Sistema Informativo per il Controllo della Velocità (SICVe)

Architettura software della UEL

Cod. Doc.: T.SCV.W.010.02.01-UEL ARC
Ed.Rev.: 02.01
Data emissione: 26/05/2017 17:49

0 CONTROLLO DEL DOCUMENTO

0.1 Identificazione del documento

Codice : T.SICVe.UEL.W.01.00	File :TSICVeUELW0100 - Documento architettura software
Titolo:*Sistema Informativo per il Controllo della Velocità (SICVe)*	

0.2 Stato delle revisioni

Revisione n°	Motivo della revisione	Data
01	Primo rilascio	15/05/2008

0.3 Gestione documento

Redatto da: 4IT S.r.l.

	Nome	Funzione	Data	Firma
Predisposto da:				
Revisionato da:				
Approvato da:				

0.4 Controllo delle copie

Copia n°	Destinatario	Ente	Note
1			
2			
3			

Sommario

1 INTRODUZIONE

1.1 Obiettivo

Il documento dell'architettura software è un elaborato fondamentale del modello di Progetto di UP; esso descrive le grandi idee dell'architettura, comprese le decisioni dell'analisi architetturale. In pratica ha lo scopo di comprendere le idee fondamentali del sistema realizzato.

1.2 Definizioni, acronimi, abbreviazioni

Questa sottosezione fornisce le definizioni di tutti i termini, acronimi, e abbreviazioni richieste per interpretare nel modo corretto questo documento.

SICVe	Sistema Informativo per il Controllo della Velocità
UEL	Unità di Elaborazione Locale
URV	Unità di Rilevazione della Velocità
J2EE	Java 2 Enterprise Edition
EJB	Enterprise Java Bean
JMS	Java Message Service
MDB	Message Driven Bean
JMX	Java Management eXtention

1.3 Riferimenti

Questo paragrafo contiene l'elenco delle pubblicazioni riferiti in questo documento tramite l'indicazione [x]

[1] Titolo:	TSICVeUELS0100-Requisiti del Software
Cod. Doc.:	T.SICVe.UEL.S
Data Doc.:	15/05/2008
Ed. Rev.:	01.00
[2] Titolo:	TSICVeUELM0100-Modello dei casi d'uso
Cod. Doc.:	T.SICVe.UEL.M
Data Doc.:	15/05/2008
Ed. Rev.:	01.00
[3] Titolo:	TSICVeUELMD0100 - Modello dei Dati
Cod. Doc.:	T.SICVe.UEL.MD

Data Doc.:	15/05/2008
Ed. Rev.:	01.00

2 RAPPRESENTAZIONE ARCHITETTURALE

L'architettura di riferimento dell'applicazione soggetto di tale documento, è stata rappresentata seguendo le raccomandazioni del Rational Unified Process e secondo le direttive dettate dalle procedure aziendali per il ciclo di vita del software. La rappresentazione dell'architettura si snoda attraverso la rappresentazione di diagrammi UML raggruppati secondo viste come descritto nella prossima figura.

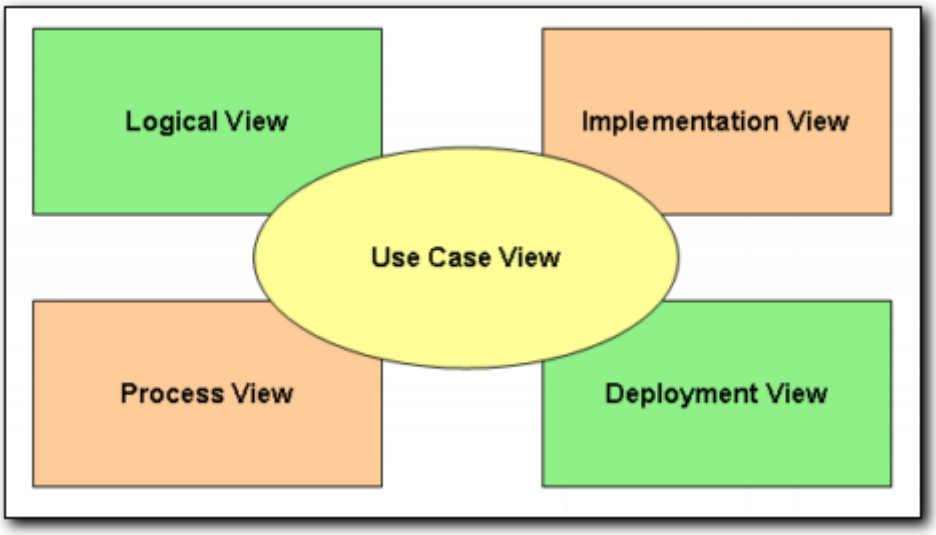

Figura 1 - Viste del'architettura

Unified Modeling Language è un linguaggio per tracciare le specifiche, disegnare e documentare la lavorazione di sistemi informativi complessi, mediante la definizione di modelli in grado di astrarre i vari aspetti del problema da risolvere. Di fatto, il linguaggio UML non intende fornire una metodologia per l'approccio al problema, ma lascia libero l'analista di decidere quale sia la strategia migliore da adottare ed il livello di astrazione di ogni modello.

I diagrammi UML danno percezione dell'intero sistema da diversi punti di vista. UML consente di disegnare il sistema definendo le entità che lo compongono, disegnando quali oggetti concorrono alla definizione di una funzionalità ed infine tutte le possibili interazioni tra gli oggetti.

3 FATTORI ARCHITETTURALI

L'Unita di Elaborazione Locale (UEL) costituisce un modulo all'interno di una architettura più ampia, costituita dalla UEL stessa, da un Server Centrale e da diverse Web Application. La fase di reingegnerizzazione della UEL deve tenere conto delle varie interazioni con gli altri sistemi, per permettere il funzionamento del sistema SICVe nella sua globalità.

La figura seguente mostra l'architettura completa del sistema SICVe.

