (GSMChannelManager to Driver)

In tale metodo di firma:

public LowLevelData ManageIncomingDataConnection(IncomingDataConnectionArgument pIDCI, string pDriverVersion)

si istanzia un oggetto della classe LowLevelData e il suo campo "TextData" viene settato con il contenuto informativo di tipo testuale del messaggio SMS, campo del parametro pIDCI (IDCI acronimo di IncomingDataConnectionArgument) che viene ricevuto dal GSMChannelManager. Il GSMChannelManager può specificare anche la versione del Driver Edor a cui fa riferimento per l'interpretazione semantica del contenuto dei dati che ha ricevuto sul canale GSM.

4.1.2.4.2 Metodo 2: Validazione dati GSM (GSMChannelManager to Driver)

Con il metodo di firma:

public bool LowLevelDataValidation(LowLevelData pLowlevelData, string pDriverVersion)

il GSMChannelManager richiede la validazione dei dati GSM che ha conferito al Driver Edor.

4.1.2.4.3 Metodo 3: Elaborazione LowLevelData (GSMChannelManager to Driver)

Nel metodo di firma:

public bool ProcessDeviceRowData(LowLevelData pLowlevelData, string pDriverVersion)

si istanzia un oggetto della classe "DataAcqCompletedEventArgs" ereditato da DriverBase, che rappresenta un evento con cui il Driver comunica all' "eCentral Core" il completamento dell'interpretazione e acquisizione dei dati.

I campi della classe "LowLevelSignal" sono:

Visibilità	Tipo primitive/Classe	Nome	Descrizione	Valore di default
private	bool	isEnabled	Flag di abilitazione	false
private	string	longDescr	Descrizione lunga	null
private	int	modifiedBy	Autore della modifica	-1
private	DateTime	modifiedOn	Data di modifica	DateTime.MinValue
private	int	sdtid	Identidicatore del SignalDataType (es.: 1 -> Integer, 2-> Text, …, 5 -> DateTime)]	-1
private	string	shortDescr	Descrizione breve	null
private	string	longDescr	Descrizione lunga	null
private	int?	period	periodicità	String.Empty
private	int	llsignalID	Identificatore del LLSignal	-1
private	int?	ialid	Identificatore dell'Internationalization abstraction layer	-1
private	int?	syncroAlgorithmGAID	Algoritmo di sincronizzazione	null
private	int?	hlsignalID	Riferimento al Signal di alto livello	-1
private	int	stid	Identificatore sel SignalType (es.: 1-> MISURA, 2-> ALLARME)	-1
private	string	uoM	Unit of measure	null

N.B.: il '?' indica che la variabile ammette valore null

Si attuano in successione le seguenti operazioni:

1. si suddivide in un array di sottostringhe non vuote (di nome “lines”) il campo “TextData” del parametro pLowLevelData in funzione del carattere di controllo “\n”;
2. si eliminano gli spazi in ogni sottostringa (lines[i]), che viene poi convertita in caratteri maiuscoli;
3. se la sottostringa di posizione 0 (lines[0]) inizia con il prefisso EA:
    a. si assegna per riferimento ad un oggetto di nome “communication” della classe Communication il valore di ritorno del metodo “**ParseSMSMessageA(lines)**”;
    b. si assegna al campo Communication di “eventArgs” il riferimento di “communication”;
    c. si setta a true il campo CommunicationSuccessful di “eventArgs”;
4. se la sottostringa di posizione 0 (lines[0]) inizia con il prefisso EB:
    a. si assegna per riferimento ad un oggetto di nome “communication” della classe Communication il valore di ritorno del metodo “**ParseSMSMessageB(lines)**”;
    b. si assegna al campo Communication di “eventArgs” il riferimento di “communication”;
    c. si setta a true il campo CommunicationSuccessful di “eventArgs”;
5. se la sottostringa di posizione 0 (lines[0]) inizia con il prefisso EC:
    a. si assegna per riferimento ad un oggetto di nome “communication” della classe Communication il valore di ritorno del metodo “**ParseSMSMessageC(lines)**”;
    b. si assegna al campo Communication di “eventArgs” il riferimento di “communication”;
    c. si setta a true il campo CommunicationSuccessful di “eventArgs”;
6. se la sottostringa di posizione 0 (lines[0]) inizia con il prefisso ED e non inizia con il prefisso EDOR:
    a. si assegna per riferimento ad un oggetto di nome “communication” della classe Communication il valore di ritorno del metodo “**ParseSMSMessageD(lines)**”;
    b. si assegna al campo Communication di “eventArgs” il riferimento di “communication”;
    c. si setta a true il campo CommunicationSuccessful di “eventArgs”;
7. se la sottostringa di posizione 0 (lines[0]) inizia con il prefisso EE:
    a. si assegna per riferimento ad un oggetto di nome “communication” della classe Communication il valore di ritorno del metodo “**ParseSMSMessageE(lines)**”;
    b. si assegna al campo Communication di “eventArgs” il riferimento di “communication”;
    c. si setta a true il campo CommunicationSuccessful di “eventArgs”;
8. se la sottostringa di posizione 0 (lines[0]) inizia con il prefisso EF:
    a. si assegna per riferimento ad un oggetto di nome “communication” della classe Communication il valore di ritorno del metodo “**ParseSMSMessageF(lines)**”;
    b. si assegna al campo Communication di “eventArgs” il riferimento di “communication”;
    c. si setta a true il campo CommunicationSuccessful di “eventArgs”;
9. se la sottostringa di posizione 0 (lines[0]) inizia con il prefisso EK:
    a. si assegna per riferimento ad un oggetto di nome “communication” della classe Communication il valore di ritorno del metodo “**ParseSMSMessageACK(lines)**”;
    b. si assegna al campo Communication di “eventArgs” il riferimento di “communication”;
    c. si setta a true il campo CommunicationSuccessful di “eventArgs”;
10. se la sottostringa di posizione 0 (lines[0]) inizia con il prefisso EDOR oppure con il carattere #:

a. si assegna per riferimento ad un oggetto di nome “communication” della classe Communication il valore di ritorno del metodo “**ParseSMSMessageA(lines)**”;
b. si assegna al campo Communication di “eventArgs” il riferimento di “communication”;
c. si setta a true il campo CommunicationSuccessful di “eventArgs”;

Terminata l’acquisizione dei dati si genera l’evento:

“FireDataAcqCompleted(eventArgs)”.

I campi della classe DataAcqCompletedEventArgs:

Visibilità	Tipo primitive/Classe	Nome	Descrizione	Valore di default
private	Communication	Communication	Pacchetto	null
private	bool	Communication Successful	Flag di esito della comunicazione	false

I campi della classe “Communication”:

Visibilità	Tipo primitive/Classe	Nome	Descrizione	Valore di default
private	int	commID	Identificatore della comunicazione	-1
private	int	did	Identificatore del device	-1
private	int	commType	Tipo di comunicazione (es. GSM)	-1
private	int	commDirection	Direzione della comunicazione	-1
private	int	modifiedBy	Autore della modifica	-1
private	DateTime	modifiedOn	Data di modifica	DateTime.MinValue
private	List<CommunicationData>	data	Informazioni	New List<CommunicationData>( )

4.1.2.5 Metodi IDriver2Core ed eventi

4.1.2.5.1 RequireDataToDevice(int pCommID, string pMessage)

Con il metodo “**RequireDataToDevice(int pCommID, string pMessage)**” eCentralCore comunica con il device per eseguire una telelettura, che è differente dalla configurazione.

4.1.2.5.2 ProgramDevice(int pCommID, List<CommunicationData> pCdList)

Il metodo “**ProgramDevice(int pCommID, List<CommunicationData> pCdList)**” viene invocato per modificare la programmazione o i parametri di funzionamento, oppure per inviare dei comandi al device.

Con tale metodo si inviano tipicamente uno o più SMS al termine dell'attivazione.

Esempio di SMS:

EN00112233;A=XXXXXXXX;D=YYYYYYYYY;E=ZZZZZZZZ;

EN è una parte fissa, 00112233 è l'ID attuale del device Edor.

Tabella 4.2: Comandi

Comando	Descrizione	Esempio
A=xxxxxxxx	Imposta l'ID dell'Edor; l'ID può essere alfanumerico e comprendere spazi, e può essere formato da un massimo di 8 caratteri. Nonostante il device possa gestire ogni tipo di ID, è buona norma utilizzare ID numerici composti esattamente da 8 cifre, per ridurre al minimo l'errore umano.	A=00112233;
B=xxxxxxxxxx	Batteria di bit (settaggi), utilizzare sempre 3 cifre. Il numero è in decimale.	B=200;
C=xxxxxxxxxx	Appuntamento download da remoto, è necessario fornire data e ora di chiamata e numero di telefono da chiamare.	C=20070329121000,+393387571270;
D=xxxxxxxxxx	Imposta il numero telefonico del reperibile; il numero può essere formato al massimo da 15 caratteri; è accettato anche il '+' per i prefissi internazionali. OFF per disattivare il numero.	D=+397534270;
E=xxxxxxxxxx	Imposta il numero telefonico del reperibile; il numero può essere formato al massimo da 15 caratteri; è accettato anche il '+' per i prefissi internazionali. OFF per disattivare il numero.	E=OFF;
F=xxxx	Imposta l'orario di invio dell'sms giornaliero; occorre sempre specificare 4 cifre, 2 per l'ora, 2 per i minuti: ad esempio per programmare le 07:05 del mattino, specificare '0705'. Poiché l'ultima misura deve coincidere con l'invio dell'sms, l'impostazione dell'orario di invio dell'sms implica il cambiamento degli orari di misura.	F=0705;
G=xxxxx	Imposta il periodo fra 2	G=00360;

	misure consecutive, in minuti. Utilizzare al massimo 5 cifre.	
H=xx	Imposta la soglia di allarme di basso TO; sono accettate al massimo 2 cifre.	H=35;
I=xx	Imposta la soglia di allarme di alto TO; sono accettate al massimo 2 cifre.	I=45;
J=xxx	Imposta il guadagno Gs del sensore, nel TBM è moltiplicato per 10 (quindi per inserire 5.7 è necessario inserire 57). Massimo 3 cifre.	J=057;
K=xxx	Imposta il guadagno di etichetta Go del sensore, nel TBM è moltiplicato per 10 (quindi per inserire 5.7 è necessario inserire 57). Massimo 3 cifre.	K=057;
L=xxx	Imposta il periodo di taratura in giorni. Massimo 3 cifre.	L=021;
M=xxx	Imposta il tasso della bombola campione. Massimo 3 cifre. Sempre moltiplicato per 10 (es. 30.0 mg/sm^3 -> 300)	M=300;
N=xxx	Imposta i telecomandi, per eseguire misure su gas di rete o su bombola	N=10;
T=sxxxxxxxx	Modifica data e ora, il valore che viene inviato è il numero di secondi da indietreggiare o avanzare, il segno – indica di indietreggiare. Massimo 8 cifre.	T=-3600 indietreggia di 1 ora.
O=xxxxxxx	Imposta il numero telefonico del 2° reperibile; il numero può essere formato al massimo da 15 caratteri; è accettato anche il '+' per i prefissi internazionali. OFF per disattivare il numero.	O=3352112121;

Si crea un array di stringhe di nome ***command*** di dimensione pari a COMMAND_TYPE.NUM. L'array *command* contiene tutti i comandi di programmazione da inviare al device. Per ogni oggetto CommunicationData del parametro pCdList (lista di comandi di programmazione), si confronta il suo campo LLSignalID con ogni possibile comando Edor per individuarne il corrispondente. Nel seguito è mostrato una porzione di codice per facilitare la comprensione:

foreach (CommunicationData cd in pCdList)
{

```
switch (cd.LLSignalID)
{
    // COMMAND A
    case EC.Signal.Edor.NSerieEdor:
            command[(int)COMMAND_TYPE.A] = Constants.ProgramCommand.A + cd.ValueT +
            ";";
    break;

    // COMMAND B
    …
```

Il Driver Edor conosce gli LLSignal, essendo scolpiti nelle costanti della solution (***EC.Signal.Edor.NSerieEdor***), per cui non deve recuperarli dalla tabella LowLevelSignal nella base di dati.

Una volta individuato il comando, nell'array "***command***" alla posizione specificata dalla costante "COMMAND_TYPE.<nome_comando>" si inserisce la stringa di programmazione costituita da:

Constants.ProgramCommand.A + cd.ValueT + ";"

che si traduce in:

A=testo;

Alcuni comandi (come B e N) richiedono l'uso di una maschera esadecimale; ad esempio, per il LLSignal *ENOS_SET_SMS_DISABLE* si inserisce in una variabile temporanea la maschera corrispondente, come nel seguito mostrato:

tempInt4B |= Constants.SettingsBitmask.ENOS_SET_SMS_DISABLE;

che si traduce in:

tempInt4B=0x01

Nell'array "***command***" alla posizione specificata dalla costante "COMMAND_TYPE.<nome_comando>" la stringa di programmazione è costituita da:

Constants.ProgramCommand.B + Convert.ToString(tempInt4B, 10).PadRight(3, '0') + ";"

che si traduce in:

B=<valore della maschera esadecimale convertita nella base decimale e tradotta in stringa, con allineati a sinistra i caratteri, aggiungendo a destra il carattere '0' fino a ottenere la lunghezza totale 3>;

Nel caso del comando C, se il LLSignalID è "AppuntamentoDownloadDaRemoto", in "tempStringADR" si inserisce *cd.ValueT*.

Nell'array "***command***" alla posizione specificata dalla costante "COMMAND_TYPE.C" si inserisce la stringa di programmazione costituita da:

Constants.ProgramCommand.C + tempStringADR + ";"

Nel caso in cui, viceversa, "tempStringNTDR" sia diversa dalla stringa vuota (è stato ricevuto prima il LLSignal "NtelDownloadDaRemoto"), si deve concatenare, con la stringa prima inserita, la stringa "tempStringNTDR" che termina con il ';'. "tempStringNTDR" è il *cd.ValueT* del LLSignalID "NtelDownloadDaRemoto".

Nel caso di comando C, se il LLSignalID è "NtelDownloadDaRemoto", in "tempStringNTDR" si inserisce *cd.ValueT* e se "tempStringADR" non è null (questa volta è stato ricevuto prima il LLSignal

“AppuntamentoDownloadDaRemoto”) si concatena la stringa “tempStringNTDR” terminata con il ‘;’ alla stringa nell’array “***command***” alla posizione “COMMAND_TYPE.C”.

Una volta settato il contenuto di *command* (insieme di tutti i comandi di programmazione) si deve, poi, procedere alla sua trasmissione. Prima si esegue un controllo sulla struttura di ogni singolo comando, invocando il metodo di signature:

private bool CheckMessageProof(IEnumerable<string> pCommandString)

Solo se il metodo restituisce true si può proseguire. Tramite una semplice query si recupera, dato il CommID, il DID (DeviceID) che viene assegnato alla variabil *pDID*. Si cita brevemente il codice per maggiore chiarezza:

```
if (CheckMessageProof(command))
{
    List<SMS> smsList = SMSCreator(command, pDID, pLoggedPID);

    foreach (SMS sms in smsList)
    {
        List<GSMEndpoint> gsmEndpoints = DeviceManager.Instance.GetGSMEndpointList(pDID, false, false, pLoggedPID, null);
        // TODO per ora si considera il primo degli endpoint della lista come quello contenente il numero di SIM destinatario
        CommunicationLogManager.Instance.InsertLogSMS(new LogSMS(pCommID, sms.Message, gsmEndpoints[0].SIMNumber, pLoggedPID), true, pLoggedPID, null);
          }

    retVal = true;
}
```

È sufficiente che un solo comando non sia ben formato e, quindi, non superi il controllo, perché non si crei l’SMS o gli SMS (se tutti i comandi non stanno all’interno di un singolo messaggio) di programmazione. I controlli sono i seguenti:

1. se la stringa *command* non ha un riferimento si restituisce false;
2. per ogni sottostringa in *command* (ogni singolo comando):
    a. tipo ‘A’ se più lungo di 8 caratteri si restituisce false;
    b. tipo ‘B’ se più lungo di 3 caratteri si restituisce false;
    c. tipo ‘C’ se lungo meno di 9, più di 15 e diverso da 14 caratteri si restituisce false;
    d. tipo ‘D’ se lungo meno di 9 e [non contiene “OFF;” oppure è più lungo di 15 caratteri] si restituisce false;
    e. tipo ‘E’ se lungo meno di 9 e [non contiene “OFF;” oppure è più lungo di 15 caratteri] si restituisce false;
    f. tipo ‘F’ se lungo diversamente da 4 si restituisce false;
    g. tipo ‘G’ se lungo più di 5 caratteri si restituisce false;
    h. tipo ‘H’ se lungo più di 2 caratteri si restituisce false;
    i. tipo ‘I’ se lungo più di 2 caratteri si restituisce false;
    j. tipo ‘L’ se lungo più di 3 caratteri si restituisce false;
    k. tipo ‘M’ se lungo più di 3 caratteri si restituisce false;

l. tipo ‘N’ se lungo più di 2 caratteri si restituisce false;
m. tipo ‘T’ se lungo più di 8 caratteri si restituisce false;
n. tipo ‘O’ se lungo meno di 9 e [non contiene “OFF;” oppure è più lungo di 15 caratteri] si restituisce false;

Esempio di comando di tipo D corretti:

D=OFF;

D=+393331234567;

D=333123456; (dopo il 333 si possono avere 6 o 7 caratteri, ma non si può comunque scendere al di sotto dei 6, per cui se < 9 viene segnalato l’errore).

La trasmissione richiede una corretta formattazione conforme alla struttura di un SMS. Un comando potrebbe richiedere più SMS. Si invoca il metodo di firma:

public List<SMS> SMSCreator(string[] pCommand, int pDID, int pLoggedPID)

in cui il parametro pCommand è *command*. All’interno del metodo si crea una lista di SMS di nome *retVal* e uno StringBuffer *sb*. Si demanda all’istanza di GeneralPropertyManager l’onere di recuperare la lista di QOPropertyValue, denominata **qoPValList**, attraverso il metodo:

“GetQOPropertyValueList(eCentralConstants.SystemProperties.EdorSN.GPID,

eCentralConstants.QualifiableObjects.Device, pDID, null, pLoggedPID, null)”

con

eCentralConstants.SystemProperties.EdorSN.GPID=16;

eCentralConstants.QualifiableObjects.Device=3;

di firma:

*public List<QOPropertyValue> GetQOPropertyValueList(int pGPID, int pQOID, int pObjectID, DateTime? **pTime**, int pLoggedPID, SqlTransaction pDbTran)*

Nel metodo GPID=16 equivale a “EdorSerialNumber”, mentre QOID=3 equivale a “Device”.

Se il **pTime** è non nullo (ad esempio, 11 agosto 2009) si esegue la seguente query:

*SELECT * FROM GetQOPropertyValue(@pGPID, @pQOID, @pObjectID, @pTime)*

altrimenti, per effettuare l’interrogazione all’istante di tempo del RTC del DBMS, si esegue:

*SELECT * FROM GetQOPropertyValue(@pGPID, @pQOID, @pObjectID, getdate())*

```
ALTER FUNCTION [dbo].[GetQOPropertyValue]
(
   @pGPID int,
   @pQOID int,
   @ObjectID int,
   @pCUID int,
   @pTime datetime
)
RETURNS
@retTable TABLE
(

   InstanceN int,
   EffectiveSince datetime,
   IsDeleted bit,
```

```
    ModifiedBy int,
    ValueB bit,
    ValueI int,
    ValueF float,
    ValueD datetime,
    ValueT nvarchar(max),
    ValueL image,
    ValueA int
)
AS
BEGIN

    -- PRINCIPAL
    if @pQOID = 1
    begin
        insert into @retTable (InstanceN, EffectiveSince, IsDeleted, ModifiedBy, ValueB, ValueI,ValueF,
ValueD,ValueT,ValueL, ValueA)
        select  pv.InstanceN, pv.EffectiveSince, pv.IsDeleted, pv.ModifiedBy, pv.ValueB, pv.ValueI,
pv.ValueF, pv.ValueD, pv.ValueT, pv.ValueL, pv.ValueA
        from dbo.GetCur_PPropertyValue(@pTime, @pCUID) pv
        where pv.PID = @ObjectID and pv.GPID = @pGPID order by InstanceN
    end

    -- GROUP
    if @pQOID = 2
    begin
        insert into @retTable (InstanceN, EffectiveSince, IsDeleted, ModifiedBy, ValueB, ValueI,ValueF,
ValueD,ValueT,ValueL, ValueA)
        select  pv.InstanceN, pv.EffectiveSince, pv.IsDeleted, pv.ModifiedBy,
              pv.ValueB, pv.ValueI, pv.ValueF, pv.ValueD, pv.ValueT, pv.ValueL, pv.ValueA
        from dbo.GetCur_GPropertyValue(@pTime, @pCUID) pv
        where pv.GID = @ObjectID and pv.GPID = @pGPID order by InstanceN
    end

    -- DEVICE
     if @pQOID = 3
    begin
        insert into @retTable (InstanceN, EffectiveSince, IsDeleted, ModifiedBy, ValueB, ValueI,ValueF,
ValueD,ValueT,ValueL, ValueA)
        select  pv.InstanceN, pv.EffectiveSince, pv.IsDeleted, pv.ModifiedBy,
              pv.ValueB, pv.ValueI, pv.ValueF, pv.ValueD, pv.ValueT, pv.ValueL, pv.ValueA
        from dbo.GetCur_DPropertyValue(@pTime, @pCUID) pv
        where pv.DID = @ObjectID and pv.GPID = @pGPID order by InstanceN
    end

    -- DEVICE MODEL
     if @pQOID = 4
    begin
        insert into @retTable (InstanceN, EffectiveSince, IsDeleted, ModifiedBy, ValueB, ValueI,ValueF,
ValueD,ValueT,ValueL, ValueA)
        select  pv.InstanceN, pv.EffectiveSince, pv.IsDeleted, pv.ModifiedBy,
              pv.ValueB, pv.ValueI, pv.ValueF, pv.ValueD, pv.ValueT, pv.ValueL, pv.ValueA
        from dbo.GetCur_DMPropertyValue(@pTime, @pCUID) pv
        where pv.DMID = @ObjectID and pv.GPID = @pGPID order by InstanceN
    end

    -- DEVICE SCHEMA
     if @pQOID = 5
```

```
   begin
      insert into @retTable (InstanceN, EffectiveSince, IsDeleted, ModifiedBy, ValueB, ValueI,ValueF,
ValueD,ValueT,ValueL, ValueA)
      select  pv.InstanceN, pv.EffectiveSince, pv.IsDeleted, pv.ModifiedBy,
            pv.ValueB, pv.ValueI, pv.ValueF, pv.ValueD, pv.ValueT, pv.ValueL, pv.ValueA
      from dbo.GetCur_DSPropertyValue(@pTime, @pCUID) pv
      where pv.DSID = @ObjectID and pv.GPID = @pGPID order by InstanceN
   end

   -- DEVICE CLASS
    if @pQOID = 6
   begin
      insert into @retTable (InstanceN, EffectiveSince, IsDeleted, ModifiedBy, ValueB, ValueI,ValueF,
ValueD,ValueT,ValueL, ValueA)
      select  pv.InstanceN, pv.EffectiveSince, pv.IsDeleted, pv.ModifiedBy,
            pv.ValueB, pv.ValueI, pv.ValueF, pv.ValueD, pv.ValueT, pv.ValueL, pv.ValueA
      from dbo.GetCur_DCPropertyValue(@pTime, @pCUID) pv
      where pv.DCID = @ObjectID and pv.GPID = @pGPID order by InstanceN
   end

   -- ACTIVITY
    if @pQOID = 8
   begin
      insert into @retTable (InstanceN, EffectiveSince, IsDeleted, ModifiedBy, ValueB, ValueI,ValueF,
ValueD,ValueT,ValueL, ValueA)
      select  pv.InstanceN, pv.EffectiveSince, pv.IsDeleted, pv.ModifiedBy,
            pv.ValueB, pv.ValueI, pv.ValueF, pv.ValueD, pv.ValueT, pv.ValueL, pv.ValueA
      from dbo.GetCur_ACTPropertyValue(@pTime, @pCUID) pv
      where pv.ACTID = @ObjectID and pv.GPID = @pGPID order by InstanceN
   end

RETURN
END


ALTER FUNCTION [dbo].[GetCur_DPropertyValue](@pDate datetime, @pCUID int)
RETURNS TABLE
AS RETURN
(
    Select O.*, G.PTID, G.ShortDescr, G.LongDescr
    From DPropertyValue O
    join GetI18NGeneralProperty(@pCUID) G on O.GPID = G.GPID
    Where O.DPVID = (Select Top 1 I.DPVID
                     From DPropertyValue I
                     Where I.DID = O.DID
                     And I.GPID = O.GPID
                     And I.InstanceN = O.InstanceN
                     And I.EffectiveSince <= @pDate
                     Order by I.EffectiveSince DESC, I.IsDeleted ASC
                    )
)
```

Una volta recuperato, il serialNumber viene concatenato alla stringa ‘EN’, che termina con il carattere di controllo ‘;’ e viene aggiunta allo StringBuilder *sb*. Per ogni sottostringa di *pCommand*, equivalente ad un comando (nella forma <tipo>=<valore>;) si verifica se la sua lunghezza più quella di *sb* supera i 160 caratteri (dimensione di un SMS):

- in caso affermativo: si setta il contenuto di SMS.Message al contenuto di *sb* e il valore di SMS.TimeStamp al valore di DateTime.now. Si aggiunge a *retVal* l'SMS appena creato, si crea un nuovo oggetto SMS ed, infine, si ripulisce *sb* reinizializzandolo a "EN<valore del serialNumber>;";
- in caso negativo: si aggiunge la sottostringa di *pCommand* (cioè il comando) a *sb*.

Terminato il ciclo *foreach* per ogni comando di *pComand*, si settano:

- il contenuto di SMS.Message al contenuto di *sb*;
- SMS.TimeStamp al valore di DateTime.now.

Si aggiunge, infine, a *retVal* l'SMS appena creato. Il metodo è concluso e restituisce al metodo chiamante la lista di SMS necessaria alla trasmissione dei comandi per configurare il device. Per ogni SMS della lista si recupera la lista List<GSMEndpoint> gsmEndpoints attraverso l'invocazione di *GetGSMEndpointList(pDID, false, false, pLoggedPID, null)* su DeviceManager.

Si inserisce, poi, l'oggetto SMS da inviare come record nella tabella LogSMS attraverso il CommunicationLogManager con il metodo:

InsertLogSMS(new LogSMS(pCommID, sms.Message, gsmEndpoints[0].SIMNumber, pLoggedPID), true, pLoggedPID, null);

considerando il primo degli endpoint della lista come la SIM del device destinatario. Poiché un device può disporre di più SIM, di cui una sola attiva, si sceglie la prima per default.

Il thread "postino" del GSMChannelManager andrà in polling sulla tabella LogSMS andando a recuperare gli SMS accodati, provvedendo a spedirli.

In figura 4.2 è mostrato il workflow per favorire la comprensione:

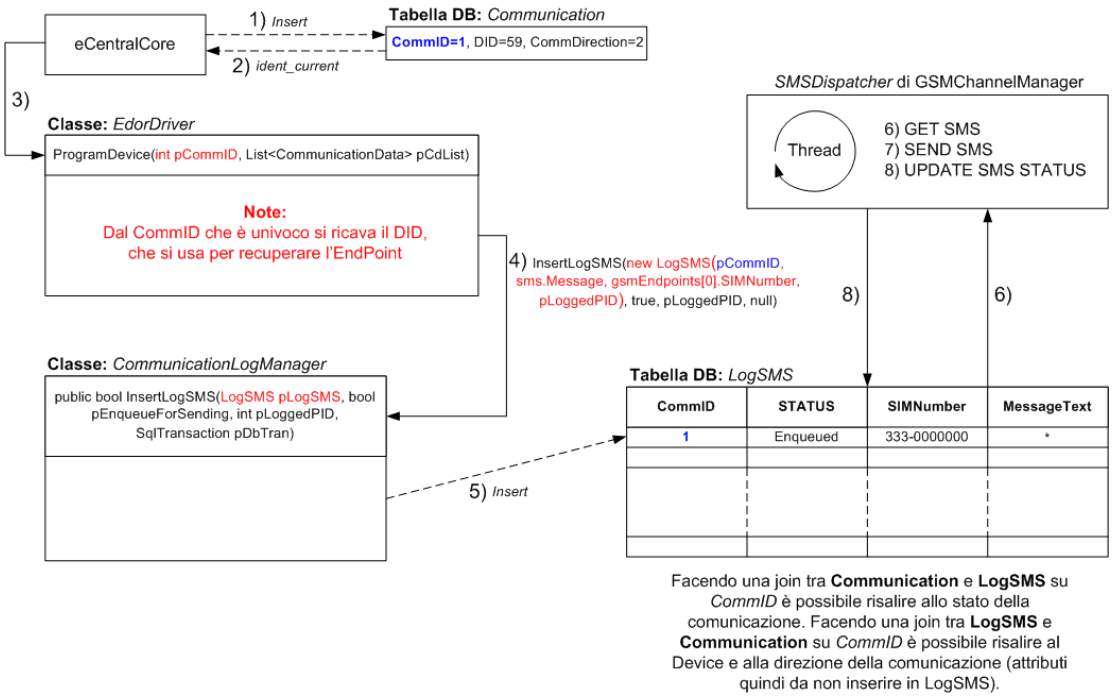


Figura 4.2: workflow di spedizione

4.1.2.6 Metodi di parsing dei messaggi SMS (protocollo SMS Edor)

4.1.2.6.1 Metodo A: parsing del contenuto di un SMS di tipo A (misure giornaliere)

Descrizione: Edor effettua 4 misure ogni giorno, a distanza di n ore (con n programmabile, tipicamente 6 or) l'una dall'altra; al termine della quarta misura, il device invia un SMS con le misure raccolte nella giornata. L'orario di invio è programmabile e di default è 7:00.

Implementazione

Nel metodo di firma:

private **Communication** ParseSMSMessageA(string[] plines)

viene istanziato un oggetto della classe Communication di nome *retVal* come valore di ritorno del metodo.

Il parsing è effettuato sulla base della conoscenza a priori della lunghezza di ciascun campo componente la sottostringa "plines" di analisi.

Si compiono le seguenti elaborazioni in funzione della sottostringa in analisi:

1. plines[1]:
    a. il giorno è dato dalla conversione della sua sottostringa (0,1) in un valore Integer con segno a 32 bit;
    b. il mese è dato dalla conversione della sua sottostringa (2,3) in un valore Integer con segno a 32 bit;
    c. l'anno è dato sommando a 2000 il valore integer con segno a 32 bit corrispondente alla sua sottostringa (4,5);
    d. si crea l'oggetto di classe "DateTime" di nome SMSOnlyDate specificando anno, mese, giorno calcolati;
2. plines[2] (la stessa procedura si ripete per "plines[3]", "plines[4]", "plines[5]" perché come detto in precedenza sono 4 le misurazioni effettuate da Edor) contiene:
    a. istante di tempo di misura del Signal (DateTime)
        i. l'ora è data dalla conversione della sua sottostringa (0,1) in un valore Integer con segno a 32 bit;
        ii. il minuto è dato dalla conversione della sua sottostringa (2,3) in un valore Integer con segno a 32 bit;
        iii. l'istante di tempo di rilevamento del Signal è creato come oggetto della classe "DateTime" passando come parametri: anno, mese, giorno (calcolati al punto 1), ora e minuto (calcolati ora) e 0 secondi;

Elenco di Signal

b. TO puro [mg/m^3] (Integer)

i. si crea un oggetto della classe “CommunicationData” di nome “cd” di campi:

Visibilità	Tipo primitive/Classe	Nome	Descrizione	Valore di default
private	Int	commID	Identificatore della comunicazione	-1
private	Int	LLSignalID	Identificatore del segnale	-1
private	DateTime	acqTime	Timestamp di competenza del dispositivo	Null
private	DateTime	dataTime	Timestamp con cui eCentral marchia l’istante di tempo di arrivo di un Communication	Null
private	bool	valueB	Valore booleano	Null
private	DateTime	valueD	Valore DateTime	Null
private	double	valueF	Valore float	null
private	int	valueI	Valore Integer	null
private	string	valueT	Valore Testo	null

N.B.: l’acqTime è la data alla quale è più corretto che la misura del Signal si riferisca, dunque se:

1. è la data di effettiva rilevazione del Signal in esame il ±Δ è nullo (nessuna compensazione software). Ciò implica che il device mittente è un device dotato di particolare “intelligenza”, poiché è in grado di mantenere uno storico dei rilievi fatti su apposita unità di memorizzazione;
2. è la data di spedizione del SMS, il tecnico deve comunicare ±Δ per calcolare la data di effettiva rilevazione. Ciò evidenza che il device mittente è meno capace.

eCentral non deve preoccuparsi del device mittente, ma per garantire una buona robustezza deve essere flessibile, basando l’acquisizione dei dati su di una struttura dati efficace in grado di gestire una granularità delle misure più o meno fine. Il differente comportamento deve essere implementato a livello Driver.

---

ii. si setta il campo “acqTime” al valore di 2.a.iii;

iii. si setta il campo “LLSignalID” sfruttando “eCentralConstants.Signal.Edor.TOPuro” pari a 2000;

iv. si setta il campo “valueI” convertendo la sua sottostringa (4,5) in un valore Integer con segno a 32 bit;

v. si aggiunge al campo “data” di *retVal* l’oggetto “cd” settato opportunamente ai punti ii,iii,iv.

c. TO compensato [mg/m^3] (Integer)

i. si crea un oggetto “CommunicationData” di nome “cd”;

ii. si setta il campo “acqTime” al valore di 2.a.iii;

iii. si setta il campo “LLSignalID” sfruttando “eCentralConstants.Signal.Edor.TOCompensato” pari a 2001;

iv. si setta il campo “valueI” convertendo la sua sottostringa (6,7) in un valore Integer con segno a 32 bit;

v. si aggiunge al campo “data” di *retVal* l’oggetto “cd” settato opportunamente ai punti ii,iii,iv.

d. Zero puro [mg/m^3] (Integer)

i. si crea un oggetto “CommunicationData” di nome “cd”;

ii. si setta il campo “acqTime” al valore di 2.a.iii;

iii. si setta il campo “LLSignalID” sfruttando “eCentralConstants.Signal.Edor.ZeroPuro” pari a 2002;

iv. si setta il campo “valueI” convertendo la sua sottostringa (8,9) in un valore Integer con segno a 32 bit;

v. si aggiunge al campo “data” di *retVal* l’oggetto “cd” settato opportunamente ai punti ii,iii,iv.

e. Zero compensato [mg/m^3] (Integer)

i. si crea un oggetto CommunicationData di nome “cd”;

ii. si setta il campo “acqTime” al valore di 2.a.iii;

iii. si setta il campo “LLSignalID” sfruttando “eCentralConstants.Signal.Edor.ZeroCompensato” pari a 2003;

iv. si setta il campo “valueI” convertendo la sua sottostringa (10,11) in un valore Integer con segno a 32 bit;

v. si aggiunge al campo “data” di *retVal* l’oggetto “cd” settato opportunamente ai punti ii,iii,iv.

f. Diagnostica

i. la diagnostica è data dalla conversione della sua sottostringa (12,15) in un valore integer con segno a 32 bit;

ii. invoco il metodo “ParseDiagnosticFrame(diag, retVal, measureDateTime[lineNumber - 2])” per effettuare il parsing opportuno;

g. Temperatura [°C] (Integer)

i. si crea un oggetto “CommunicationData” di nome “cd”;

ii. si setta il campo “acqTime” al valore di 2.a.iii;

iii. si setta il campo “LLSignalID” sfruttando “eCentralConstants.Signal.Edor.Temperatura” pari a 2004;

iv. si setta il campo “valueI” convertendo la sua sottostringa (16,18) in un valore Integer con segno a 32 bit;

v. si aggiunge al campo “data” di *retVal* l’oggetto “cd” settato opportunamente ai punti ii,iii,iv.

h. Numero di secondi di apertura dell’elettrovalvola (Integer) (da moltiplicare per 2)

i. si crea un oggetto “CommunicationData” di nome “cd”;

ii. si setta il campo “acqTime” al valore di 2.a.iii;

iii. si setta il campo “LLSignalID” sfruttando “eCentralConstants.Signal.Edor.SecElettrovalvAperta” pari a 2005;
iv. si setta il campo “valueI” convertendo la sua sottostringa (19,20) in un valore Integer con segno a 32 bit;
v. si aggiunge al campo “data” di *retVal* l’oggetto “cd” settato opportunamente ai punti ii,iii,iv.

i. Minuti di zero (Integer)

i. si crea un oggetto “CommunicationData” di nome “cd”;
ii. si setta il campo “acqTime” al valore di 2.a.iii;
iii. si setta il campo “LLSignalID” sfruttando “eCentralConstants.Signal.Edor.MinZero” pari a 2006;
iv. si setta il campo “valueI” convertendo la sua sottostringa di un carattere alla posizione 21 in un valore Integer con segno a 32 bit;
v. si aggiunge al campo “data” di *retVal* l’oggetto “cd” settato opportunamente ai punti ii,iii,iv.