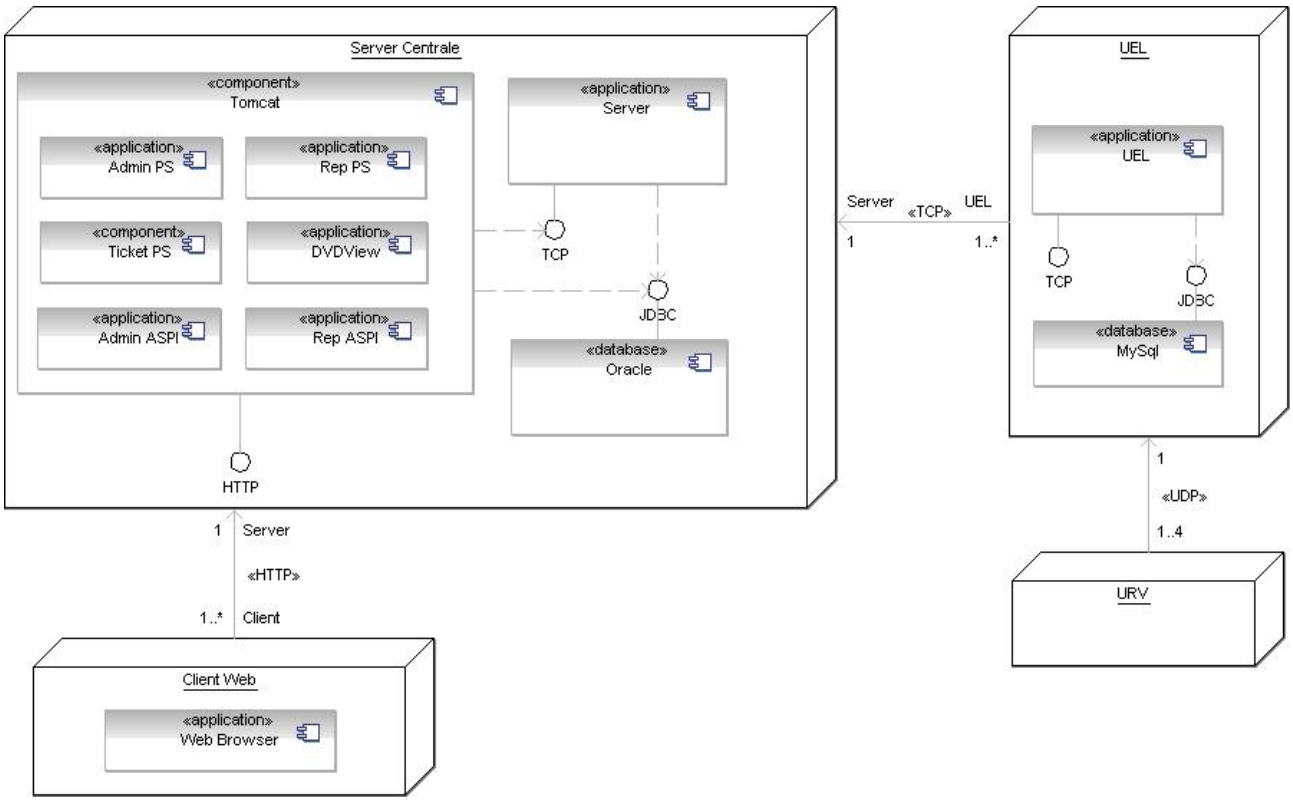

Figura 2 - Architettura del sistema SICVe

Gli obiettivi principali della reingegnerizzazione della UEL sono il rifacimento della comunicazione con il Server Centrale, della comunicazione con le URVs e delle funzionalità interne alla stessa UEL.

In questa ottica, la reingegnerizzazione della UEL si estende verso una parte del Server Centrale, in particolare verso i componenti che si occupano della comunicazione con i sistemi periferici (URV e UEL stessa) e del salvataggio dei dati. Gli elementi cardini che guidano la scelta dell'architettura da realizzare sono:

- La gestione completa e puntuale dello Stato Operativo della UEL, in particolare sarà necessario individuare un modello di macchina a stati finiti che consenta la verifica delle

funzionalità per ciascuno stato

- La comunicazione tra UEL e Server Centrale deve garantire la consegna sicura ed affidabile di tutti i messaggi scambiati; deve essere garantito che ogni messaggio inviato venga consegnato correttamente, che il canale di trasmissione sia sicuro e che la consegna avvenga entro un certo slot di tempo.
- La comunicazione con le URV deve garantire la ricezione di tutte le immagini inviate, il controllo sull'esatta sequenza di ricezione, nonché il completo controllo delle unità stesse.
- Le funzionalità interne della UEL devono garantire l'integrità dei dati, la transazionalità delle operazioni svolte e il rispetto delle prestazioni.
- Deve essere garantita la perfetta integrazione con la parte del Server Centrale e le applicazioni Web non oggetto della reingegnerizzazione.

4 DECISIONI ARCHITETTURALI

In questo capitolo viene descritto l'insieme dei promemoria tecnici che riassumono le decisioni architetturali prese per risolvere i fattori descritti nel capitolo precedente.

4.1 Promemoria tecnico: Gestione dello stato della UEL

4.1.1 Fattori

- La UEL deve essere gestita come una macchina a stati; deve essere chiaro, in ogni istante, in quale stato si trova e quali funzioni sta svolgendo.
- Deve essere possibile, in seguito alla caduta del sistema, ritornare nello stato operativo precedente (sempre a seguito dell'attivazione del sistema periferico da parte di un operatore di Polizia)

4.1.2 Soluzione

Attualmente lo stato operativo della UEL non viene gestito in modo localizzato. La soluzione proposta prevede di suddividere le funzionalità della UEL tra diversi moduli indipendenti, nel seguito servizi, e di demandare la gestione di tali servizi ad un componente principale.

Tale componente è il responsabile della gestione dello stato operativo della UEL, dell'avvio, dell'arresto e della notifica dei cambiamenti di stato agli altri servizi sottostanti. Inoltre è previsto che lo stato operativo della UEL venga memorizzato in modo persistente su database, in modo che, a seguito della caduta del sistema, sia possibile ripristinare lo stato precedente.

4.1.3 Motivazione

La presenza di un componente principale (nel seguito UELManager) che si occupi di gestire lo stato operativo della UEL, e che orchestri i servizi sottostanti, permette di centralizzare la gestione della macchina a stati all'interno di uno specifico componente dell'applicazione. I servizi sottostanti si limitano, quindi, a svolgere le proprie funzionalità sulla base dello stato stesso della UEL.