3. plines[6]

Elenco di Signal

a. Tipo di sensore (text)

i. si crea un oggetto “CommunicationData” di nome “cd”;
ii. si setta il campo “acqTime” al valore di 1.d;
iii. si setta il campo “LLSignalID” sfruttando “eCentralConstants.Signal.Edor.TipoSensore” pari a 2007;
iv. si setta il campo “valueT” a:
    1. TBM (Constants.SensorType.TBM): se la sua sottostringa di un carattere alla posizione 0 è B;
    2. THT (Constants.SensorType.THT): se la sua sottostringa di un carattere alla posizione 0 è H;
v. si aggiunge al campo “data” di *retVal* l’oggetto “cd” settato opportunamente ai punti ii,iii,iv.

b. GS cell (integer): guadagno amplificatore

i. si crea un oggetto “CommunicationData” di nome “cd”;
ii. si setta il campo “acqTime” al valore di 1.d;
iii. si setta il campo “LLSignalID” sfruttando “eCentralConstants.Signal.Edor.GsCella” pari a 2008;
iv. si setta il campo “valueI” convertendo la sua sottostringa (1,3) in un valore Integer con segno a 32 bit;
v. si aggiunge al campo “data” di *retVal* l’oggetto “cd” settato opportunamente ai punti ii,iii,iv.

c. Tensione di batteria [Volt] (float)

i. si crea un oggetto “CommunicationData” di nome “cd”;
ii. si setta il campo “acqTime” al valore di 1.d;
iii. si setta il campo “LLSignalID” sfruttando “eCentralConstants.Signal.Edor.Batteria” pari a 2009;
iv. si setta il campo “valueI” convertendo la sua sottostringa (4,7) in un valore double;
v. si aggiunge al campo “data” di *retVal* l’oggetto “cd” settato opportunamente ai punti ii,iii,iv.

d. Campo GSM [es. 19/31] (Integer)

i. si crea un oggetto “CommunicationData” di nome “cd”;
ii. si setta il campo “acqTime” al valore di 1.d;
iii. si setta il campo “LLSignalID” sfruttando “eCentralConstants.Signal.Edor.CampoGSM” pari a 2010;
iv. si setta il campo “valueI” convertendo la sua sottostringa (8,9) in un valore Integer con segno a 32 bit;
v. si aggiunge al campo “data” di *retVal* l’oggetto “cd” settato opportunamente ai punti ii,iii,iv.

e. TO evacuazione misura 1 (Integer)

i. si crea un oggetto “CommunicationData” di nome “cd”;
ii. si setta il campo “acqTime” al valore di 1.d;
iii. si setta il campo “LLSignalID” sfruttando “eCentralConstants.Signal.Edor.TOEvacMisura1” pari a 2011;
iv. si setta il campo “valueI” convertendo la sua sottostringa (10,11) in un valore Integer con segno a 32 bit;
v. si aggiunge al campo “data” di *retVal* l’oggetto “cd” settato opportunamente ai punti ii,iii,iv.

f. TO evacuazione misura 2 (Integer)

i. si crea un oggetto “CommunicationData” di nome “cd”;
ii. si setta il campo “acqTime” al valore di 1.d;
iii. si setta il campo “LLSignalID” sfruttando “eCentralConstants.Signal.Edor. TOEvacMisura2” pari a 2012;
iv. si setta il campo “valueI” convertendo la sua sottostringa (12,13) in un valore Integer con segno a 32 bit;
v. si aggiunge al campo “data” di *retVal* l’oggetto “cd” settato opportunamente ai punti ii,iii,iv.

g. TO evacuazione misura 3 (Integer)

i. si crea un oggetto “CommunicationData” di nome “cd”;
ii. si setta il campo “acqTime” al valore di 1.d;

iii. si setta il campo “LLSignalID” sfruttando “eCentralConstants.Signal.Edor. TOEvacMisura3” pari a 2013;
iv. si setta il campo “valueI” convertendo la sua sottostringa (14,15) in un valore Integer con segno a 32 bit;
v. si aggiunge al campo “data” di *retVal* l’oggetto “cd” settato opportunamente ai punti ii,iii,iv.

h. TO evacuazione misura 4 (Integer)

i. si crea un oggetto “CommunicationData” di nome “cd”;
ii. si setta il campo “acqTime” al valore di 1.d;
iii. si setta il campo “LLSignalID” sfruttando “eCentralConstants.Signal.Edor. TOEvacMisura4” pari a 2014;
iv. si setta il campo “valueI” convertendo la sua sottostringa (16,17) in un valore Integer con segno a 32 bit.
v. si aggiunge al campo “data” di *retVal* l’oggetto “cd” settato opportunamente ai punti ii,iii,iv.

4.1.2.6.2 Metodo B: parsing del contenuto di un SMS di tipo B (misura singola)

Descrizione: Edor può essere programmato per inviare un SMS dopo ogni singola misura (ogni <n> ore), anche se questa operazione è normalmente disabilitata, per minimizzare il consumo della batteria. Il messaggio può anche essere inviato in risposta al relativo comando da parte di eCentral o dopo la pressione di un tasto sulla centralina.

Implementazione

Nel metodo di firma:

private **Communication** ParseSMSMessageB(string[] plines) operando elaborazioni analoghe al metodo A basate sempre sulla conoscenza a priori della lunghezza dei campi costituenti una data “plines”:

1. plines[1] di 6 caratteri contiene solo la data di invio dell’SMS (anno di 2 caratteri, mese di 2 caratteri, giorno di 2 caratteri);
2. plines[2] contiene:
   a. l’ora (2 caratteri) e il minuto (2 caratteri) in cui è stata effettuata la misura del Signal.
      L’istante di tempo del rilevamento del Signal è composta dall’anno, mese, giorno del punto 1 e dall’ora e del minuto del punto in esame. I Signal di seguito avranno come timestamp tale l’istante di tempo di rilevamento.

Elenco di Signal

   b. TO puro (2 caratteri);
   c. TO compensato (2 caratteri);
   d. zero puro (2 caratteri);
   e. zero compensato (2 caratteri);

f. diagnostica (4 caratteri): richiede il parsing "ParseDiagnosticFrame(diag, retVal, measureDateTime)";
g. temperatura (3 caratteri: segno = 1 caratteri e valore = 2 caratteri);
h. numero di secondi di apertura dell'elettrovalvola (2 caratteri);
i. minuti di zero (1 carattere);

3. plines[3] contiene:

sottinteso al tempo misurato al punto 2.a si ha:

Elenco di Signal

a. tipo di sensore (1 carattere):
    1. TBM;
    2. THT.
b. Gs Cella (guadagno amplificatore, 3 caratteri);
c. tensione di batteria (4 caratteri);
d. campo GSM (2 caratteri).

4.1.2.6.3 Metodo C: parsing del contenuto di un SMS di tipo C (report configurazione)

Descrizione: il report di configurazione, inviato da Edor ad eCentral dopo aver ricevuto il relativo comando da parte di eCentral stesso, contiene i parametri di configurazione.

Implementazione

Nel metodo di firma:

private **Communication** ParseSMSMessageC(string[] plines)

si compiono le seguenti elaborazioni:

1. plines[1] di 10 caratteri contiene:
    a. la data di invio dell'SMS di tipo C (anno, mese, giorno) da parte di Edor nella sottostringa (0,5). Ogni campo è lungo 2 caratteri;
    b. l'ora (2 caratteri) e il minuto (2 caratteri) in cui è stata effettuata la misura del Signal nella sottostringa (6,9). L'istante di tempo del rilevamento del Signal è composta dall'anno, mese, giorno del punto 1.a dall'ora e dal minuto del punto in esame. I Signal di seguito avranno come timestamp tale istante di tempo di rilevamento.
2. plines[2] contiene 11 sottostringhe con la seguente struttura:
   <nomeSignal>=<valoreSignal>;
   sottointeso al tempo misurato al punto '1.b' si ha:

Elenco dei Signal

    a. versione base del firmware;
    b. versione dell'applicativo del firmware;
    c. configurazione Edor: richiede il parsing dei bit impostazioni "ParsingSettingFrame (set, retVal, SMSDateTime)";

d. numero telefonico del centro;
e. numero telefonico del reperibile;
f. ora e minuto a cui Edor è configurato per spedire il messaggio di tipo A (misure) ad eCentral;
g. periodo di misura in minuti (360 = 1 misura ogni 6 ore);
h. soglia bassa;
i. soglia alta;
j. tipo di sensore:
    1. TBM;
    2. THT.

4.1.2.6.4 Metodo D: parsing del contenuto di un SMS di tipo D (report taratura)

Descrizione: il report di taratura, inviato dall'Edor ad eCentral dopo aver ricevuto il relativo comando da parte di eCentral stesso, contiene i parametri di taratura. Sugli Edor dotati di Sampling System, l'SMS viene inviato anche dopo l'esecuzione dell'autoratura periodica.

Implementazione

Nel metodo di firma:

private **Communication** ParseSMSMessageD(string[] plines) si compiono le seguenti elaborazioni:

1. plines[1] di 10 caratteri contiene:
    a. la data di invio dell'SMS (anno, mese, giorno) nella sottostringa (0,5). Ogni campo è lungo 2 caratteri;
    b. l'ora e il minuto in cui è stata effettuata la misura del Signal nella sottostringa (6,9).

    L'istante di tempo del rilevamento del Signal è composta dall'anno, mese, giorno del punto 1.a dall'ora e dal minuto del punto in esame. I Signal di seguito avranno come timestamp tale istante di tempo di rilevamento.
2. plines[2] contiene 13 sottostringhe con la seguente struttura:
    <nomeSignal>=<valoreSignal>;

    Elenco di Signal

    a. tipo di sensore:
        1. TBM;
        2. THT.
    b. Gs Cell (guadagno amplificatore);
    c. Go gella (guadagno di etichetta);
    d. zero amplificatore (es. 1076 bit);
    e. fondoscala amplificatore (es. 3865 bit);
    f. zero catena (es. 1080 bit);
    g. tipo taratura:
        1. F: factory;

2. L: autolocal;
3. R: autoremote;
4. M: manual;

h. la data e l'ora dell'ultima taratura (anno, mese, giorno, ora, minuto);
i. intervallo di autotaratura (in giorni);
j. TO bombola campione moltiplicatore per 10 [mg/smc];
k. guadagno batteria (es. 3011 bit);
l. zero scala PT100 (es. 2381 bit);
m. fondoscala PT100 (es. 2901 bit).

4.1.2.6.5 Metodo E: parsing del contenuto di un SMS di tipo E (report eventi)

Descrizione: il report eventi contiene gli ultimi 10 eventi con data ed ora. I valori sono codificati in esadecimale.

Implementazione

Nel metodo di firma:

private **Communication** ParseSMSMessageE(string[] plines)

si compiono elaborazioni analoghe, in particolare:

1. plines[0] contiene gli ultimi 10 eventi con data ed ora:
   si posiziona il cursore con il metodo "IndexOf" al carattere successivo a quello di controllo ':' (trascurando così l'edor ID) in modo da considerare la sottostringa in cui sono contenuti i 10 eventi separati dal carattere di controllo ';'. Ogni evento è costituito da 10 caratteri esadecimali di cui i primi 8 costituiscono la data e l'ora in esadecimale dell'evento, mentre i restanti 2 il codice di errore in esadecimale. La prima sottostringa di 8 caratteri in realtà rappresenta il numero di secondi a partire dal 1/01/2000. Per ciascuna delle 10 sottostringhe si esegue:
   1. si crea un oggetto "CommunicationData" di nome "cd" e "Communication" di nome "retVal";
   2. si crea un oggetto DateTime dt = new DateTime(2000, 01, 01);
   3. si converte la sottostringa (0,7) in un valore double rappresentante il numero di secondi;
   4. si aggiunge i secondi alla data del punto 2, dt = dt.AddSeconds(seconds), per ottenere la data aggiornata;
   5. si converte la restante sottostringa (8,9) in un valore Integer con segno a 32 bit che si confronta, tramite uno switch-case (nel metodo ParsingEventFrame), con ciascuna constante che rappresenta un possibile evento "Constants.EventCode". Una volta trovata la corrispondenza si setta a X il campo "LLSignalID" di "cd" sfruttando la costante X corrispondente "eCentralConstants.Signal.Edor.Events.X";
   6. si setta il campo "valueB" a true;
   7. si setta il campo "dataAcq" al valore del punto 4;
   8. si aggiunge al campo "data" di *retVal* l'oggetto "cd" settato opportunamente.

4.1.2.6.6 Metodo F: parsing del contenuto di un SMS di tipo F (report totalizzatori)

Descrizione: il report totalizzatore contiene informazioni sulla "vita" della cella elettrochimica.

Implementazione

Nel metodo di firma:

private **Communication** ParseSMSMessageF(string[] plines)

si compiono le seguenti elaborazioni:

1. plines[1] di 6 caratteri contiene la data di invio dell'SMS (anno di 2 caratteri, mese di 2 caratteri, giorno di 2 caratteri);
   I Signal nel seguito avranno come timestamp l'istante di tempo del punto 1.
2. plines[2] contiene 7 sottostringhe con la seguente struttura:
   <nomeSignal>:<valoreSignal>;

   Elenco di Signal

   a. tempo totale di misura gas in secondi dall'attivazione della cella;
   b. tempo totale di misura evacuazione dall'attivazione della cella;
   c. numero totale di misure dall'attivazione della cella;
   d. somma temperature dall'attivazione della cella;
   e. numero di seriale dell'Edor;
   f. numero di serie della cella;
   g. data di attivazione della cella.

4.1.2.6.7 Metodo ACK: parsing del contenuto di un SMS di tipo ACK (conferma ricezione sms di programmazione)

Descrizione: SMS di conferma della corretta ricezione della programmazione da parte dell'Edor.

Implementazione

Nel metodo di firma:

private **Communication** ParseSMSMessageACK(string[] plines)

si eseguono elaborazioni analoghe basate sempre sulla conoscenza a priori della lunghezza dei campi costituenti una data "plines":

1. plines[1] di 19 caratteri di cui si considerano i primi 6 caratteri per la data di invio dell'SMS (anno di 2 caratteri, mese di 2 caratteri, giorno di 2 caratteri). La sottostringa seguente "- SMS RICEVUTO" per completare i 19 caratteri viene trascurata, perchè già utilizzata;

I Signal di seguito avranno come timestamp l'istante di tempo del punto 1.

2. plines[2] di 10 caratteri contiene:

Elenco di Signal

a. tipo di sensore (1 carattere):
    1. TBM;
    2. THT.
b. Gs Cella (guadagno amplificatore, 3 caratteri);
c. tensione di batteria (4 caratteri);
d. campo GSM (2 caratteri).

Nella figura 4.3 viene mostrata la struttura di una comunicazione contenente LLSignal.

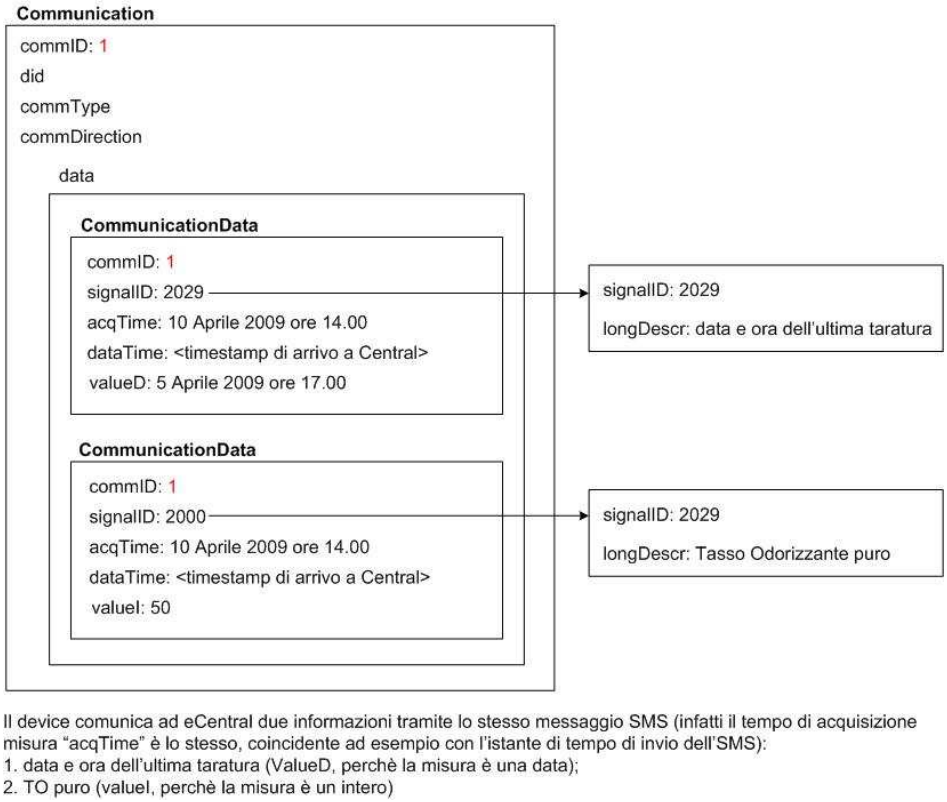


Figura 4.3: Communication

4.1.2.7 Metodi di parsing

4.1.2.7.1 ParseDiagnosticFrame

Il metodo di parsing per il contenuto di diagnostica avente la seguente firma:

private void ParseDiagnosticFrame(int pDiag, Communication pRetVal, DateTime pmeasureDateTime)

riceve:

- il corrispondente intero dei 4 caratteri rappresentanti il sottocampo "diagnostica" del messaggio di tipo B, di nome "pDiag";
- l'oggetto Communication da completare (parziale perchè possiede solo la data);
- istante di tempo della misura, di nome "pmeasureDateTime";

Il parsing viene effettuato compiendo un'operazione logica AND bit a bit su "pDiag" con una maschera esadecimale che varia in funzione del tipo di diagnostica da detectare. Compiendo uno switch su pDiag, per esempio, per individuare l' "un overflow sulla misura TO" si effettua il seguente controllo bit a bit:

if (pDiag & maschera<overflow sulla misura TO>)

0=no overflow sulla misura TO

1=sì overflow sulla misura TO

Dove "maschera<overflow sulla misura TO>" vale 0x0001.

Se il controllo è maggiore di 0 si crea l'oggetto Communicationdata di cui si setta:

1. il campo "dataTime" all'istante di tempo di misura;
2. il campo LLSignalID alla costante "eCentralConstants.Signal.Edor.Settings.ENOS_DIAGN_VALUE_OVF" (perchè 1>0);
3. il campo "ValueB" a true;

Si inserisce, infine, l'oggetto CommunicationData alla lista di CommunicationData.

4.1.2.7.2 ParsingSettingFrame

Il metodo di parsing per il contenuto di programmazione degli Edor avente la seguente firma:

private void ParsingSettingFrame (int pSet, Communication pRetVal, DateTime pSMSDateTime)

riceve:

- il corrispondente intero dei 2 caratteri rappresentanti il sottocampo "settaggi Edor" del messaggio di tipo C, di nome "pSet";
- l'oggetto Communication da completare (parziale perchè possiede solo la data);
- istante di tempo dell'invio dell'SMS, di nome "pSMSDateTime";

Il parsing viene effettuato compiendo un'operazione logica AND bit a bit su "pSet" con una maschera esadecimale che varia in funzione del tipo di impostazione da rilevare (vedi tabella dei bit delle impostazioni). Compiendo uno switch su pSet, per esempio, per individuare la "disabilitazione invio sms" effettuo il seguente controllo bit a bit:

if (pSet & maschera<disabilitazione invio sms>)

0=sms attivi

1=sms disabilitati

Dove "maschera< disabilitazione invio sms >" vale 0x01.

Se il controllo è maggiore di 0 si crea l'oggetto Communicationdata di cui si setta:

1. il campo "dataTime" alla data di invio dell'SMS;
2. il campo LLSignalID alla costante "eCentralConstants.Signal.Edor.Settings.ENOS_SET_SMS_DISABLE" (perchè 1>0);
3. il campo "ValueB" a true;

Si aggiunge, infine, l'oggetto CommunicationData alla lista di CommunicationData.

4.1.2.7.3 ParsingEventFrame

Il metodo di parsing per il contenuto dei messaggi di tipo E avente la seguente firma:

private void ParsingEventFrame(int pEvent, Communication pRetVal, DateTime pdt)

Si crea un oggetto di nome ‘cd’ della classe CommunicationData e sulla base del matching di pEvent con un certo EventCode, si setta il campo LLSignalID di *cd* con l’evento trovato e si setta il ValueB a true, come di seguito riportato in esempio:

```
switch (pEvent)
{
    case Constants.EventCode.ERR_RESET:
    cd.LLSignalID = eCentralConstants.Signal.Edor.Events.ERR_RESET;
    cd.ValueB = true;
    break;
...
```

Alla fine

Se il riferimento a cd non è nullo si setta:

cd.DataTime = pdt;

e si aggiunge all’array di ComunicationData l’oggetto CommunicationData “cd”:

pRetVal.Data.Add(cd);

4.2 Query

Nel seguito sono descritte alcune interrogazioni per verificare la correttezza del modello dei dati progettato. Le query sono eseguite dopo aver inserito opportuni record nelle corrispondenti tabelle del database di testing denominato *ECENTRAL_PMASCHI*.

4.2.1 Query n.1

Testo:

Dato un impianto ed un intervallo di tempo (data1 → data2), trovare per ogni device, la lista di eventi che attivano un allarme. **N.B.:** non sono richiesti tutti gli eventi prodotti, ma solo quelli di attivazione.

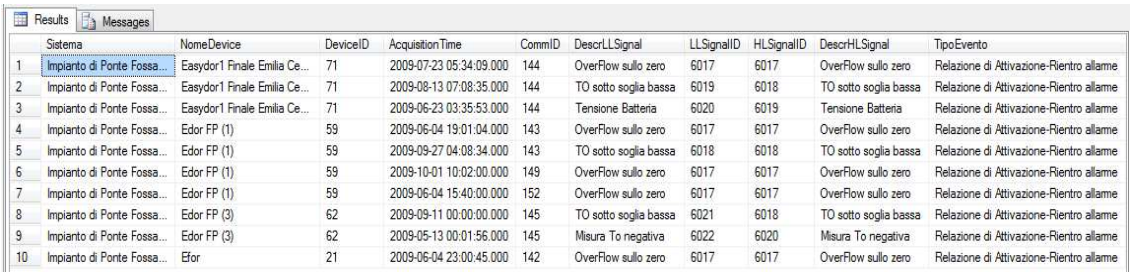
Script SQL:

```
use ECENTRAL_PMASCHI;

DECLARE @d1 datetime;
DECLARE @d2 datetime;
SET @d1 = '2009-01-01 12:00:04'; -- anno, mese, giorno, ora, minuti, secondi
SET @d2 = '2009-10-12 12:00:09';
DECLARE @GID int;
SET @GID = 7;

SELECT     G.LongDescr AS Sistema, D.ShortDescr AS NomeDevice, D.DID AS DeviceID, CD.AcqTime AS AcquisitionTime,
CD.CommID, A3.LongDescr AS DescrLLSignal, LL.LLSignalID, HL.HLSignalID, A1.LongDescr AS DescrHLSignal,
A2.LongDescr AS TipoEvento
FROM       [Group] AS G INNER JOIN
GroupMShipD AS GMD ON GMD.GID = G.GID INNER JOIN
dbo.Device AS D ON D.DID = GMD.MemberDID INNER JOIN -- dato il sistema si individuano i device
dbo.Communication AS C ON C.DID = D.DID INNER JOIN -- per ogni device si recuperano le sue comunicazioni
dbo.CommunicationData AS CD ON CD.CommID = C.CommID INNER JOIN -- per ogni comunicazione compiuta si individuano
i valori delle misure LLSignal acquisite da device
dbo.LowLevelSignal AS LL ON LL.LLSignalID = CD.LLSignalID INNER JOIN -- per ogni valore si recuperano le informazioni
del corrispondente LLSignal (LLSignal è dinamica)
dbo.HighLevelSignal AS HL ON HL.HLSignalID = LL.HLSignalID INNER JOIN -- si recupera l'HLSignal corrispondente
all'LLSignal in esame
dbo.H2HRShipElem AS HSE ON HSE.HLSignalID1 = HL.HLSignalID INNER JOIN -- si controlla se l'HLSignal in esame sia un
evento di attivazione nella tabella statica
-- H2HRShipElem, che conserva tutte le possibili coppie di eventi consentite
dbo.H2HRShip AS HS ON HS.H2HRSID = HSE.H2HRSID INNER JOIN -- si controlla il tipo di relazione tra eventi
dbo.I18NAbstractionLayer AS I1 ON I1.IALID = HL.IALID INNER JOIN -- interazione con l'abstraction layer di culture
dbo.ApplicationObjectI18N AS A1 ON A1.IALID = I1.IALID INNER JOIN -- si recupera la descrizione dell'HLSignal in funzione
della lingua
dbo.I18NAbstractionLayer AS I2 ON I2.IALID = HS.IALID INNER JOIN -- interazione con l'abstraction layer di culture
dbo.ApplicationObjectI18N AS A2 ON A2.IALID = I2.IALID INNER JOIN -- si recupera il tipo di evento in funzione della lingua
dbo.I18NAbstractionLayer AS I3 ON I3.IALID = LL.IALID INNER JOIN -- interazione con l'abstraction layer di culture
dbo.ApplicationObjectI18N AS A3 ON A3.IALID = I3.IALID -- si recupera la descr di LLSignal in funzione della lingua
WHERE     (LL.STID = 3) AND (HS.H2HRSID = 1) AND (CD.acqTime BETWEEN @d1 AND @d2) AND (G.GID = @GID)
ORDER BY NomeDevice
-- si specifica di volere i LLevelSignal di tipo evento, si specifica di volere la relazione arrivaz-rientro, si specifica l'intervallo di
tempo in cui analizzare
-- gli eventi
```

Risultati:



Results | Messages

	Sistema	NomeDevice	DeviceID	AcquisitionTime	CommID	DescrLLSignal	LLSignalID	HLSignalID	DescrHLSignal	TipoEvento
1	Impianto di Ponte Fossa...	Easydor1 Finale Emilia Ce...	71	2009-07-23 05:34:09.000	144	OverFlow sullo zero	6017	6017	OverFlow sullo zero	Relazione di Attivazione-Rientro allarme
2	Impianto di Ponte Fossa...	Easydor1 Finale Emilia Ce...	71	2009-08-13 07:08:35.000	144	TO sotto soglia bassa	6019	6018	TO sotto soglia bassa	Relazione di Attivazione-Rientro allarme
3	Impianto di Ponte Fossa...	Easydor1 Finale Emilia Ce...	71	2009-06-23 03:35:53.000	144	Tensione Batteria	6020	6019	Tensione Batteria	Relazione di Attivazione-Rientro allarme
4	Impianto di Ponte Fossa...	Edor FP (1)	59	2009-06-04 19:01:04.000	143	OverFlow sullo zero	6017	6017	OverFlow sullo zero	Relazione di Attivazione-Rientro allarme
5	Impianto di Ponte Fossa...	Edor FP (1)	59	2009-09-27 04:08:34.000	143	TO sotto soglia bassa	6018	6018	TO sotto soglia bassa	Relazione di Attivazione-Rientro allarme
6	Impianto di Ponte Fossa...	Edor FP (1)	59	2009-10-01 10:02:00.000	149	OverFlow sullo zero	6017	6017	OverFlow sullo zero	Relazione di Attivazione-Rientro allarme
7	Impianto di Ponte Fossa...	Edor FP (1)	59	2009-06-04 15:40:00.000	152	OverFlow sullo zero	6017	6017	OverFlow sullo zero	Relazione di Attivazione-Rientro allarme
8	Impianto di Ponte Fossa...	Edor FP (3)	62	2009-09-11 00:00:00.000	145	TO sotto soglia bassa	6021	6018	TO sotto soglia bassa	Relazione di Attivazione-Rientro allarme
9	Impianto di Ponte Fossa...	Edor FP (3)	62	2009-05-13 00:01:56.000	145	Misura To negativa	6022	6020	Misura To negativa	Relazione di Attivazione-Rientro allarme
10	Impianto di Ponte Fossa...	Efor	21	2009-06-04 23:00:45.000	142	OverFlow sullo zero	6017	6017	OverFlow sullo zero	Relazione di Attivazione-Rientro allarme

Figura 4.4: risultati query n.1

4.2.2 Query n.2

Testo:

Dato un device, individuarne gli allarmi attivi ad una certa data.

Script SQL:

```
use ECENTRAL_PMASCHI;

DECLARE @istante_di_tempo datetime;
SET @istante_di_tempo = '2009-12-31 12:00:04'; -- anno, mese, giorno, ora, minuti, secondi
DECLARE @DID int;
SET @DID = 59;

SELECT HL1.HLSignalID AS AllarmeAttivo,  A.LongDescr AS Descrizione -- Un gruppetto di un dato evento di attivazione (batteria, TO sotto soglia, etc.) deve avere AcqTime maggiore del gruppetto del corrispondente evento di rientro (ricavato dalla tabella statica H2HRShipElem)
FROM dbo.Device AS D1 INNER JOIN
dbo.Communication AS C1 ON C1.DID = @DID INNER JOIN -- si recuperano le comunicazioni del device @DID
dbo.CommunicationData AS CD1 ON CD1.CommID = C1.CommID INNER JOIN -- per ogni comunicazione compiuta si individuano gli LLSignal
dbo.LowLevelSignal AS LL1 ON LL1.LLSignalID = CD1.LLSignalID INNER JOIN -- si recuperano le informazioni del corrispondente LLSignal (LLSignal è dinamica)
dbo.HighLevelSignal AS HL1 ON HL1.HLSignalID = LL1.HLSignalID INNER JOIN -- si recupera l'HLSignal corrispondente all'LLSignal in esame per conoscere la sua descrizione,
 -- cioè per avere lo IALID con cui recuperare LongDescr di ApplicationObjectI18N. Se
 -- non servisse di potrebbe utilizzare direttamente l'HLSignalID del LLSignal per andare
 -- in join su H2HRShipElmem
dbo.H2HRShipElem AS HSE1 ON HSE1.HLSignalID1 = HL1.HLSignalID INNER JOIN -- si prende l'HLSignal di attivazione (join su HLSignalID1)
dbo.H2HRShip AS HS1 ON HS1.H2HRSID = HSE1.H2HRSID INNER JOIN -- si controlla il tipo di relazione tra eventi, che deve essere "attivazione-rientro"
dbo.I18NAbstractionLayer AS I ON I.IALID = HL1.IALID INNER JOIN -- interazione con il layer della lingua
dbo.ApplicationObjectI18N AS A ON A.IALID = I.IALID -- si recupera la descrizione dell'HLSignal in funzione della lingua
                        -- interazione con l'abstraction layer di culture
WHERE     ((LL1.STID = 3) AND (HS1.H2HRSID = 1) AND (CD1.acqTime <= @istante_di_tempo))
-- LL1.STID = 3, perchè lo LLSignal deve essere di tipo "evento" (un trigger determina automaticamente il filtraggio HL1.STID = 3, che quindi non si deve fare, perchè superfluo)
-- HS1.H2HRSID = 1, perchè la relazione deve essere di tipo "attivazione-rientro"
-- D1.DID = @DID, perchè devo filtrare su tutti i device per ottenere quello richiesto
-- CD1.acqTime <= @istante_di_tempo, perchè devo considerare lo storico
GROUP BY HL1.HLSignalID, C1.DID, A.LongDescr -- raggruppo per evento di attivazione (batteria, TO sotto soglia, etc.) e per ogni gruppo prendo l'ultimo evento in ordine temporale.
-- Affinchè l'allarme sia attivo, l'ultimo evento di attivazione deve avere un tempo maggiore dell'ultimo evento di rientro corrispondente.
-- Come operatore di confronto utilizzo il "maggiore (>)" e non il ">=", poiché ritengo errato che il device comunichi
-- di essere entrato in allarme e che, nello stesso istante di tempo, l'allarme è rientrato.
HAVING MAX(CD1.acqTime) > (SELECT MAX(CD2.acqTime) -- prendo il massimo fra i tempi di acquisizione dell'evento di rientro rispetto a quello di attivazione in esame
                                FROM dbo.Communication AS C2 INNER JOIN
                                dbo.CommunicationData AS CD2 ON CD2.CommID = C2.CommID INNER JOIN -- per ogni comunicazione compiuta si individuano gli LLSignal
                                dbo.LowLevelSignal AS LL2 ON LL2.LLSignalID = CD2.LLSignalID INNER JOIN -- si recuperano le informazioni del corrispondente LLSignal (LLSignal è dinamica)
```

```
-- dbo.HighLevelSignal AS HL2 ON HL2.HLSignalID = LL2.HLSignalID INNER JOIN -- join inutile, perchè non mi servono informazioni sull'HLSignal
dbo.H2HRShipElem AS HSE2 ON HSE2.HLSignalID2 = LL2.HLSignalID INNER JOIN-- si recuperano tutti gli eventi di rientro corrispondenti all'evento di attivazione in esame
dbo.H2HRShip AS HS2 ON HS2.H2HRSID = HSE2.H2HRSID -- si controlla il tipo di relazione tra eventi
WHERE     (LL2.STID = 3) AND (HS2.H2HRSID = 1) AND (CD2.acqTime <= @istante_di_tempo)
AND  (C2.DID = C1.DID) -- si recuperano le comunicazioni del device fissato esternamente
AND  (HSE2.HLSignalID1 = HL1.HLSignalID)) -- fisso l'evento di attivazione pari a quello in esame (cioè della query esterna)
```

Risultati:

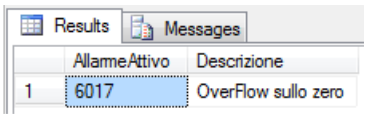


Figura 4.5: risultati query n.2

4.2.3 Query n.3

Testo:

Dato un impianto ed un intervallo di tempo (data1 → data2), ricavare la lista di allarmi più frequenti.