A seguito del verificarsi di un evento che richiede la transizione da uno stato operativo a l'altro (ad esempio l'avvio di un servizio di verifica violazioni), il servizio che ha provocato tale evento si limita a notificare all'UEL Manager l'evento stesso. L'UEL Manager, conoscendo la logica della

macchina a stati della UEL, si prende carico di aggiornare lo stato operativo e di informare tutti i servizi sottostanti interessati della transizione di stato occorsa. I servizi sottostanti si occuperanno di modificare il proprio funzionamento interno, sulla base dell'informazione ricevuta dall'UEL Manager.

A seguito di una caduta di sistema della UEL, e conseguente riavvio, l'UELManager si occupa di verificare lo stato operativo precedente, e, successivamente alla fase di attivazione del sistema periferico, esegue le operazioni necessarie per tornare in tale stato operativo.

4.2 Promemoria tecnico: Affidabilità e scalabilità della UEL

4.2.1 Fattori

- La UEL deve svolgere le proprie funzioni in modo affidabile, garantendo l'integrità dei dati elaborati;
- La UEL deve svolgere le proprie funzionalità in modo ottimale anche in seguito ad un aumento del carico di lavoro.

4.2.2 Soluzione

La soluzione proposta prevede l'utilizzo della piattaforma J2EE per la realizzazione della nuova UEL. Tale tecnologia rappresenta lo standard de facto per le realizzazione di applicazioni enterprise distribuite.

La tecnologia principale fornita dalla piattaforma J2EE è rappresentata dagli EJB (Enterprise Java Bean). Tale tecnologia fornisce supporto per la gestione della persistenza, il supporto alle transazioni, la gestione della concorrenza e della sicurezza, nonché l'integrazione con altre tecnologie, come JMS (Java Message Service).

In particolare, per quanto riguarda la UEL, verranno utilizzati i seguenti due tipi di EJB:

- EJB di sessione (Session EJB): gestiscono l'elaborazione delle informazioni dell'applicazione. Forniscono un'interfaccia pubblica che permettere di richiedere l'esecuzione di specifiche funzionalità; in particolare, verranno utilizzati gli EJB si sessione senza stato, i quali non mantengono informazioni relative all'utilizzatore(client) del componente stesso. In questo modo, al termine dell'esecuzione di un metodo di un EJB, l'EJB stesso torna subito disponibile per soddisfare la richiesta di un differente client. Gli EJB Session vengono gestiti

all'interno di un pool da parte dell'Application Server.

- EJB guidati da messaggi (message-driven-bean): sono gli unici componenti con funzionamento asincrono. Tramite Java Message Service (JMS), vengono registrati su una particolare coda, e si attivano alla ricezione dei messaggi inviati sula coda a cui sono iscritti. Permettono di gestire in modo parallelo l'arrivo di più messaggi sulla stessa coda, attraverso un meccanismo di pooling gestito interamente dall'Application Server.
- EJB timer: sono degli EJB Session che, attraverso una opportuna interfaccia, permettono l'esecuzione schedulata di particolari funzioni. Vengono utilizzati per svolgere quelle funzionalità che sono eseguite ad intervalli regolari (come ad esempio, l'aggregazione dei dati statistici).

Gli EJB sono dei componenti che vivono all'interno di un Application Server, che ne gestisce il ciclo di vita. Un Application Server rappresenta un sistema che fornisce dei servizi di fondamentale importanza per le applicazioni J2EE, sollevando il programmatore dall'onere di dover gestire alcune problematiche in modo diretto, come ad esempio la gestione della comunicazione, la transazionalità e le connessioni al database, la gestione dello stato e il multi-thread.

La scelta di quale Application Server utilizzare è ricaduta su JBoss.

4.2.3 Motivazione

Attraverso l'utilizzo della tecnologia J2EE, si può usufruire di diversi servizi, quali la concorrenza, le transazioni e la sicurezza, la cui gestione viene demandata direttamente all'Application Server utilizzato. In questo modo si ha la garanzia dell'affidabilità della UEL stessa, ed inoltre, attraverso l'uso della transazioni, viene garantita l'integrità dei dati. La gestione in pooling dei componenti EJB, permette all'Application Server di scalare correttamente nel caso aumenti il carico di lavoro.

La scelta di utilizzare JBoss è stata fatta sulla base delle seguenti considerazioni:

- E' un prodotto open source con la più vasta comunità di supporto;
- E' uno dei prodotti più utilizzati ed affidabili del mercato;
- Fornisce, direttamente integrato, un sistema di messaggistica JMS (JBossMQ) di livello Enterprise.

4.3 Promemoria tecnico: Comunicazione con il Server Centrale

4.3.1 Fattori

- La comunicazione con il Server Centrale deve garantire la corretta consegna di tutti i messaggi scambiati;
- La comunicazione con il Server Centrale deve garantire gli stessi tempi di consegna dei messaggi anche in caso di aumento del carico di lavoro.

4.3.2 Soluzione

La soluzione proposta prevede che la comunicazione tra UEL e Server Centrale avvenga mediante l'utilizzo di JMS (Java Message Service).

JMS definisce un insieme di API capaci di fornire servizi di messaggistica nel software; più precisamente si tratta di una serie d'interfacce che permettono di accedere e di utilizzare i servizi di un sistema di middleware orientato ai messaggi, utilizzando Java come linguaggio.
In sostanza JMS fornisce un metodo standard di notifica degli eventi tramite il quale le applicazioni possono creare, inviare e ricevere i messaggi.

E' previsto, sia sul Server Centrale che sulle UEL, la presenza di un Provider JMS (in particolare, JBoss MQ) attraverso il quale, la UEL e il Server Centrale, ricevono i messaggi inviati dalla controparte. Compito del provider JBoss MQ è garantire l'affidabilità della comunicazione, mentre grazie all'utilizzo dei componenti MDB viene garantita la scalabilità del sistema in caso di aumento del carico di lavoro.

4.3.3 Motivazione

I sistemi di messaggistica rappresentano il modo migliore per implementare lo scambio di messaggi tra applicazioni distribuite. In particolare, l'utilizzo di JMS permette di disaccoppiare l'applicazione dal provider JMS che si occupa realmente della comunicazione, garantendo, di fatto, la portabilità dell'applicazione su qualsiasi provider JMS. Attraverso l'utilizzo di code, è possibile suddividere i flussi dati tra canali differenti, demandandone la gestione direttamente al provider JMS.

JMS è per natura un sistema di messaggistica asincrona, ma permette comunque di implementare scenari di comunicazione sincrona, attraverso l'utilizzo di code di richiesta e code di risposta.

La gestione delle code, del rinvio dei messaggi e del pool di connessioni viene demandata

completamente al provider JMS. Le applicazioni devono solamente utilizzare le API JMS per accedere a tutti i servizi resi disponibili dal provider JMS.

4.4 Promemoria tecnico: Comunicazione con le URVs

4.4.1 Fattori

- La comunicazione con le URVs deve garantire la corretta ricezione di tutte le immagini inviate;
- La comunicazione con le URVs deve garantire l'utilizzo minimo necessario delle risorse di sistema.

4.4.2 Soluzione

La soluzione proposta prevede di utilizzare il protocollo TCP per lo scambio di messaggi con le URVs, in sostituzione dell'attuale protocollo proprietario TOP/TBP su UDP.

E' inoltre previsto un nuovo tipo messaggio che informi la UEL della ricezione di un'immagine tramite FTP, invece di dover effettuare periodicamente un'operazione di "dir" delle directory del server FTP per verificare la presenza di nuove immagini.

La comunicazione tra UEL e URVS viene gestita da un componente particolare, denominato Resource Adapter, che si occupa di mascherare i dettagli della comunicazione ai servizi soprastanti. Il Resource Adapter e un componente specifico della piattaforma J2EE.

4.4.3 Motivazione

La verifica della presenza di nuove immagini scaricate è attualmente implementata effettuando, periodicamente, una dir per ogni directory all'interno dell'alberatura del server FTP. Tale modalità ingenera un sovraccarico della UEL stessa, eseguendo periodicamente operazioni di lettura da file-system che risultano molto pesanti a livello prestazionale.

In questa ottica è stato pensata l'introduzione di un nuovo tipo di messaggio che informi la UEL dell'arrivo di un'immagine dal server FTP. In questo modo, non è più necessario effettuare la dir delle directory del server FTP, ma, alla ricezione del messaggio stesso, la UEL può prelevare direttamente l'immagine, essendo a conoscenza del percorso e del nome file. In questo modo vengono risolti i problemi di concorrenza per l'accesso al file-system, e si ha la garanzia che le

immagini vengano elaborate dalla UEL nello stesso ordine in cui vengono inviate dalle URVs.

Poiché il numero di messaggi inviati dalle URVs aumenta sostanzialmente con introduzione di quest'ultimo tipo di messaggio, si è deciso di sostituire l'attuale protocollo di comunicazione, TOP/TBP proprietario su UDP, con il più comune protocollo di comunicazione TCP. Il protocollo TCP garantisce la consegna dei messaggi inviati, inoltre, a differenza del TOP/TBP, si occupa di riordinare in sequenza i pacchetti dati che costituiscono i messaggi stessi. Il protocollo TOP/TBP prevede invece lo scarto dei messaggi i cui pacchetti non arrivino in sequenza, penalizzando quindi l'utilizzo della banda disponibile.

Attraverso il protocollo TCP, dunque, si aumenta l'affidabilità del canale di comunicazione e si ottimizza l'utilizzo della banda stessa.

L'utilizzo di un Resource Adapter permette di disaccoppiare la gestione della comunicazione con l'elaborazione del messaggio ricevuto. In questo modo, le modifiche al protocollo di comunicazione non impattano su servizi che utilizzano i messaggi provenienti dalle URVs; viene realizzato dunque un forte disaccoppiamento tra la logica di business e la gestione della comunicazione.

4.5 Promemoria tecnico: Integrazione con il Server Centrale

4.5.1 Fattori

- La nuova UEL deve integrarsi perfettamente con quei moduli non oggetto della reingegnerizzazione (parte del Server Centrale e Web Application).

4.5.2 Soluzione

La soluzione proposta prevede di estendere la reingegnerizzazione della UEL ai moduli del Server Centrale che si occupano direttamente di comunicare con la UEL stessa. Resteranno esclusi solamente i moduli non connessi alla comunicazione (flusso delle violazioni, MCTC, export PS).

Il vecchio Server Centrale, oltre a svolgere le attività non connesse alla comunicazione, si occuperà, in una fase transitoria, di comunicare con le vecchie UEL (versione software precedente alla reingegnerizzazione). Il nuovo Server Centrale, invece, si occuperà di dialogare con le nuove UEL. In tale scenario, sarà necessario modificare le Web Application, che necessitano di comunicare con le UEL, per inserire un controllo che indirizzi la particolare richiesta effettuata verso il Server Centrale che gestisce la UEL interessata. In pratica, se viene richiesta un'operazione che coinvolge

una UEL con la nuova versione del software, la richiesta sarà demandata al nuovo Server Centrale, altrimenti verrà demandata al vecchio. Le applicazioni web coinvolte saranno le due di amministrazione e Ticket PS. Un'altra sezione del server centrale oggetto di adattamento è quella responsabile del salvataggio di tutti i dati provenienti dalle nuove UEL ed i processi dedicati allo scarico e alla cancellazione delle immagini .

4.5.3 Motivazione

La reingegnerizzazione dei moduli di comunicazione del Server Centrale è dovuta alla modifica del protocollo utilizzato per colloquiare con le UEL (da TCP/IP a JMS).

La decisione di modificare le Web Application per dialogare con il nuovo Server Centrale è stata fatta partendo dalla considerazione che, nel breve, tutte le UEL saranno collegate al nuovo Server Centrale, e quindi le Web Application non avranno più necessità di interagire con il vecchio. Tutta la parte di comunicazione del vecchio Server Centrale potrà essere rimossa, lasciandogli il solo compito di svolgere operazioni che coinvolgono solamente l'utilizzo del database (operazioni batch).

Una soluzione alternativa è quella di indirizzare tutte le richieste delle Web Application verso il vecchio Server Centrale e delegare a quest'ultimo il compito di reindirizzare le richieste al nuovo Server Centrale nel caso si tratti di richieste indirizzate alle nuove UEL. In questo modo, le Web Application non necessitano di modifiche, ma si rimane legati sempre al vecchio Server Centrale, che non potrà essere quindi sostituito.