Script SQL:

```
use ECENTRAL_PMASCHI;

DECLARE @d1 datetime;
DECLARE @d2 datetime;
SET @d1 = '2009-01-01 12:00:04'; -- anno, mese, giorno, ora, minuti, secondi
SET @d2 = '2009-10-12 12:00:09';
DECLARE @GID int;
SET @GID = 7;

SELECT     A1.LongDescr AS DescrHLSignal, COUNT(*) as QTY
FROM      dbo.GroupMShipD AS GMD INNER JOIN
dbo.Device AS D ON D.DID = GMD.MemberDID INNER JOIN -- dato il sistema si individuano i device
dbo.Communication AS C ON D.DID = C.DID INNER JOIN -- per ogni device si recuperano le sue comunicazioni
dbo.CommunicationData AS CD ON CD.CommID = C.CommID INNER JOIN -- per ogni comunicazione compiuta si individuano i valori delle misure LLSignal acquisite da device
dbo.LowLevelSignal AS LL ON LL.LLSignalID = CD.LLSignalID INNER JOIN -- per ogni valore si recuperano le informazioni del corrispondente LLSignal (LLSignal è dinamica)
dbo.HighLevelSignal AS HL ON HL.HLSignalID = LL.HLSignalID INNER JOIN -- si recupera l'HLSignal corrispondente all'LLSignal in esame
dbo.H2HRShipElem AS HSE ON HSE.HLSignalID1 = HL.HLSignalID INNER JOIN -- si controlla se l'HLSignal in esame sia un evento di attivazione nella tabella statica
-- H2HRShipElem, che conserva tutte le possibili coppie di eventi consentite
dbo.H2HRShip AS HS ON HS.H2HRSID = HSE.H2HRSID INNER JOIN -- si controlla il tipo di relazione tra eventi
dbo.I18NAbstractionLayer AS I1 ON I1.IALID = HL.IALID INNER JOIN -- interazione con l'abstraction layer di culture
dbo.ApplicationObjectI18N AS A1 ON A1.IALID = I1.IALID INNER JOIN -- si recupera la descrizione dell'HLSignal in funzione della lingua
dbo.I18NAbstractionLayer AS I2 ON I2.IALID = HS.IALID INNER JOIN -- interazione con l'abstraction layer di culture
dbo.ApplicationObjectI18N AS A2 ON A2.IALID = I2.IALID INNER JOIN -- si recupera il tipo di evento in funzione della lingua
dbo.I18NAbstractionLayer AS I3 ON I3.IALID = LL.IALID INNER JOIN -- interazione con l'abstraction layer di culture
dbo.ApplicationObjectI18N AS A3 ON A3.IALID = I3.IALID -- si recupera la descr di LLSignal in funzione della lingua
WHERE     (LL.STID = 3) AND (HS.H2HRSID = 1) AND (CD.acqTime BETWEEN @d1 AND @d2) AND (GMD.GID = @GID)
GROUP BY A1.LongDescr
ORDER BY QTY DESC
-- si specifica di volere i LowLevelSignal di tipo evento, si specifica di volere la relazione arrivaz-rientro, si specifica l'intervallo di tempo in cui analizzare
-- gli eventi
```

Risultati:

Results | Messages

	DescrHLSignal	QTY
1	OverFlow sullo zero	5
2	TO sotto soglia bassa	3
3	Misura To negativa	1
4	Tensione Batteria	1

Figura 4.6: risultati query n.3

4.2.4 Query n.4

Testo:

Individuare la composizione di un sistema in modo statico (senza riferimenti temporali), indicando quali sono gli impianti (gruppi di device). Valutare *EvaluareRecursively* per decicere se includere o meno gli appartenenti al gruppo tra i risultati.

Rappresentazione grafica

Nella figura 4.7 è rappresentata la composizione del sistema di odorizzazione di Sassuolo, in tre diverse configurazioni espressamente enumerate:

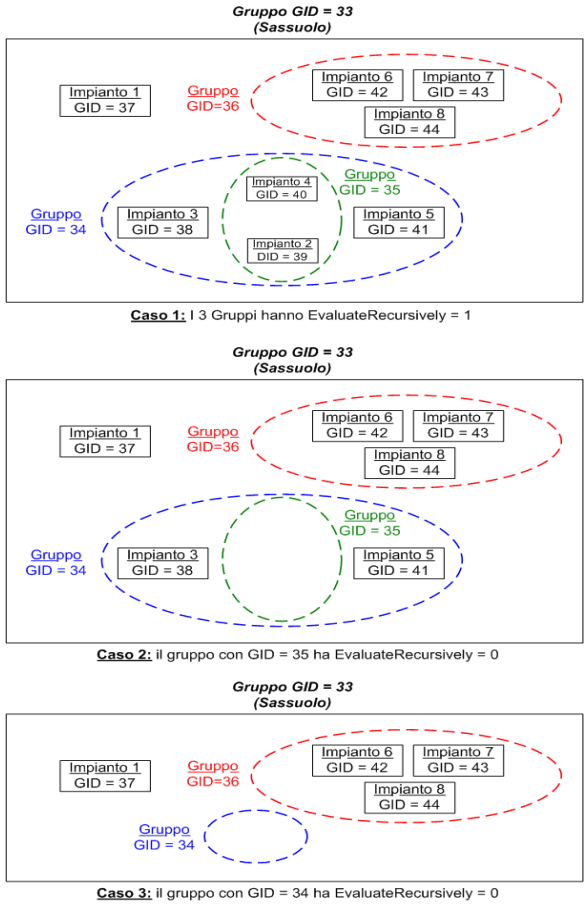


Figura 4.7: composizione del sistema di odorizzazione di Sassuolo

Script SQL:

```
use ECENTRAL_PMASCHI;

DECLARE @GID int;
SET @GID = 33;

select GID, ShortDescr, LongDescr
from [group]
where gid=@gid;

WITH tree(GID, MemberGID, GroupLevel, EvaluateRecursively) AS
(
    select distinct GMS.GID, MemberGID, 0 AS GroupLevel, GMS.EvaluateRecursively from dbo.GroupMShipG as GMS,
dbo.GroupMShipD as Impianti -- ANCHOR MEMBER
    where (GMS.GID = @GID)
    UNION ALL
    select G_arrivo.GID, G_arrivo.MemberGID, GroupLevel + 1, G_arrivo.EvaluateRecursively from dbo.GroupMShipG as
G_arrivo -- RECURSIVE MEMBER
    INNER JOIN tree t on G_arrivo.GID = t.MemberGID
    where t.EvaluateRecursively = 1
)
select t.GID, t.MemberGID, g.shortDescr as ShortDescrMemberGID, g.longDescr as LongDescrMemberGID
from tree as t join [group] as g on g.GID = t.memberGID
```

Nel seguito sono mostrati i risultati classificati in funzione del tipo di composizione del sistema di odorizzazione:

Caso 1

Results | Messages

	GID	ShortDescr	LongDescr
1	33	Sassuolo	Città di Sassuolo

	GID	MemberGID	ShortDescrMemberGID	LongDescrMemberGID
1	33	37	Impianto 1	Impianto di Via Goldoni
2	33	34	Sistema di odorizzazione	Sistema di odorizzazione - zona industriale
3	33	36	Sistema di odorizzazione	Sistema di odorizzazione - centro storico
4	36	42	Impianto 6	Impianto di Via Tassoni
5	36	43	Impianto 7	Impianto di Via Mozart
6	36	44	Impianto 8	Impianto di Via Beethoven
7	34	35	Gruppo Logico Impianti	Gruppo Logico Impianti
8	34	38	Impianto 3	Impianto di Via Respighi
9	34	41	Impianto 5	Impianto di Via Montanara
10	35	39	Impianto 2	Impianto di Via Ravetta
11	35	40	Impianto 4	Impianto di Via Ferrari

Figura 4.8: risultati caso 1

Caso 2:

Viene mostrata come ulteriore prova della correttezza dei risultati il valore dell'EvaluateRecursively (ultima colonna) della tabella GroupMShipG in figura 4.9:

35	34	35	04/06/2009 09:...	False	1	False

Figura 4.9: GroupMShipG

I risultati della query sono:

Results | Messages

	GID	ShortDescr	LongDescr
1	33	Sassuolo	Città di Sassuolo

	GID	MemberGID	ShortDescrMemberGID	LongDescrMemberGID
1	33	37	Impianto 1	Impianto di Via Goldoni
2	33	34	Sistema di odorizzazione	Sistema di odorizzazione - zona industriale
3	33	36	Sistema di odorizzazione	Sistema di odorizzazione - centro storico
4	36	42	Impianto 6	Impianto di Via Tassoni
5	36	43	Impianto 7	Impianto di Via Mozart
6	36	44	Impianto 8	Impianto di Via Beethoven
7	34	35	Gruppo Logico Impianti	Gruppo Logico Impianti
8	34	38	Impianto 3	Impianto di Via Respighi
9	34	41	Impianto 5	Impianto di Via Montanara

Figura 4.10: risultati caso 2

Caso 3:

Viene mostrata come ulteriore prova della correttezza dei risultati il valore dell'EvaluateRecursively (ultima colonna) della tabella GroupMShipG in figura 4.11:

34	33	34	04/06/2009 09:...	False	1	False

Figura 4.11: GroupMShipG

I risultati della query sono:

Results | Messages

	GID	ShortDescr	LongDescr
1	33	Sassuolo	Città di Sassuolo

	GID	MemberGID	ShortDescrMemberGID	LongDescrMemberGID
1	33	34	Sistema di odorizzazione	Sistema di odorizzazione - zona industriale
2	33	37	Impianto 1	Impianto di Via Goldoni
3	33	36	Sistema di odorizzazione	Sistema di odorizzazione - centro storico
4	36	42	Impianto 6	Impianto di Via Tassoni
5	36	43	Impianto 7	Impianto di Via Mozart
6	36	44	Impianto 8	Impianto di Via Beethoven

Figura 4.12: Risultati caso 3

4.2.5 Query n.5

Testo:

Individuare le misure (STID=1) dei dispositivi (mobili e non) solo negli intervalli utili; le misure devono essere attribuite all'impianto a cui il device appartiene solo per quei periodi in cui il device appartiene appunto a tale impianto. Per ogni intervallo di legame indicare il numero di misure di un dato Signal. Calcolare, infine, il numero totate di misure acquisite per device. (SQLQueryTestata per visio)

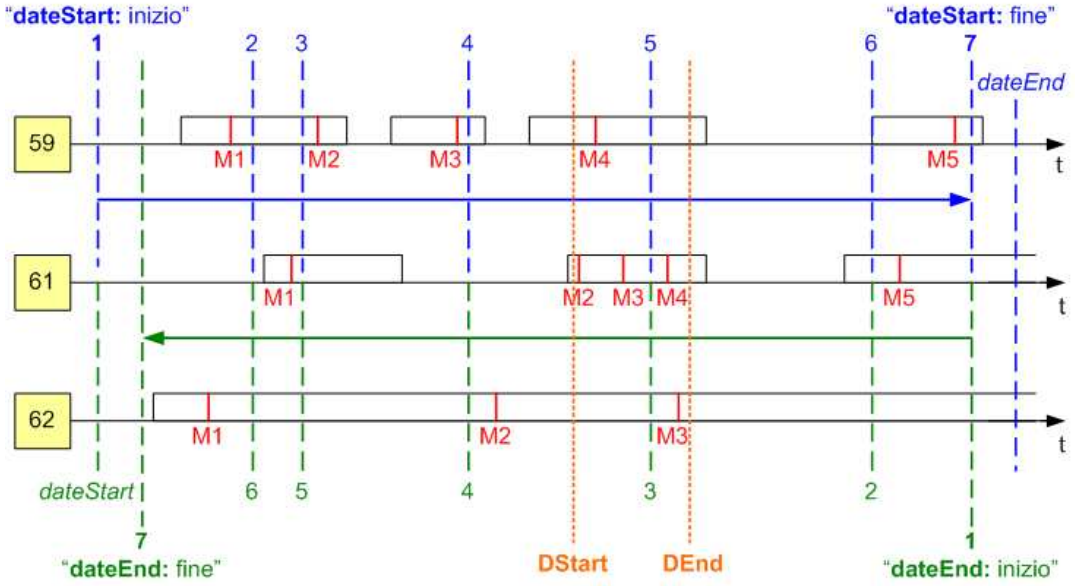
La tabella di riferimento utilizzata per tale query è mostrata in figura 4.13.

	GID	memberDID	EffectiveSince	IsRevoked
▶	50	61	01/01/2009 00:00:00	False
	37	62	11/01/2009 11:46:52	False
	37	59	02/02/2009 04:00:00	False
	37	61	24/02/2009 17:45:00	False
	37	59	02/03/2009 08:30:00	True
	37	59	05/05/2009 09:04:16	False
	37	61	07/05/2009 04:46:42	True
	37	59	07/07/2009 22:30:00	True
	37	59	12/08/2009 07:23:32	False
	37	61	13/08/2009 13:00:25	False
	37	59	04/10/2009 08:00:00	True
	37	61	07/10/2009 06:11:45	True
	37	61	06/11/2009 06:30:25	False
	37	59	08/11/2009 11:00:34	False
	37	59	18/11/2009 00:00:22	True
*	NULL	NULL	NULL	NULL

Figura 4.13: tabella di riferimento.

Nella figura 4.14 di pag. 216 viene descritta la procedura per testare la correttezza della query creata.

Con le linee tratteggiate verticali di colore blu si rappresentano le possibili collocazioni temporali della data di inizio analisi **dateStart** ("dateStart: inizio" a "dateStart: fine"), mantenendo fissa quella di fine analisi *dateEnd* in blu. La data di inizio **dateStart**, pertanto, assume le posizioni in blu: 1, 2, 3, 4, 5, 6 e 7. Con le linee tratteggiate verticali di colore verde si rappresentano le possibili collocazioni temporali della data di fine analisi **dateEnd** ("dateEnd: inizio" a "dateEnd: fine"), mantenendo fissa quella di inizio analisi *dateStart* in verde. La data di fine analisi **dateEnd**, pertanto, assume le posizioni in verde: 1, 2, 3, 4, 5, 6 e 7. Si è, poi, esaminato un intervallo preciso (**dStart** to **dEnd**) tra le linee tratteggiate di colore arancio. Le linee tratteggiate intersecano tutti e tre gli assi temporali. Nei riquadri in giallo sono indicati gli identificatori dei device (deviceID o DID).



	DID = 59				
	Misura	*LLSignalID*	*T_Acq*	*EffSince_Start*	*EffSince_End*
	M1	7001	22/02/2009 04:00:00	02/02/2009 02:30:00	02/03/2009 08:30:00
	M2	7001	01/03/2009 14:00:15	02/02/2009 02:30:00	02/03/2009 08:30:00
	M3	7000	18/06/2009 23:15:04	05/05/2009 09:04:16	07/07/2009 22:30:00
1	M4	7001	15/08/2009 10:00:00	12/08/2009 07:23:32	04/10/2009 08:00:00
	M5	7000	16/11/2009 11:22:31	08/11/2009 11:00:34	18/11/2009 00:00:22

	DID = 61				
	Misura	*LLSignalID*	*T_Acq*	*EffSince_Start*	*EffSince_End*
	M1	7000	22/02/2009 04:00:00	24/02/2009 17:45:05	07/05/2009 04:46:42
2	M2	7001	14/08/2009 07:22:45	13/08/2009 13:00:25	07/10/2009 06:11:45
3	M3	7001	18/08/2009 19:00:32	13/08/2009 13:00:25	07/10/2009 06:11:45
4	M4	7001	25/09/2009 09:42:35	13/08/2009 13:00:25	07/10/2009 06:11:45
	M5	7001	14/11/2009 05:00:49	06/11/2009 06:30:25	null

	DID = 62				
	Misura	*LLSignalID*	*T_Acq*	*EffSince_Start*	*EffSince_End*
	M1	7000	19/01/2009 12:00:00	11/01/2009 11:46:52	null
	M2	7001	09/07/2009 14:50:14	11/01/2009 11:46:52	null
5	M3	7000	30/09/2009 20:25:40	11/01/2009 11:46:52	null

Figura 4.14: procedura di testing

Nel seguito vengono mostrate le figure 4.15 ÷ 4.21 che contengono i risultati per i casi 1-2-3-4-5-6-7 mostrati nella figura 4.14, che rappresentano lo spostamento temporale in avanti della data di inizio analisi.

Caso 1

Si evidenziano le date di inizio e di fine analisi:

SET @d1='2009-01-11 01:00:00';

SET @d2='2009-11-20 18:22:19';

```
DECLARE @d1 as datetime;
SET @d1='2009-01-11 01:00:00';
DECLARE @d2 as datetime;
SET @d2='2009-11-20 18:22:19';
DECLARE @COUNT as int;
SET @COUNT=0;
DECLARE @GID as int;
SET @GID = 37
DECLARE @DID_Start as int;
DECLARE @EffectiveSince_Start as datetime;
DECLARE @DID_End as int;
DECLARE @EffectiveSince_End as datetime;
DECLARE @EffectiveSince_End_Supporto as datetime;
DECLARE @dID_Attuale as int
DECLARE @FETCH_STATUS_RICERCA AS INT
DECLARE @FETCH_STATUS_SINGOLO AS INT
DECLARE @FETCH_STATUS_CStart AS INT
DECLARE @statoC_End AS INT
DECLARE @correctStart AS datetime

create table #TempTable (deviceID int, LLSignal int, QTYmisure int)

-- il GID è fisso
```

Results | Messages

	deviceID	LLSignal	QTYmisure
1	59	7001	2
2	59	7000	1
3	59	7001	1
4	59	7000	1
5	61	7000	1
6	61	7001	3
7	61	7001	1
8	62	7000	2
9	62	7001	1

Figura 4.15: risultati caso 1

Caso 2

Si evidenziano le date di inizio e di fine analisi:

SET @d1='2009-02-23 10:00:00';

SET @d2='2009-11-20 18:22:19';

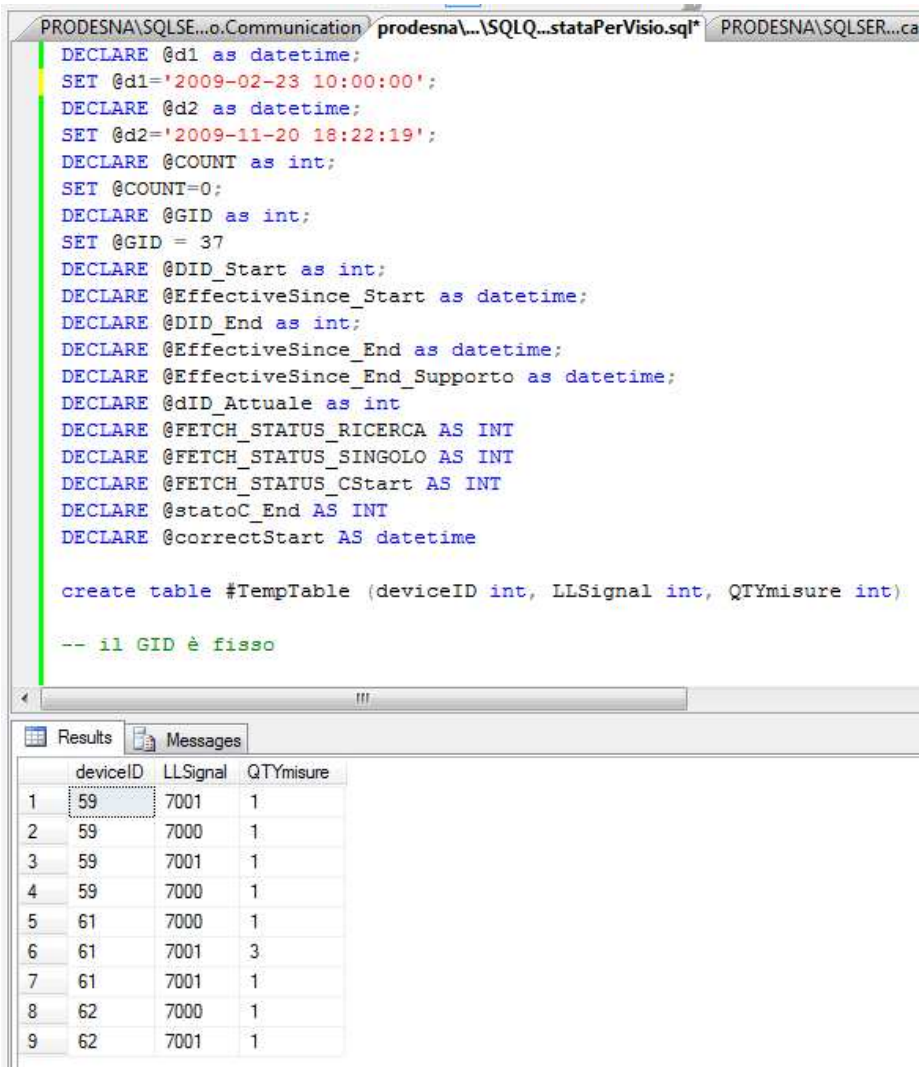


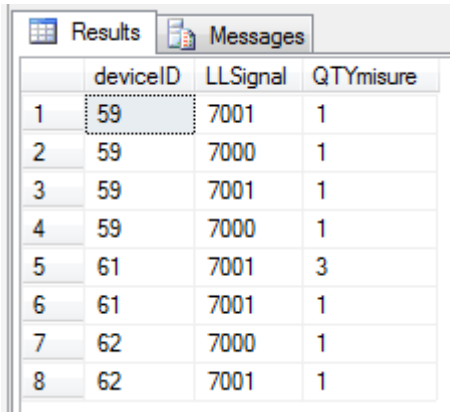
Figura 4.16: risultati caso 2

Caso 3

Si evidenziano le date di inizio e di fine analisi:

SET @d1='2009-02-28 10:00:00';

SET @d2='2009-11-20 18:22:19';



Results | Messages

	deviceID	LLSignal	QTYmisure
1	59	7001	1
2	59	7000	1
3	59	7001	1
4	59	7000	1
5	61	7001	3
6	61	7001	1
7	62	7000	1
8	62	7001	1

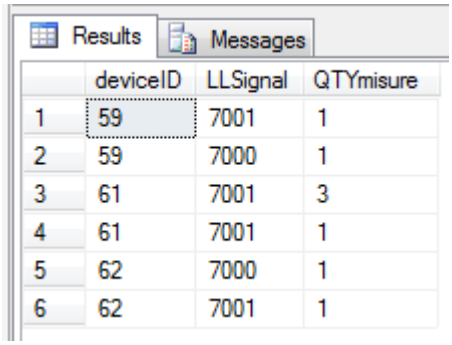
Figura 4.17: risultati caso 3

Caso 4

Si evidenziano le date di inizio e di fine analisi:

SET @d1='2009-06-20 10:00:00';

SET @d2='2009-11-20 18:22:19';



Results | Messages

	deviceID	LLSignal	QTYmisure
1	59	7001	1
2	59	7000	1
3	61	7001	3
4	61	7001	1
5	62	7000	1
6	62	7001	1

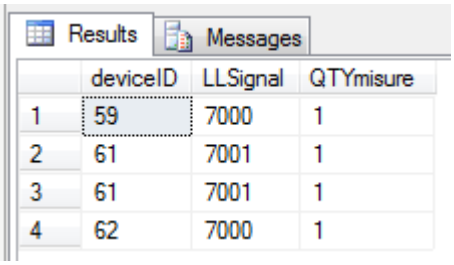
Figura 4.18: risultati caso 4

Caso 5

Si evidenziano le date di inizio e di fine analisi:

SET @d1='2009-08-19 10:00:00';

SET @d2='2009-11-20 18:22:19';



Results | Messages

	deviceID	LLSignal	QTYmisure
1	59	7000	1
2	61	7001	1
3	61	7001	1
4	62	7000	1

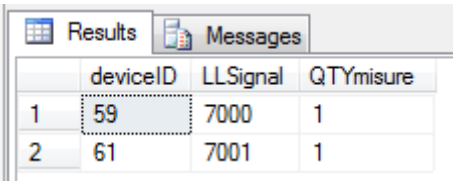
Figura 4.19: risultati caso 5

Caso 6

Si evidenziano le date di inizio e di fine analisi:

SET @d1='2009-11-8 11:00:34';

SET @d2='2009-11-20 18:22:19';



Results | Messages

	deviceID	LLSignal	QTYmisure
1	59	7000	1
2	61	7001	1

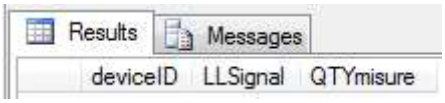
Figura 4.20: risultati caso 6

Caso 7

Si evidenziano le date di inizio e di fine analisi:

SET @d1='2009-11-19 11:00:34';

SET @d2='2009-11-20 18:22:19';



Results | Messages

deviceID	LLSignal	QTYmisure

Figura 4.21: risultati caso 7

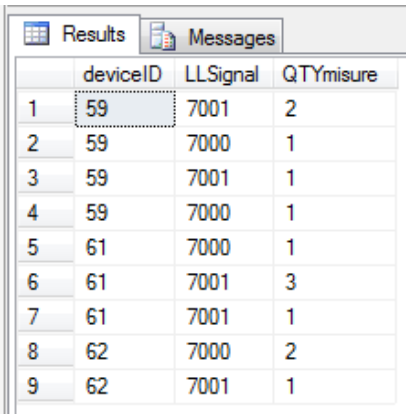
Nel seguito vengono mostrate le figure 4.22 ÷ 4.28 che contengono i risultati per i casi 7-6-5-4-3-2-1 mostrati nella figura 4.14, che rappresentano lo spostamento temporale a ritroso della data di fine analisi.

Caso 7

Si evidenziano le date di inizio e di fine analisi:

SET @d1='2009-01-11 01:00:00';

SET @d2='2009-11-19 11:00:34';



Results | Messages

	deviceID	LLSignal	QTYmisure
1	59	7001	2
2	59	7000	1
3	59	7001	1
4	59	7000	1
5	61	7000	1
6	61	7001	3
7	61	7001	1
8	62	7000	2
9	62	7001	1

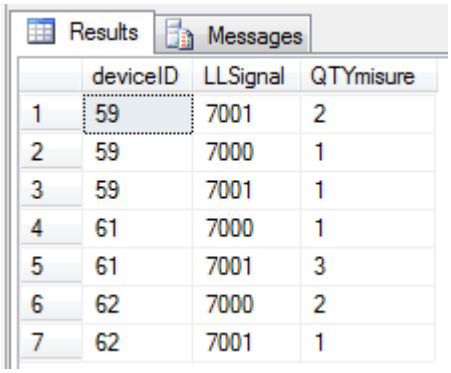
Figura 4.22: risultati caso 7

Caso 6

Si evidenziano le date di inizio e di fine analisi:

SET @d1='2009-01-11 01:00:00';

SET @d2='2009-11-8 11:00:34';



	deviceID	LLSignal	QTYmisure
1	59	7001	2
2	59	7000	1
3	59	7001	1
4	61	7000	1
5	61	7001	3
6	62	7000	2
7	62	7001	1

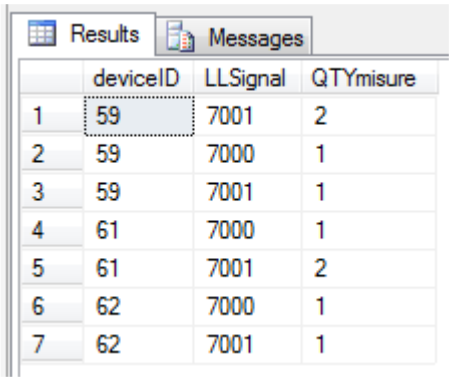
Figura 4.23: risultati caso 6

Caso 5

Si evidenziano le date di inizio e di fine analisi:

SET @d1='2009-01-11 01:00:00';

SET @d2='2009-08-19 10:00:00';



	deviceID	LLSignal	QTYmisure
1	59	7001	2
2	59	7000	1
3	59	7001	1
4	61	7000	1
5	61	7001	2
6	62	7000	1
7	62	7001	1

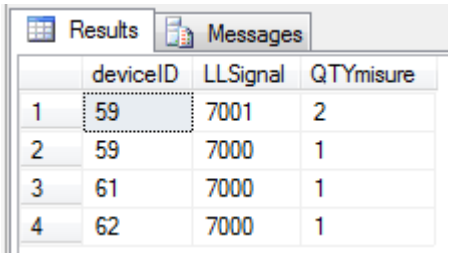
Figura 4.24: risultati caso 5

Caso 4

Si evidenziano le date di inizio e di fine analisi:

SET @d1='2009-01-11 01:00:00';

SET @d2='2009-06-20 10:00:00';



	deviceID	LLSignal	QTYmisure
1	59	7001	2
2	59	7000	1
3	61	7000	1
4	62	7000	1

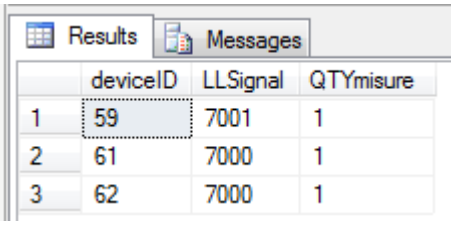
Figura 4.25: risultati caso 4

Caso 3

Si evidenziano le date di inizio e di fine analisi:

SET @d1='2009-01-11 01:00:00';

SET @d2='2009-02-28 10:00:00';



	deviceID	LLSignal	QTYmisure
1	59	7001	1
2	61	7000	1
3	62	7000	1

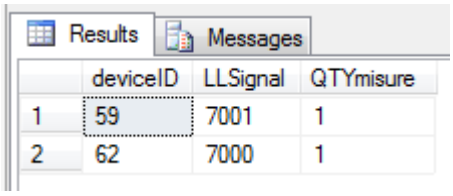
Figura 4.26: risultati caso 3

Caso 2

Si evidenziano le date di inizio e di fine analisi:

SET @d1='2009-01-11 01:00:00';

SET @d2='2009-02-23 10:00:00';



	deviceID	LLSignal	QTYmisure
1	59	7001	1
2	62	7000	1

Figura 4.27: risultati caso 2

Caso 1

Si evidenziano le date di inizio e di fine analisi:

SET @d1='2009-01-11 01:00:00';

SET @d2='2009-01-12 01:00:00';

deviceID	LLSignal	QTYmisure

Figura 4.28: risultati caso 1

Rimane, infine, da analizzare il caso di individuazione delle misure tra le date di inizio e di fine analisi specificate in colore arancione nella figura 4.14 di pag. 216. Si evidenziano le date di inizio e di fine analisi:

SET @d1='2009-08-13 14:00:00';

SET @d2='2009-09-30 21:00:00';

Results | Messages

	deviceID	LLSignal	QTYmisure
1	59	7001	1
2	61	7001	3
3	62	7000	1

Figura 4.29: risultati

Per mostrare i risultati con maggiore chiarezza, si enumerano i record attesi con 1-2-3-4-5, come mostrato in figura 4.30:

	CommID	LLSignalID	AcqTime	ValueI
▶	143	7001	22/02/2009 04:...	50
	162	7001	01/03/2009 14:...	35
	152	7000	18/06/2009 23:...	23
	153	1	15/08/2009 10:...	34
	154	7000	16/11/2009 11:...	46
	155	7000	27/02/2009 15:...	47
	156	3	18/08/2009 19:...	45
	157	4	25/09/2009 09:...	34
	158	7001	14/11/2009 05:...	57
	159	7000	19/01/2009 12:...	46
	160	7001	09/07/2009 14:...	23
	161	5	30/09/2009 20:...	47
	163	2	14/08/2009 07:...	43
*	NULL	NULL	NULL	NULL

Figura 4.30: risultati attesi

Come si può osservare in figura 4.31 la query conferma i risultati attesi in figura 4.31:

Results | Messages

	deviceID	LLSignal	QTYmisure
1	59	1	1
2	61	2	1
3	61	3	1
4	61	4	1
5	62	5	1

Figura 4.31: risultati finali

Results | Messages

	deviceID	LLSignalID	AcqTime
1	59	1	2009-08-15 10:00:00.000
2	61	2	2009-08-14 07:22:45.000
3	61	3	2009-08-18 19:00:32.000
4	61	4	2009-09-25 09:42:35.000
5	62	5	2009-09-30 20:25:40.000