La scelta di rifare anche la parte che riguarda il salvataggio dei dati provenienti dalle UEL è nata dal fatto di rendere stabile una parte molto importante dell'intero sistema che nella sua versione attuale non garantisce un livello minino di sicurezza

La scelta di modificare i processi dedicati allo scarico e alla cancellazione delle immagini si rende necessaria al fine di rispettare le specifiche approvate in fase di analisi

5 LOGICAL VIEW

Questo capitolo si occupa di illustrare l'architettura logica della UEL attraverso la descrizione dei componenti che costituiscono la UEL stessa e delle iterazioni tra di essi. Vengono inoltre descritte le applicazioni stand-alone utilizzate per la configurazione e l'attivazione del sistema periferico (singola o doppia istanza); attraverso tali applicazione viene reso possibile il corretto avvio e funzionamento del sistema periferico stesso.

5.1 Sistema periferico

Il termine sistema periferico viene utilizzato per indicare il server fisico dove possono essere in esecuzione una o due istanza dell'applicazione UEL. Con il termine UEL, viene fatto riferimento solamente all'applicazione che si occupa della ricezione dei dati dalle telecamere (URV) e dell'invio dell'elaborazione di tali dati verso il Server Centrale.

Nel caso in cui sia in esecuzione una sola istanza del processo UEL su un sistema periferico, si parlerà di singola istanza ed i termini UEL e sistema periferico potranno essere utilizzati per indicare lo stesso sito di rilevazione; nel caso siano presenti due istanze differenti del processo UEL sullo stesso sistema periferico, si parlerà di doppia istanza ed i termini UEL e sistema periferico non potranno essere confusi tra di loro.

La possibilità di utilizzare due differenti istanze del processo UEL, in esecuzione su un singolo sistema periferico, è prevista qualora siano presenti due gruppi distinti di URVs, ognuno dei quali orientato in uno senso di marcia possibili. In questo caso, ogni istanza del processo UEL si occupa di gestire uno dei gruppi di URVs, occupandosi della ricezione, elaborazione ed invio al Server Centrale dei dati relativi al solo senso di marcia monitorato (carreggiata).

Le due istanze del processo UEL sono indipendenti tra di loro e non è prevista nessuna interazione tra le due istanze. Lato Server Centrale, il fatto che le due istanze si trovino sullo stesso sistema periferico piuttosto che su due differenti sistemi periferici è indifferente e verranno gestite in maniera trasparente.

L'unica eccezione viene fatta dalle applicazioni stand-alone utilizzate per la configurazione ed attivazione del sistema periferico; tali applicazioni, infatti, si occuperanno di svolgere le proprie funzionalità per entrambe le UEL, come descritto nei paragrafi successivi.

L'identificazione di una UEL avviene in modo univoco attraverso il nome e il numero di istanza

(1 o 2); non è quindi necessario che il nome di due UEL sullo stesso sistema periferico sia differente; si consiglia però vivamente di utilizzare nomi differenti, per non ingenerare problemi di identificazione delle UEL stesse a quei moduli (parte del Server Centrale ed Applicazioni Web) che non sono oggetto della reingegnerizzazione della UEL.

I paragrafi successivi descrivono le due applicazioni stand-alone utilizzate per la configurazione ed attivazione del sistema periferico, quindi viene descritta l'architettura logica dell'applicazione UEL propriamente detta.

5.2 Configurazione del sistema periferico

Una volta installato, il sistema periferico necessita di alcune attività preliminari di configurazione per permetterne il corretto funzionamento. La configurazione del sistema periferico avviene per mezzo di una specifica applicazione stand-alone TUI (Text User Interface), denominata PConfTUI. Tale applicazione può essere utilizzata solamente da uno specifico utente di sistema (utente Linux), l'utente manutentore. Al momento della login dell'utente manutentore sul sistema, l'applicazione PConfTUI verrà avviata in modalità automatica. Al termine della configurazione, l'utente manutentore salva la configurazione stessa ed esegue il logout dall'applicazione; in automatico il sistema provvederà ad effettuare il logout dell'utente dal sistema.

Le operazioni permesse all'utente manutentore tramite l'applicazione PConfTUI sono le seguenti

- configurare i parametri della rete (indirizzo IP, nome macchina, gateway, etc.);
- impostare/verificare il collegamento con il Server Centrale;
- verificare il funzionamento dell'NTP e del GPS;
- configurare i parametri di funzionamento di una od entrambe le istanze della UEL (Nome istanza, numero corsie, direzione);
- configurare la tipologia di tecnologia utilizzata dalla UEL (OCR o PlateMatching)
- configurare i parametri di funzionamento del PlateMatching (parametri di start di EnDetector, ecc.)
- abilitare/disabilitare una od entrambe le istanze della UEL
- visualizzare i dati raccolti per le statistiche di traffico locali per ognuna delle istanze;
- visualizzare lo stato delle istanze della UEL;
- verificare la tipologia di UEL
- visualizzare i log dell'EnDetector

- visualizzare gli errori generati dalle istanze della UEL e dalle relative URVs;
- visualizzare i log delle istanze della UEL.

Al termine della procedura di configurazione del sistema periferico, le impostazioni effettuate verranno salvate in maniera opportuna attraverso file XML, utilizzato successivamente dalle istanze UEL per caricare i parametri necessari al corretto funzionamento. Inoltre, ove modificate, vengono salvate le impostazioni sulla configurazione di rete del sistema periferico stesso. Tale operazione necessita il riavvio del sistema periferico stesso.

La figura seguente descrive l'architettura dell'applicazione ConfTUI.

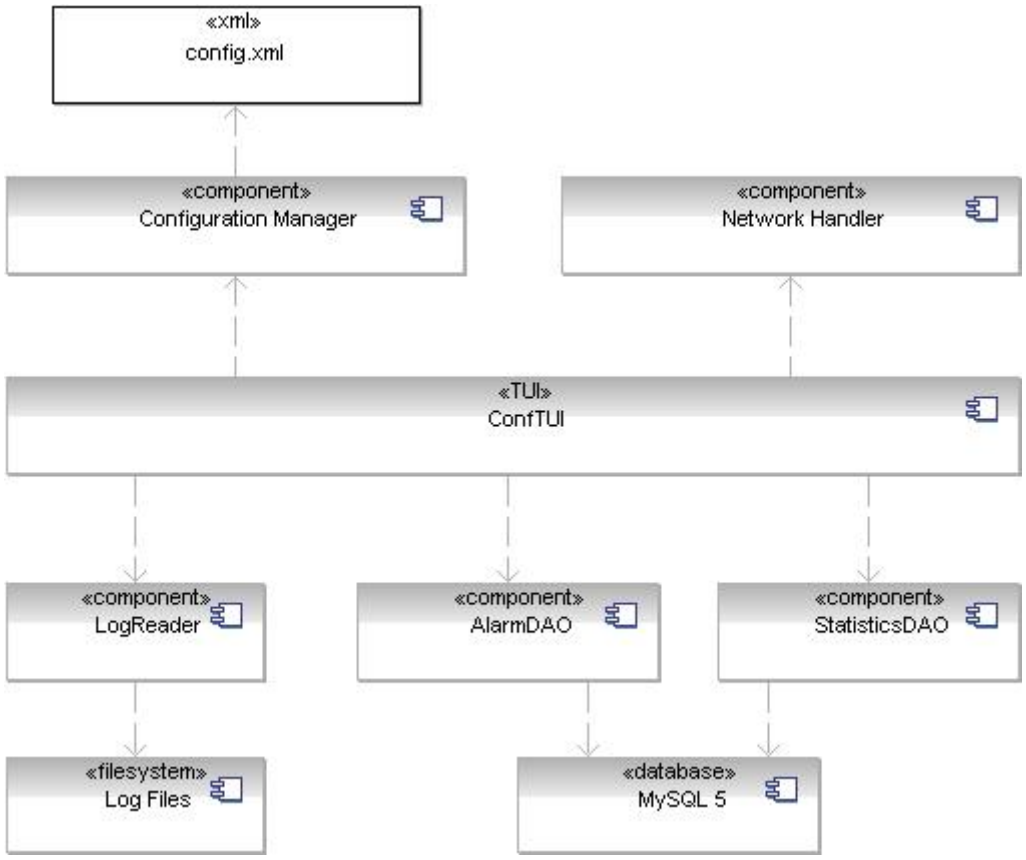

Figura 3, ConfTUI, Architettura logica

L'applicazione PConfTUI utilizza diversi moduli per svolgere le proprie funzioni. Nello specifico, vengono utilizzati i seguenti moduli:

- ConfigurationManager, si occupa di caricare e salvare la configurazione del sistema periferico da file XML;
- NetworkHandler, si occupa di configurare l'interfaccia di rete del sistema periferico;
- LogReader, si occupa di effettuare la lettura dei file di log associati ad ognuna delle istanze di UEL presenti;

- AlarmDAO e StatisticsDAO, consentono di effettuare l'accesso ai dati sul database per quel che riguarda, rispettivamente, gli allarmi generati dalle URVs/UEL e le statistiche calcolate in locale.

Se una od entrambe e istanze sono state abilitate, al termine dell'applicazione PConfTUI, o comunque ad ogni riavvio del sistema periferico, verranno avviate e si porranno nello stato "Statistiche in locale" (vedi paragrafi successivi), rimanendo in attesa che venga effettuata la procedura di attivazione.

L'applicazione PConfTUI sarà disponibile solamente durante il funzionamento della UEL in "Statistiche in locale"; a seguito della fase di attivazione l'acceso al sistema per l'utente manutentore sarà inibito, quindi non sarà più possibile eseguire l'applicazione PConfTUI.

5.3 Attivazione del sistema periferico

L'attivazione è quel processo che permette alla UEL (od ad entrambe, nel caso di doppia istanza) di scaricare dal Server Centrale le proprie quantità di sicurezza necessarie per la cifratura del canale di comunicazione e dei dati sensibili, nonché di una parte della configurazione della UEL presente proprio sul Server Centrale. Al termine di questa fase, la UEL sarà in grado di comunicare in maniera sicura con il Server Centrale e di avviare gli altri servizi. La conclusione, con esito positivo, della fase di attivazione è requisito fondamentale per l'esecuzione delle attività della UEL.

La fase di attivazione viene eseguita attraverso uno specifico operatore di sistema, l'utente attivatore (Pubblico Ufficiale). Requisito fondamentale è il possesso di una smart-card contenete il certificato di attivazione della UEL. Tale certificato serve ad instaurare una connessione sicura con il Server Centrale e può essere usato solamente durante la fase di attivazione, durante le successive operazioni verrà invece utilizzato il certificato di autenticazione ricevuto dal Server Centrale.

La fase di attivazione viene effettuata attraverso l'applicazione stand-alone SysAct.

Il Pubblico Ufficiale, utilizzando l'utenza di sistema *attivatore*, effettua la login sul sistema operativo del sistema periferico; automaticamente viene eseguita l'applicazione SysAct. Attraverso l'interfaccia testuale proposta, l'utente attivatore è in grado di attivare entrambe le istanze eventualmente presenti sul sistema periferico. Nel caso di doppia istanza, l'attivazione verrà eseguita contemporaneamente per entrambe le istanze. Durante la fase di attivazione la UEL comunica se si tratta di una UEL con tecnologia OCR o tecnologia PM, e il server centrale effettua le dovute verifiche di coerenza tra tipologia UEL attesa e tipologia dichiarata in fase di attivazione.

Al termine della fase di attivazione, non sarà più possibile accedere in alcun modo sul sistema

periferico.

La descrizione dei flussi dei dati scambiati tra UEL e Server Centrale viene descritta in dettaglio nel paragrafo che descrive il servizio di attivazione.

5.4 Servizi della UEL

Questo paragrafo descrive un generico servizio in esecuzione sulla UEL, a partire dal significato stesso del concetto di servizio fino alla sua architettura.