Figura 4.32: risultati finali con AcqTime

Script SQL:

```
use ECENTRAL_PMASCHI;

DECLARE @d1 as datetime;
SET @d1='2009-08-13 14:00:00';
DECLARE @d2 as datetime;
SET @d2='2009-09-30 21:00:00';
DECLARE @COUNT as int;
SET @COUNT=0;
DECLARE @DID_Start as int;
DECLARE @EffectiveSince_Start as datetime;
DECLARE @DID_End as int;
DECLARE @EffectiveSince_End as datetime;
DECLARE @EffectiveSince_End_Supporto as datetime;
DECLARE @dID_Attuale as int
DECLARE @FETCH_STATUS_RICERCA AS INT
DECLARE @FETCH_STATUS_SINGOLO AS INT
DECLARE @FETCH_STATUS_CStart AS INT
DECLARE @statoC_End AS INT
DECLARE @correctStart AS datetime
DECLARE @GID as int;
SET @GID = 33;

-- #TempTable contiene l'esplorazione del gruppo dato ricorsivamente fino alle foglie
create table #TempTable (memberGID int);
-- #TempTableView serve per visualizzare le misure
create table #TempTableView (deviceID int, LLSignal int, QTYmisure int);

-- esplorazione del gruppo dato (hp: staticità)
WITH tree(GID, MemberGID, GroupLevel, EvaluateRecursively) AS
(
	select distinct GMS.GID, MemberGID, 0 AS GroupLevel, GMS.EvaluateRecursively from dbo.GroupMShipG as GMS, dbo.GroupMShipD as
Impianti -- ANCHOR MEMBER
	where (GMS.GID = @GID)
	UNION ALL
	select G_arrivo.GID, G_arrivo.MemberGID, GroupLevel + 1, G_arrivo.EvaluateRecursively from dbo.GroupMShipG as G_arrivo --
RECURSIVE MEMBER
	INNER JOIN tree t on G_arrivo.GID = t.MemberGID
	where t.EvaluateRecursively = 1 -- per prendere i gruppi collegati a quello precedente
)
insert #TempTable (memberGID)
select t.MemberGID
from tree as t
order by t.MemberGID


-- popolata la tabella con i gruppi, si inseriscono nel cursore impianti solo i gruppi di device e non i gruppi di -- gruppi
DECLARE Impianti CURSOR
FOR
	select distinct GSMD.GID
	from #TempTable AS TT JOIN dbo.GroupMShipD_Visio AS GSMD on GSMD.GID=TT.MemberGID

OPEN Impianti
FETCH NEXT FROM Impianti INTO @GID -- si legge il primo record di impianti
-- Per ogni impianto del gruppo dato si vanno a prendere le misure dell'intervallo in esame
WHILE (@@FETCH_STATUS = 0)
BEGIN
	PRINT '@GID: ' + RTRIM(CAST(@GID AS NVARCHAR(30)))
	DECLARE C_Start CURSOR
	FOR
		SELECT GMSD.memberDID, GMSD.EffectiveSince
		FROM dbo.GroupMShipD_Visio AS GMSD
		WHERE GMSD.GID = @GID
		AND IsRevoked = 0 -- inizio legame device con impianto
		ORDER BY GMSD.memberDID, GMSD.EffectiveSince

	DECLARE C_End CURSOR
	FOR
		SELECT GMSD.memberDID, GMSD.EffectiveSince
		FROM dbo.GroupMShipD_Visio AS GMSD
		WHERE GMSD.GID = @GID
		AND IsRevoked = 1 -- fine legame device con impianto
		ORDER BY GMSD.memberDID, GMSD.EffectiveSince

	OPEN C_Start
	FETCH NEXT FROM C_Start INTO @DID_Start, @EffectiveSince_Start -- si legge il primo record di C_start
	SET @FETCH_STATUS_CStart = @@FETCH_STATUS
	WHILE (@FETCH_STATUS_CStart = 0) -- per resettare il conteggio, quando si cambia device
	BEGIN
		PRINT 'Si analizza DID_Start: ' + RTRIM(CAST(@DID_Start AS NVARCHAR(30)))
```

```
		PRINT 'con -> EFF_SINCE_Start: ' + RTRIM(CAST(@EffectiveSince_Start AS NVARCHAR(30)))
		SET @COUNT=1
		SET @DID_Attuale=@DID_Start
		OPEN C_End -- si apre il cursore C_End
		FETCH NEXT FROM C_End INTO @DID_End, @EffectiveSince_End -- si prende il primo elemento del
-- cursore End
		SET @FETCH_STATUS_SINGOLO = @@FETCH_STATUS
			PRINT 'll record corrispondente di chiusura è -> DID_End: ' + RTRIM(CAST(@DID_End AS NVARCHAR(30)))
			PRINT 'con -> EFF_SINCE_End: ' + RTRIM(CAST(@EffectiveSince_End AS NVARCHAR(30)))
		SET @FETCH_STATUS_RICERCA = 0
			WHILE (((@FETCH_STATUS_RICERCA = 0) AND (@DID_Attuale <> @DID_End)) -- se non si è
-- allineati si individua lo stesso DID del cursore C_End da cui partire
			BEGIN
				FETCH NEXT FROM C_End INTO @DID_End, @EffectiveSince_End
				SET @FETCH_STATUS_RICERCA = @@FETCH_STATUS
				PRINT 'Eseguita ricerca, allineato con: ' + RTRIM(CAST(@DID_End AS NVARCHAR(30)))
			END
		WHILE ((@FETCH_STATUS_CStart = 0) AND (@DID_Attuale=@DID_Start)) -- quando cambia il DID si esce -- fuori da questo while e
si fa la ricerca esterna
		BEGIN
			IF ((@FETCH_STATUS_SINGOLO <> 0) OR (@DID_Start <> @DID_End)) -- ((the row of 2° cursor is missing) OR (d2 is not the
same device of d1)) se l'intervallo è aperto, perchè il Tend è dopo il d2 (da query) il record corrispondente in C_End non esiste. Può anche accadere che
sia cambiato il DID. Cosa si fa? si deve chiudere l'intervallo con d2
			BEGIN
				PRINT 'Trovato intervallo aperto del device in esame: ' + + RTRIM(CAST(@DID_Start AS NVARCHAR(30)))
				if (@EffectiveSince_Start < @d1)
				begin
					PRINT '@effectiveSince_Start < @d1, quindi si chiude con quello maggiore: @d1'
					set @correctStart = @d1
				end
				else
				begin
					PRINT '@effectiveSince_Start > @d1, quindi si chiude con quello maggiore: @EffectiveSince_Start'
					set @correctStart = @EffectiveSince_Start
				end
				INSERT #TempTableView -- oppure ##Mia tabella se la Store Procedure viene invocata dall'esterno
				SELECT @DID_Start, LLSignalID, count(*)
				FROM dbo.Communication as C inner join CommunicationData_Visio as CD on CD.CommID = C.CommID
				WHERE C.DID = @DID_Start
				AND CD.STID = 1 -- per prendere le misure
				AND CD.AcqTime BETWEEN @correctStart AND @d2
				group by LLSignalID
			END
			IF (((@COUNT = 1) AND (@EffectiveSince_Start > @EffectiveSince_End) AND (@DID_Start = @DID_End) and
(@EffectiveSince_End > @d1)) -- l'inizio dell'intervallo è dopo la fine di un intervallo precedente
			BEGIN
				PRINT 'ENTRATO IN SECONDO IF: effectiveSince del record del cursore C_Start è maggiore dell effectiveSince del record del
cursore C_End di pari device'
				-- da fare di sicuro
				BEGIN
					PRINT 'Chiudo record del cursore C_end con @d1'
					INSERT #TempTableView -- oppure ##Mia tabella se la Store Procedure viene invocata dall'esterno
					SELECT @DID_Start, LLSignalID, count(*)
					FROM dbo.Communication as C inner join CommunicationData_Visio_Visio as CD on CD.CommID = C.CommID
					WHERE C.DID = @DID_Start
					AND CD.STID = 1 -- per prendere le misure
					AND CD.AcqTime BETWEEN @d1 AND @EffectiveSince_End -- si chiude l'intervallo finito con la data di inizio query e
l'EffectiveSince del record di C_End
					group by LLSignalID
				END
				FETCH NEXT FROM C_End INTO @DID_End, @EffectiveSince_End_Supporto -- si legge il prossimo EffectiveSinceEnd per
non agganciare l'EffectiveSinceStart che verrebbe saltato, perchè l'EffectiveSinceEnd in esame viene chiuso con @d1
				-- se c'è il record corrispondente in C_End (@@FETCH_STATUS = 0) ed è riferito allo stesso device di C_start si chiude
l'intervallo con gli EffectiveSinceStart ed EffectiveSinceStart
				IF ((@@FETCH_STATUS = 0) and (@DID_End = @DID_Start))
				BEGIN
					PRINT 'Record adiacente di C_End: c è il record nel cursore C_end di pari device, che quindi si chiude con
EffectiveSince_Start'
					INSERT #TempTableView -- oppure ##Mia tabella se la Store Procedure viene invocata dall'esterno
					SELECT @DID_Start, LLSignalID, count(*)
					FROM dbo.Communication as C inner join CommunicationData_Visio_Visio as CD on CD.CommID = C.CommID
					WHERE C.DID = @DID_Start
					AND CD.STID = 1 -- per prendere le misure
					AND	CD.AcqTime BETWEEN @EffectiveSince_Start AND @EffectiveSince_End_Supporto -- dato dal 2° fetch
consecutivo sul cursore 2
					group by LLSignalID
				END
				-- se c'è un record nel cursore di chiusura intervallo, ma non corrisponde allo stesso device di quello di partenza,
				-- oppure il record del cursore di chiusura intervallo non c'è, bisogna chiudere l'intervallo con @d2
				if ((((@@FETCH_STATUS = 0) and (@DID_End <> @DID_Start)) or (@@FETCH_STATUS <> 0))
				BEGIN
					PRINT 'Record adiacente di C_End: Non C è il record nel cursore C_end di pari device, oppure c è ma è di un altro device, si
chiude con @d2'
```

```
                    INSERT #TempTableView -- oppure ##Mia tabella se la Store Procedure viene invocata dall'esterno
                    SELECT @DID_Start, LLSignalID, count(*)
                    FROM dbo.Communication as C inner join CommunicationData_Visio as CD on CD.CommID = C.CommID
                    WHERE C.DID = @DID_Start
                    AND CD.STID = 1 -- per prendere le misure
                    AND      CD.AcqTime BETWEEN @EffectiveSince_Start AND @d2 -- dato dal 2° fetch consecutivo sul cursore 2
                    group by LLSignalID
                END
            END
            IF (((@COUNT >= 1) AND (@EffectiveSince_End > @EffectiveSince_Start) AND (@DID_Start = @DID_End)) -- caso normale
(intervallo chiuso) e si controlla sempre il match tra device per essere sicuri di riferirsi sempre allo stesso
            BEGIN
                PRINT 'Trovato intervallo chiuso cioè i due record si riferiscono allo stesso device. EffectiveSinceStart (record cursore C_start) e
EffectiveSinceEnd (record cursore C_End)'
                IF ((@EffectiveSince_Start <= @d1) and (@EffectiveSince_End > @d1)) -- se la data d'inizio @D1 è dentro l'intervallo
(effsinceStart to effsinceEnd) si prende da @d1
                BEGIN
                    IF (@d2 > @EffectiveSince_End)
                    BEGIN
                        PRINT 'Intervallo chiuso: lo start dell intervallo è prima di d1 richiesto e d2 richiesto è dopo la fine dell intervallo'
                        INSERT #TempTableView -- oppure ##Mia tabella se la Store Procedure viene invocata dall'esterno
                        SELECT @DID_Start, LLSignalID, count(*)
                        FROM dbo.Communication as C inner join CommunicationData_Visio as CD on CD.CommID = C.CommID
                        WHERE C.DID = @DID_Start
                        AND CD.STID = 1 -- per prendere le misure
                        AND CD.AcqTime BETWEEN @d1 AND @EffectiveSince_End -- fisso @d1
                        group by LLSignalID
                    END
                    ELSE -- (@d2 < @EffectiveSince_End))
                    BEGIN
                        PRINT 'Intervallo chiuso: lo start dell intervallo è prima di d1 richiesto e d2 richiesto è prima della fine dell intervallo'
                        INSERT #TempTableView -- oppure ##Mia tabella se la Store Procedure viene invocata dall'esterno
                        SELECT @DID_Start, LLSignalID, count(*)
                        FROM dbo.Communication as C inner join CommunicationData_Visio as CD on CD.CommID = C.CommID
                        WHERE C.DID = @DID_Start
                        AND CD.STID = 1 -- per prendere le misure
                        AND CD.AcqTime BETWEEN @d1 AND @d2 -- fisso @d1
                        group by LLSignalID
                    END
                END
                if ((@EffectiveSince_Start > @d1) and (@EffectiveSince_End > @d1))
                BEGIN
                    IF (@d2 > @EffectiveSince_End)
                    BEGIN
                        PRINT 'Intervallo chiuso: lo start dell intervallo è dopo di d1 richiesto e d2 richiesto è dopo la fine dell intervallo'
                        PRINT 'EFF_SINCE_End: ' + RTRIM(CAST(@EffectiveSince_End AS NVARCHAR(30))) + ' deve essere maggiore di
' + RTRIM(CAST(@d1 AS NVARCHAR(30)))
                        INSERT #TempTableView -- oppure ##Mia tabella se la Store Procedure viene invocata dall'esterno
                        SELECT @DID_Start, LLSignalID, count(*)
                        FROM dbo.Communication as C inner join CommunicationData_Visio as CD on CD.CommID = C.CommID
                        WHERE C.DID = @DID_Start
                        AND CD.STID = 1 -- per prendere le misure
                        AND CD.AcqTime BETWEEN @EffectiveSince_Start AND @EffectiveSince_End -- fisso @EffectiveSince_Start
                        group by LLSignalID
                    END
                    ELSE -- (@d2 < @EffectiveSince_End)
                    BEGIN
                        PRINT 'Intervallo chiuso: lo start dell intervallo è dopo di d1 richiesto e d2 richiesto è dopo la fine dell intervallo'
                        INSERT #TempTableView -- oppure ##Mia tabella se la Store Procedure viene invocata dall'esterno
                        SELECT @DID_Start, LLSignalID, count(*)
                        FROM dbo.Communication as C inner join CommunicationData_Visio as CD on CD.CommID = C.CommID
                        WHERE C.DID = @DID_Start
                        AND CD.STID = 1 -- per prendere le misure
                        AND CD.AcqTime BETWEEN @EffectiveSince_Start AND @D2 -- fisso @EffectiveSince_Start
                        group by LLSignalID
                    END
                END
            END
            SET @COUNT=@COUNT+1
            FETCH NEXT FROM C_Start INTO @DID_Start, @EffectiveSince_Start
            PRINT 'L attuale DID in DID_Attuale è: ' + RTRIM(CAST(@DID_Attuale AS NVARCHAR(30))) + ', mentre il prossimo DID in
DID_Start è: ' + RTRIM(CAST(@DID_Start AS NVARCHAR(30)))
            SET @FETCH_STATUS_CStart = @@FETCH_STATUS
            if (@DID_Attuale <> @DID_Start)
            BEGIN
                Close C_End -- si chiude il cursore C_End, perchè si deve uscire dal while interno
                set @statoC_End = @@CURSOR_ROWS
            END
            IF (@@CURSOR_ROWS <> 0) -- non è chiuso
            BEGIN
                FETCH NEXT FROM C_End INTO @DID_End, @EffectiveSince_End
                SET @FETCH_STATUS_SINGOLO = @@FETCH_STATUS
                PRINT 'FETCH STATUS SINGOLO è: ' + RTRIM(CAST(@FETCH_STATUS_SINGOLO AS NVARCHAR(30)))
```

```
                END
            END
        END
        CLOSE C_Start
        DEALLOCATE C_Start
        DEALLOCATE C_End
        FETCH NEXT FROM Impianti INTO @GID -- si legge il successivo record di Impianti
END
CLOSE Impianti
DEALLOCATE Impianti
select t.deviceID, CDV.LLSignalID, CDV.AcqTime from #TempTableView as t join dbo.CommunicationData_Visio as CDV on t.LLSignal =
CDV.LLSignalID
Order by deviceID, CDV.AcqTime
drop table #TempTable
drop table #TempTableView
```

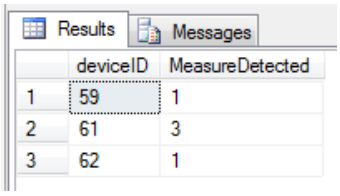
Se si volesse ricavare il numero di misure inviate da ogni device di un impianto nell'intervallo di tempo richiesto, si esegue la seguente interrogazione:

select deviceID, SUM(QTYmisure) as MeasuresDetected

from #Temptable

group by deviceID

drop table #Temptable



	deviceID	MeasureDetected
1	59	1
2	61	3
3	62	1

Figura 4.33: risultati con *group by*

4.2.6 Query n.6

Testo:

Ricavare quali sono i gruppi correnti che compongono un dato gruppo.

Per agevolare la comprensione sono state indicate le funzioni invocate con colori diversi.

Script SQL:

```
USE [ECENTRAL_PMASCHI]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER FUNCTION [dbo].[GetCur_GroupMShipGGID_Recursively](@pGID int, @pDate datetime)
RETURNS TABLE
AS
RETURN
(
    WITH tree(GID, EffectiveSince, GroupLevel) AS
    (
        select I.MemberGID, I.EffectiveSince, 1 from dbo.GetCur_GroupMShipGGID(@pGID, @pDate) as I
        UNION ALL
        select II.MemberGID, II.EffectiveSince, GroupLevel + 1 from dbo.GetCur_GroupMShipG(@pDate) as II
        join tree on tree.GID = II.GID
    )
    select * from tree
)
```

```
USE [ECENTRAL_PMASCHI]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER FUNCTION [dbo].[GetCur_GroupMShipGGID](@pGID int, @pDate datetime)
RETURNS TABLE
AS
RETURN
(
    SELECT *
    FROM [GroupMShipG] G2G
    WHERE G2G.EffectiveSince <= @pDate
    AND G2G.GMSGID = (SELECT TOP 1 I.GMSGID
                FROM [GroupMShipG] I
                WHERE I.GID = G2G.GID
                AND I.MemberGID = G2G.MemberGID
                AND I.EffectiveSince <= @pDate
                ORDER BY I.EffectiveSince DESC, I.IsRevoked ASC
            )
    AND G2G.GID = @pGID
)

USE [ECENTRAL_PMASCHI]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER FUNCTION [dbo].[GetCur_GroupMShipGI](@pDate datetime)
RETURNS TABLE
AS
RETURN
(
    SELECT *
    FROM [GroupMShipG] G2G
    WHERE G2G.EffectiveSince <= @pDate
    AND G2G.GMSGID = (SELECT TOP 1 I.GMSGID
                FROM [GroupMShipG] I
                WHERE I.GID = G2G.GID
                AND I.MemberGID = G2G.MemberGID
                AND I.EffectiveSince <= @pDate
                ORDER BY I.EffectiveSince DESC, I.IsRevoked ASC
                )
)
```

Input di esempio:

set @pGID = 6;

set @pDate = '2009-07-04 09:19:15';

Risultati:

Results | Messages

	GID	EffectiveSince	GroupLevel
1	1	2009-04-07 09:19:24.560	1
2	2	2009-04-07 09:19:15.263	1
3	3	2009-04-07 09:26:59.577	2
4	7	2009-04-07 09:27:02.547	2
5	8	2009-04-07 09:31:53.200	2
6	9	2009-04-07 09:31:56.950	2

Figura 4.34: risultati query n.6

4.2.7 Query n.7

Testo:

Recuperare gli impianti attivi di un dato gruppo ad una certa data.

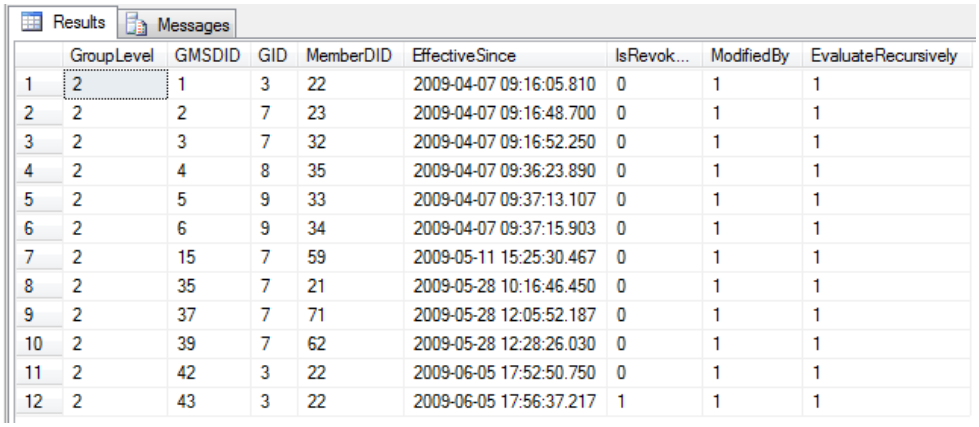
Per agevolare la comprensione sono state indicate le funzioni invocate con colori diversi.

Script SQL:

```
USE [ECENTRAL_PMASCHI]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER FUNCTION [dbo].[GetCur_GroupMShipDGID_Recursively](@pGID int, @pDate datetime)
RETURNS TABLE
AS
RETURN
(
    WITH tree(GID, EffectiveSince, GroupLevel) AS
    (
        select I.MemberGID, I.EffectiveSince, 1 from dbo.GetCur_GroupMShipGGID(@pGID, @pDate)
as I
        UNION ALL
        select II.MemberGID, II.EffectiveSince, GroupLevel + 1 from
dbo.GetCur_GroupMShipG(@pDate) as II
        join tree on tree.GID = II.GID
    )
    select tree.GroupLevel as GroupLevel, GD.*
    from tree join [GroupMShipD] as GD on GD.GID = tree.GID
    UNION
    select 0, D.* from dbo.GetCur_GroupMShipDGID(@pGID, @pDate) as D
)


ALTER FUNCTION [dbo].[GetCur_GroupMShipDGID](@pGID int, @pDate datetime)
RETURNS TABLE
AS
RETURN
(
    SELECT *
        FROM [GroupMShipD] O
        WHERE O.EffectiveSince <= @pDate
        AND O.GMSDID = (SELECT TOP 1 I.GMSDID
                FROM [GroupMShipD] I
                WHERE I.MemberDID = O.MemberDID
                AND I.GID = O.GID
                AND I.EffectiveSince <= @pDate
                ORDER BY I.EffectiveSince DESC, I.IsRevoked ASC
        )
        AND O.GID = @pGID
)
```

Risultati:



	GroupLevel	GMSDID	GID	MemberDID	EffectiveSince	IsRevok...	ModifiedBy	EvaluateRecursively
1	2	1	3	22	2009-04-07 09:16:05.810	0	1	1
2	2	2	7	23	2009-04-07 09:16:48.700	0	1	1
3	2	3	7	32	2009-04-07 09:16:52.250	0	1	1
4	2	4	8	35	2009-04-07 09:36:23.890	0	1	1
5	2	5	9	33	2009-04-07 09:37:13.107	0	1	1
6	2	6	9	34	2009-04-07 09:37:15.903	0	1	1
7	2	15	7	59	2009-05-11 15:25:30.467	0	1	1
8	2	35	7	21	2009-05-28 10:16:46.450	0	1	1
9	2	37	7	71	2009-05-28 12:05:52.187	0	1	1
10	2	39	7	62	2009-05-28 12:28:26.030	0	1	1
11	2	42	3	22	2009-06-05 17:52:50.750	0	1	1
12	2	43	3	22	2009-06-05 17:56:37.217	1	1	1

Figura 4.35: risultati query n.7

4.2.8 Query n.8

Testo:

Individuare tutti gli HighLevelSignal correnti (ultimi) di un dato device.