Un servizio è un componente auto-contenuto che si occupa di svolgere un insieme di funzionalità comuni, come ad esempio la gestione delle URVs; il termine auto-contenuto si riferisce al fatto che un servizio incorpora tutto ciò di cui ha bisogno per poter funzionare correttamente. Un servizio espone un'interfaccia pubblica, che può essere utilizzata dagli altri servizi, per eseguire particolari funzioni. Tale interfaccia rappresenta l'unico punto di accesso al servizio stesso. Un servizio, per poter svolgere le proprie funzionalità, si avvarrà dell'utilizzo di altri sotto-componenti, solitamente EJB di tipo Session ed EJB a messaggi (MDB). La figura seguente mostra l'architettura logica generale di un servizio:

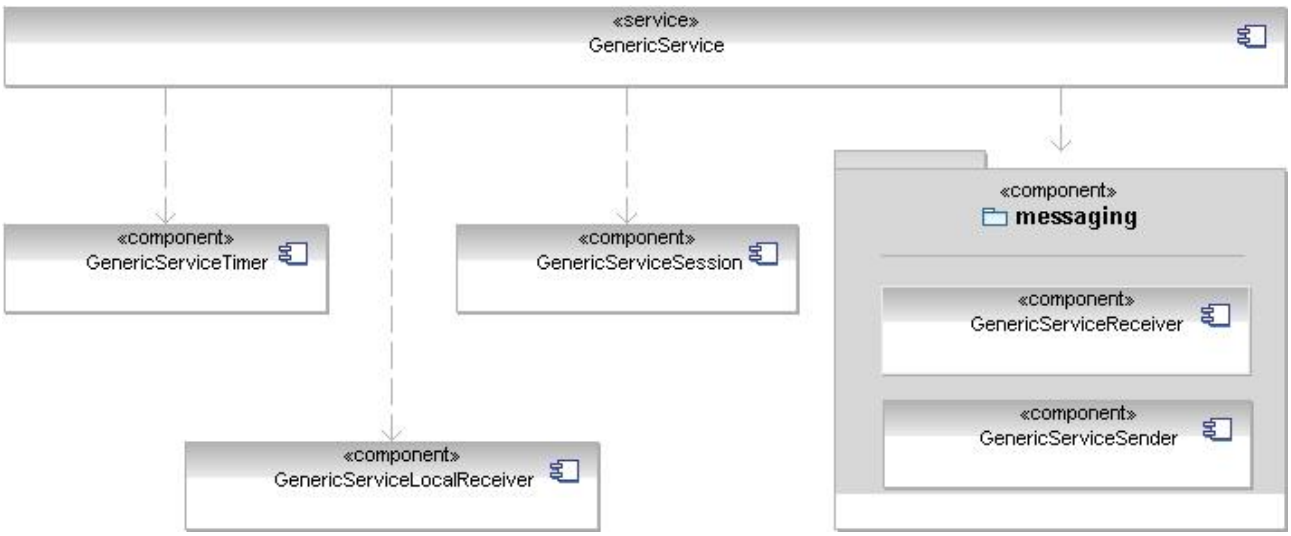

Figura 4 - Architettura di un servizio generico

Un servizio può utilizzare tutti i componenti sopra descritti oppure utilizzarne solo in parte. Di seguito vengono descritti i componenti rappresentati in figura:

- Timer, vengono utilizzati nel caso il servizio debba svolgere delle operazioni schedulate nel tempo (ad esempio, il controllo dei messaggi di vita dalle URVs);
- Session, vengono utilizzati per svolgere delle operazioni di elaborazione (come ad esempio l'invio di un comando verso una URV);
- LocalReceiver, vengono utilizzati per ricevere messaggi asincroni da altri servizi interni (ad esempio, la ricezione dei messaggi di transito inviati dal servizio URVManager al servizio

Capture)

- Sender, utilizzato per inviare messaggi verso il Server Centrale;
- Receiver, utilizzato per ricevere messaggi dal Server Centrale.
- GenericService, utilizzato per lo start e/o lo stop dell'intero servizio (o di alcuni dei suoi sottomoduli) in base allo stato in cui si trova la UEL

Nei capitoli successivi verranno descritti tutti i servizi in esecuzione sulla UEL, e, per ognuno di essi, verrà fornita l'architettura logica interna, definendo quali componenti vengono utilizzati dal servizio stesso e quali no.

5.5 Architettura logica della UEL

L'architettura logica della UEL prevede la suddivisone delle funzionalità della UEL stessa tra differenti sottomoduli, nel seguito servizi, indipendenti tra di loro, che comunicano attraverso lo scambio di messaggi asincroni o attraverso delle interfacce pubbliche ben definite.

E' presente un servizio principale che si occupa di orchestrare tutti i sottoservizi presenti. Tale servizio, l'UELManager, è il primo servizio ad essere avviato e si occupa principalmente di gestire lo stato operativo della UEL stessa e le transazioni da uno stato ad un altro, informando i servizi sottostanti dell'avvenuto cambiamento. L'UELManager si occupa quindi di gestire i seguenti sottoservizi:

- URVManager, si occupa di gestire tutto ciò che è inerente alla comunicazione con le URVs;
- PActivationService, si occupa di gestire la fase di attivazione della UEL;
- PRCService, si occupa di gestire le operazioni di monitoraggio e di controllo remoto della UEL da parte del Server Centrale;
- PCaptureService, si occupa di gestire l'elaborazione e l'invio al Server Centrale dei dati relativi alle rilevazioni effettuate dalle URVs;
- PStatisticsService, si occupa dell'elaborazione e l'invio delle statistiche di traffico verso il Server Centrale;
- SecurityManager, fornisce agli altri servizi funzionalità per la gestione della sicurezza (autenticazione, firma, etc.);
- Messaging, fornisce agli altri servizi funzionalità per l'invio e la ricezione di messaggi (JMS) con il Server Centrale. Di per se non costituisce un servizio, ma piuttosto un sottosistema a disposizione dei servizi atti alla comunicazione con il Server Centrale.