Script SQL:

```
USE [ECENTRAL_PMASCHI]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE FUNCTION [GetLastDeviceHighLevelData](@pDID int, @pDate datetime)
RETURNS TABLE
AS
RETURN
(
    WITH HighLevelData(HLSignalID, EffectiveOn, ValueI, ValueF, ValueD, ValueB, ValueT, EDID,
LLSignalID, CommID) AS
    (
        SELECT HLSignalID, EffectiveOn, ValueI, ValueF, ValueD, ValueB, ValueT, EDID, NULL,
NULL
        FROM ElaboratedData
        WHERE DID = @pDID AND EffectiveOn <= @pDate

        UNION ALL

        SELECT LL.HLSignalID, CD.DataTime AS EffectiveOn, CD.ValueI, CD.ValueF, CD.ValueD,
CD.ValueB, CD.ValueT, NULL, CD.LLSignalID, CD.CommID
        FROM CommunicationData CD
        INNER JOIN LowLevelSignal LL ON CD.LLSignalID = LL.LLSignalID
        INNER JOIN Communication C ON CD.CommID = C.CommID
        WHERE C.DID = @pDID AND CD.DataTime <= @pDate
    )
    SELECT H.*, HS.SDTID
    FROM HighLevelData H
```

```
    INNER JOIN (SELECT HLSignalID, MAX(EffectiveOn) EO
                FROM HighLevelData
                GROUP BY HLSignalID) M ON H.HLSignalID = M.HLSignalID AND H.EffectiveOn =
M.EO
    INNER JOIN HighLevelSignal HS ON H.HLSignalID = HS.HLSignalID
)
```

Con HLSignal che derivano da comunicazioni:

set @pDID = 21;

set @pDate = '2009-09-04 09:19:15';

Risultati:

Results | Messages

	HLSignalID	EffectiveOn	ValueI	ValueF	ValueD	ValueB	ValueT	EDID	LLSignalID	CommID	SDTID
26	1001	2009-04-17 00:00:00.000	50	NULL	NULL	NULL	NULL	NULL	1001	30	1
27	1001	2009-04-17 00:00:00.000	50	NULL	NULL	NULL	NULL	NULL	1001	31	1
28	1001	2009-04-17 00:00:00.000	50	NULL	NULL	NULL	NULL	NULL	1001	32	1
29	1001	2009-04-17 00:00:00.000	50	NULL	NULL	NULL	NULL	NULL	1001	33	1
30	1002	2009-04-18 00:00:00.000	NULL	NULL	NULL	NULL	pippo	NULL	1002	33	2
31	1003	2009-04-19 00:00:00.000	NULL	32,75	NULL	NULL	NULL	NULL	1003	33	3
32	1004	2009-04-20 00:00:00.000	NULL	NULL	NULL	1	NULL	NULL	1004	33	4
33	1005	2009-04-21 00:00:00.000	NULL	NULL	1976...	NULL	NULL	NULL	1005	33	5
34	1001	2009-04-17 00:00:00.000	50	NULL	NULL	NULL	NULL	NULL	1001	34	1
35	1001	2009-04-17 00:00:00.000	50	NULL	NULL	NULL	NULL	NULL	1001	35	1
36	1001	2009-04-17 00:00:00.000	50	NULL	NULL	NULL	NULL	NULL	1001	36	1
37	1001	2009-04-17 00:00:00.000	20	NULL	NULL	NULL	NULL	NULL	1001	37	1

Figura 4.36: risultati query n.8 parte prima

Con HLSignal che derivano da elaborazioni e quindi non mappate in alcuna comunicazione.

set @pDID = 61;

set @pDate = '2009-09-04 09:19:15';

Risultati:

Results | Messages

	HLSignalID	EffectiveOn	ValueI	ValueF	ValueD	ValueB	ValueT	ED...	LLSignalID	CommID	SDTID
1	502	2009-05-03 00:00:00.000	NULL	NULL	NULL	1	NULL	61	NULL	NULL	4

Figura 4.37: risultati query n.8 parte seconda

4.2.9 Query n.9

Testo:

Individuare il primo GSMEndPoint di un dato device o lato eCentral

Script SQL:

```
USE [ECENTRAL_PMASCHI]
GO
```

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE FUNCTION [dbo].[GetDeviceFirstGSMEndpoint]
(
    @pLocalToServer bit
)
RETURNS
@retTable TABLE
(
    DID int,
    GSMEPID int,
    SIMNumber nvarchar(20),
    IsEnabled bit,
    ModifiedOn datetime,
    ModifiedBy int
)
AS
BEGIN
    insert into @retTable (DID,GSMEPID,SIMNumber,IsEnabled,ModifiedOn,ModifiedBy)
    select top 1 A.DID, B.GSMEPID, B.SIMNumber, B.IsEnabled, B.ModifiedOn, B.ModifiedBy
    from Device2GSMEndpoint A join GSMEndpoint B on A.GSMEPID = B.GSMEPID
    where A.IsLocalToServer = @pLocalToServer and B.IsEnabled = 1 and A.IsEnabled = 1
    group by A.DID, B.GSMEPID, B.SIMNumber, B.IsEnabled, B.ModifiedOn, B.ModifiedBy

    RETURN
END
```

Risultati lato eCentral:

Results | Messages

	DID	GSMEPID	SIMNumber	IsEnabled	ModifiedOn	ModifiedBy
1	21	1	+393346839971	1	2009-04-22 15:37:24.160	1

Figura 4.38: risultati lato eCentral

Risultati lato device:

Results | Messages

	DID	GSMEPID	SIMNumber	IsEnabled	ModifiedOn	ModifiedBy
1	21	4	2222222222	1	2009-04-07 11:35:33.420	1

Figura 4.39: risultati lato device

4.2.10 Query n.10

Testo:

Recuperare le RepositoryProperty (con descrizione localizzata) di un QualifiableObject

Script SQL:

```
USE [ECENTRAL_PMASCHI]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE FUNCTION [dbo].[GetGeneralProperty]
(
    @pGPID int,
    @pQOID int,
    @pCUID int
)
RETURNS TABLE
AS RETURN
(
    select PropertyType.FormatString, Property2QO.IsSystem, Property2QO.IsEnabled,
Property2QO.IsMandatory,
    Property2QO.MaxValuesN, Property2QO.RelevanceFactor, Property2QO.CanBeModified,
   Property2QO.DefaultGPAVCode, Property2QO.ManageSilently, Property2QO.ShowLongDescr,
   G.*, Property2QO.IsClassifiedProperty,  Property2QO.ShowInWidget,
Property2QO.ShowInWidgetDetail
    from GetI18NGeneralProperty(@pCUID) G join PropertyType on G.PTID = PropertyType.PTID
    join Property2QO on G.GPID = Property2QO.GPID
    where Property2QO.QOID = @pQOID and G.GPID = @pGPID
)
```

Per l'esempio sono stati specificati:

- QualifiableObject: Device
- Lingua (Culture): italiano
- RepositoryPropertyID: 14 (corrisponde alla RepositoryProperty "Device mobile")

ovvero:

set @pCUID = 2; -- italiano

set @pQOID = 3; -- device

set @pGPID = 14; -- device

Risultati:

Results | Messages

	y	ModifiedOn	LDAPAttName	IsSortable	IsFilterable	OverrideFormatString	IntegrationTag	IALID	ShortDescr	LongDescr
1		2009-04-17 15:33:50.623	NULL	0	0	NULL	NULL	237	Device mobile	Indica se il device è un device mobile utilizzato per la manutenzione

Figura 4.40: risultati query n.10

4.2.11 Query n.11

Testo:

Determinare le attività compiute su di un device in un certo intervallo di tempo.

Poiché l'Activity è QualifiableObject, la sua descrizione è una RepositoryProperty per cui occorre effettuare il join tra Activity e Qualifiable2Property su ACTID per conoscere i GPID posseduti e tra questi prendere la GPID corrispondente alla descrizione. La valorizzazione della descrizione è contenuta in ActPropValue, più precisamente nell'attributo ValueT. Qualora ci siano più descrizione, perché dello

stesso tipo, ma con stringhe di lunghezza diversa (es. cambiate pile, sostituite pile da 3 V con pile da 1.5 V perché è cambiato…) diversi avrò più istanceN della RepositoryProperty “Descrizione”.

Script SQL:

```
USE [ECENTRAL_PMASCHI]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE FUNCTION [dbo].[GetDeviceActivities](@pDID int, @pStartTime datetime, @pEndTime
datetime, @pGPID int, @pGPAVCode int)
RETURNS TABLE
AS RETURN
(
    SELECT DISTINCT A2O.ACTID
    FROM Activity A
    JOIN Activity2Object A2O on A.ACTID = A2O.ACTID
    JOIN GetCur_ACTPropertyValue(GETDATE(), 1) ACTP ON ACTP.ACTID = A.ACTID
    AND ACTP.GPID = @pGPID and ACTP.IsDeleted = 0
    AND ACTP.ValueA = @pGPAVCode
    WHERE A2O.DID = @pDID AND  A2O.EffectiveSince BETWEEN @pStartTime AND
@pEndTime
)

declare @pGPID AS int;
set @pGPID = 4;
declare @pDID AS int;
set @pDID = 65;
declare @pStartTime AS datetime;
set @pStartTime = '2007-01-01 09:19:15';
declare @pEndTime AS datetime;
set @pEndTime = '2009-09-04 09:19:15';
declare @pGPAVCode AS int;
set @pGPAVCode = 1;
```

Risultati:

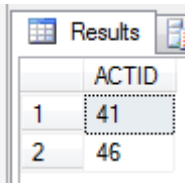


Figura 4.41: risultati query n.11

4.2.12 Query n.12

Testo:

Determinare le attività compiute su di un gruppo in un certo intervallo di tempo.

Script SQL:

```
USE [ECENTRAL_PMASCHI]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE FUNCTION [dbo].[GetGroupActivities1](@pGID int, @pStartTime datetime, @pEndTime
datetime, @pGPID int, @pGPAVCode int, @pRecurvive bit)
RETURNS @retTable TABLE ( ACTID int)
AS
BEGIN
        IF ((SELECT COUNT(GID) FROM GetCur_GroupMShipGGID_Recursively(@pGID,
GETDATE()) WHERE  GID  = @pGID) > 0 AND (@pRecurvive = 1))
            INSERT INTO @retTable(ACTID)
            SELECT  A2O.ACTID
            FROM Activity A
            JOIN Activity2Object A2O on A.ACTID = A2O.ACTID
            JOIN GetCur_ACTPropertyValue(GETDATE(), 1) ACTP ON ACTP.ACTID = A.ACTID
AND ACTP.GPID = @pGPID AND ACTP.IsDeleted = 0
            AND ACTP.ValueA = @pGPAVCode
            WHERE A2O.EffectiveSince BETWEEN @pStartTime AND @pEndTime
                AND A2O.GID IN ( SELECT GID FROM
GetCur_GroupMShipGGID_Recursively(@pGID, GETDATE())) OR  A2O.GID  = @pGID

        ELSE
            INSERT INTO @retTable(ACTID)
            SELECT  A2O.ACTID
            FROM Activity A
            JOIN Activity2Object A2O on A.ACTID = A2O.ACTID
            JOIN GetCur_ACTPropertyValue(GETDATE(), 1) ACTP ON ACTP.ACTID = A.ACTID
AND ACTP.GPID = @pGPID AND ACTP.IsDeleted = 0
            AND ACTP.ValueA = @pGPAVCode
            WHERE A2O.EffectiveSince BETWEEN @pStartTime AND @pEndTime
                AND A2O.GID  = @pGID

        IF((SELECT  COUNT(MemberDID) FROM  GetCur_GroupMShipDGID(@pGID,
GETDATE())) > 0)
            INSERT INTO @retTable(ACTID)
            SELECT DISTINCT A2O.ACTID
            FROM Activity A
            JOIN Activity2Object A2O on A.ACTID = A2O.ACTID
            JOIN GetCur_ACTPropertyValue(GETDATE(), 1) ACTP ON ACTP.ACTID = A.ACTID
AND ACTP.GPID = @pGPID AND ACTP.IsDeleted = 0
            AND ACTP.ValueA = @pGPAVCode
            WHERE A2O.EffectiveSince BETWEEN @pStartTime AND @pEndTime
                AND A2O.DID IN ( SELECT MemberDID FROM
dbo.GetCur_GroupMShipDGID(@pGID, GETDATE()))



    RETURN
END

declare @pGPID as int;
set @pGPID = 4;
declare @pGID as int;
set @pGID = 30;
declare @pGPAVCODE as int;
```

```
set @pGPAVCODE = 1;
declare @pRecurvive as bit;
set @pRecurvive = 0;
declare @pStartTime as datetime;
set @pStartTime = '2007-1-1 1:00:01';
declare @pEndTime as datetime;
set @pEndTime = '2009-12-1 1:00:01';
```

Risultati:

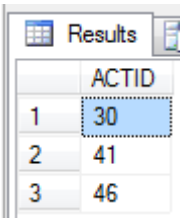


Figura 4.42: risultati query n.12

CONCLUSIONI

In questa tesi è stato affrontato il ciclo di sviluppo di una sottoparte del sistema eCentral relativa all'ambito dell'odorizzazione. L'architettura (moduli GSMChannelManager e EdorDriver, modello dati, etc.) realizzata durante il tirocinio, permette un'efficiente ed efficace gestione dei device. I moduli possono essere utilizzati per il canale di comunicazione GSM nell'ambito dell'odorizzazione, come già più volte indicato; gli sviluppi futuri potranno interessare la gestione di altre tipologie di canali di comunicazione e di altri ambiti, come la protezione catodica.

Nella fase di progettazione del modello di comunicazione si è preferito scomporre il problema in sottoproblemi, evitando le difficoltà connesse con la necessità di dover raggiungere una soluzione immediata dell'intero problema, come nel caso della creazione del workflow di identificazione del Driver Edor.

La fase di analisi è stata entusiasmante per l'interazione diretta con il committente.

Nella fase di progettazione sono stati rispettati con successo, superando non poche difficoltà, i seguenti criteri e principi di progettazione:

- scomponibilità: possibilità di scomporre il progetto in parti;
- componibilità: possibilità di riutilizzare in modo efficace le parti in cui è stato suddiviso il progetto;
- comprensibilità: possibilità di spiegare una singola parte senza dover spiegare l'intero progetto;
- proporzionalità: possibilità di coinvolgere un numero di moduli proporzionale alle modifiche apportate al progetto;
- isolamento: possibilità di circoscrivere ad un numero limitato di moduli la propagazione dell'errore verificatosi in uno di essi;
- alta coesione dei moduli;
- basso accoppiamento (poche interfacce): possibilità di comunicazione di ogni modulo con pochi altri;
- interfacce piccole: possibilità di scambiare la minor quantità possibile di informazione, quando non si possono eliminare le interfacce;
- interfacce esplicite: possibilità di una collaborazione evidente tra moduli;
- information hiding: possibilità di nascondere le informazione sotto uno strato di funzioni, che sono le uniche abilitate ad accedervi.

È stato individuato un modello di gestione più efficiente della storicizzazione, che verrà implementato prossimamente, mentre di alcune query, che fanno uso di cursori, si prevede una futura ottimizzazione.

I componenti software risultato delle attività svolte durante il tirocinio verranno integrati nel sistema software eCentral, tuttora in fase di sviluppo.

APPENDICE

In tale appendice vengono mostrati gli schemi ER: ER1 e ER2.

BIBLIOGRAFIA

1. Leszek A. Maciaszek, “Sviluppo di sistemi informativi con UML: Analisi dei requisiti e progetto di sistema”, Addison-Wesley, 2002.
2. Luca Cabibbo, Craig Larman, “APPLICARE UML: Analisi e progettazione orientata agli oggetti”, Terza Edizione, Pearson Prentice Hall, 2005.
3. Lucidi del corso di Progettazione del Software, A.A. 2007-2008, prof. Flavio Bonfatti, docente presso l’Università degli studi di Modena e Reggio Emilia.
4. Stefano Ceri, Piero Fraternali, Aldo Bongio, Marco Brambilla, Sara Comai, Maristella Matera, “Progettazione di dati e applicazioni per il Web”, McGraw-Hill, 2003.
5. Andrew S. Tanenbaum, “Reti di calcolatori”, Quarta Edizione, Pearson Prentice Hall, settembre 2005.
6. Lucidi del corso di “Sicurezza dei Sistemi in Rete”, A.A. 207-2008, prof. Michele Colajanni, Università di Modena e Reggio Emilia.
7. http://msdn.microsoft.com/it–it/library/cc185071.aspx
8. http://blog.teammatelabs.com/category/IIS.aspx
9. http://learn.iis.net/page.aspx/101/introduction–to–iis7–architecture/
10. http://learn.iis.net/page.aspx/128/iis-7-configuration-reference/
11. http://learn.iis.net/page.aspx/156/understanding-iis-70-configuration-delegation/
12. http://www.asp101.com/articles/sample_chapters/sitepoint_byoaspnet20/chapter1.asp
13. http://www.asp101.com/articles/sample_chapters/sitepoint_byoaspnet20/chapter2.asp
14. http://blogs.msdn.com/tmarq/archive/2007/08/30/iis–7–0–asp–net–pipelines–modules–handlers–and–preconditions.aspx
15. http://www.west–wind.com/presentations/howaspnetworks/howaspnetworks.asp
16. http://learn.iis.net/page.aspx/244/how–to–take–advantage–of–the–iis7–integrated–pipeline/
17. http://msdn.microsoft.com/it–it/library/bb515251.aspx
18. http://technet.microsoft.com/it–it/magazine/2008.07.iis7.aspx
19. http://msdn.microsoft.com/it–it/library/ms178685(VS.80).aspx
20. http://blogs.devleap.com/rob/archive/2005/11/11/6203.aspx
21. http://msdn.microsoft.com/en–us/library/7d6sws33(VS.80).aspx
22. Matthew MacDonald e Mario Szpuszta, “Pro ASP.NET 3.5 in C# 2008”, Second Edition, apress, 2007.
23. Lucidi del corso di Tecnologia delle Basi di Dati, A.A. 2006-2007, prof.ssa Sonia Bergamaschi, docente presso l’Università degli studi di Modena e Reggio Emilia.
24. Domenico Beneventano, Sonia Bergamaschi, Francesco Guerra, Maurizio Vincini, “Progetto di Basi di Dati Relazionali: lezioni ed esercizi”, Pitagora Editrice Bologna, 2007.
25. Paolo Atzeni, Stefano Ceri, Stefano Paraboschi, Riccardo Torlone, “Basi di dati: Modelli e linguaggi di interrogazione”, Seconda Edizione, McGraw-Hill, 2006.
26. http://www.di.unipi.it/~leoni/BDeSI/4.Basi%20di%20Dati%20Relazionali.pdf