La figura seguente mostra l'architettura logica della UEL, mettendo in risalto il disaccoppiamento introdotto tra i differenti servizi descritti:

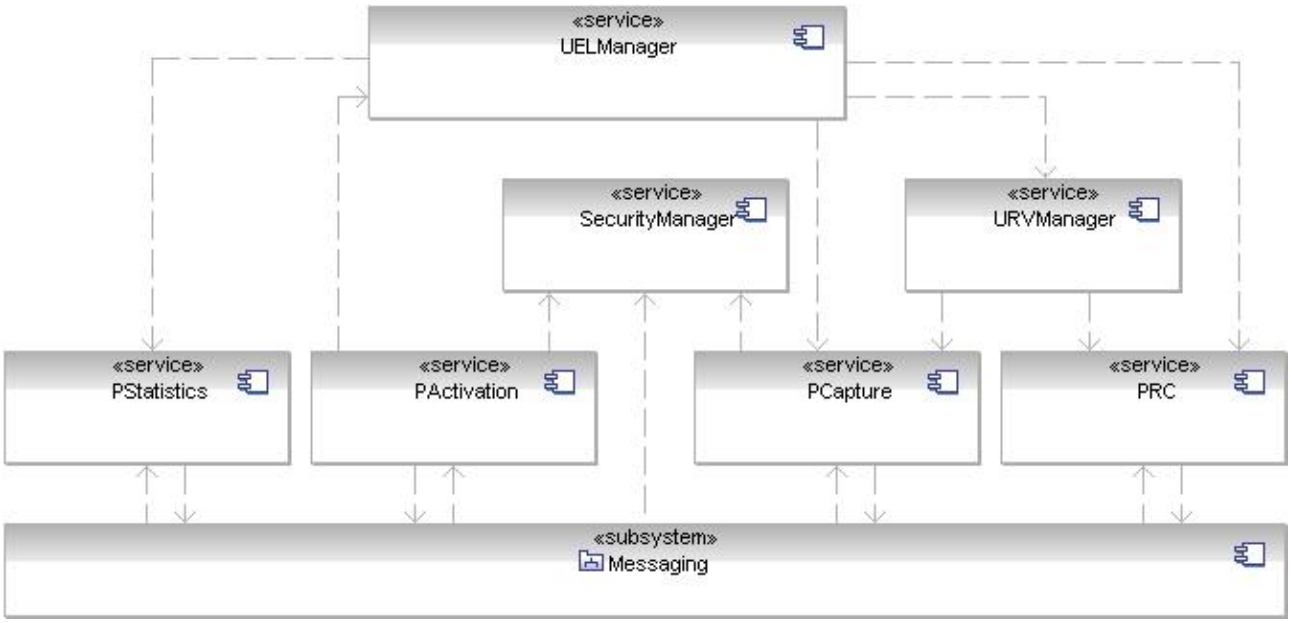

Figura 5 - UEL, Diagramma dei componenti

Nel seguito del capitolo vengono descritti i servizi introdotti precedentemente, definendone le funzionalità svolte, l'architettura interna e le interazioni con gli altri servizi.

5.6 UELManager

Come già anticipato, l'UELManager è il servizio principale della UEL, responsabile della gestione dello stato operativo della UEL stessa e dell'orchestrazione degli altri servizi sulla base di quest'ultimo. L'UELManager si fa carico di informare gli altri servizi del cambiamento di stato della UEL, al fine di permettere lo svolgimento o meno delle funzionalità richieste durante lo stato operativo raggiunto. Ad esempio, a seguito della fase di attivazione, l'UELManager informerà i servizi interessati alla comunicazione con il Server Centrale, di poter iniziare ad inviare e ricevere messaggi dal Server Centrale stesso. Inoltre tale componente si occupa anche della gestione della configurazione.

Nei paragrafi successivi vengono descritti tutti gli stati operativi della UEL, fornendo indicazioni sulle funzionalità svolte. Maggiori dettagli verranno dati nei paragrafi relativi ad ogni sottoservizio.

5.6.1 Statistiche in locale

Lo stato di Statistiche in locale è lo stato operativo in cui si trova la UEL all'avvio. Dunque è il primo stato operativo attraversato dalla UEL stessa. In questo stato, la UEL permette di effettuare operazioni ai soli utenti locali, attivatore e manutentore. Il collegamento con il Server Centrale non è

ancora stabilito.

Le operazioni svolte dalla UEL in questo stato sono le seguenti:

- Comunicazione con le URVs per la ricezione degli allarmi;
- Raccolta dei transiti (immagini) inviati dalle URVs via FTP al solo fine di raccolta dati statistici;
- Calcolo di un sottoinsieme di dati statistici per fini di controllo di funzionamento da parte dell'operatore manutentore.

Attraverso gli allarmi e i dati statistici raccolti, l'operatore manutentore potrà verificare il corretto funzionamento della UEL stessa, utilizzando l'apposita applicazione riservata ad esso. In particolare, i dati statistici raccolti in questa fase, sono i seguenti:

- Numero di veicoli transitati per corsia all'interno di un determinato slot di tempo (5 minuti, un'ora o un giorno);
- Tempo medio interveicolo per corsia all'interno di un determinato slot di tempo (5 minuti, un'ora o un giorno);
- Numero di veicoli transitati per corsia e per classe all'interno di un determinato slot di tempo (5 minuti, un'ora o un giorno);
- Velocità media dei transiti per corsia e per classe all'interno di un determinato slot di tempo (5 minuti, un'ora o un giorno).

Tutti i dati raccolti in questa fase non vengono mai inviati al Server Centrale, ma vengono cancellati quando la UEL raggiunge lo stato operativo Disponibile, a seguito della fase di attivazione. Come già detto, tali dati servono solo per permettere all'utente manutentore di verificare il corretto funzionamento della UEL stessa.

5.6.2 Disponibile

La UEL transita dallo stato Statistiche in Locale allo stato Disponibile a seguito della procedura di attivazione, eseguita dall'utente attivatore. La procedura di attivazione ha lo scopo di recuperare le quantità di sicurezza necessarie alla UEL per poter comunicare in sicurezza con il Server Centrale e di recuperare dati di configurazione presenti sul Server Centrale stesso.

Nello stato Disponibile, la UEL è in grado di comunicare con il Server Centrale attraverso un canale di comunicazione sicuro. Vengono svolte tutte le operazioni tipiche dello stato Statistiche in locale, ma viene generato un flusso dati tra UEL e Server Centrale. In particolare vengono inviati i

seguenti dati:

- Allarmi generati dalle URVs e dalla UEL stessa;
- Statistiche puntuali aggregate per slot di tempo (5 minuti, un'ora o un giorno);
- Dati di traffico non aggregati per il calcolo delle statistiche in velocità media (qualora la UEL risulta accoppiata per la generazione delle statistiche su tratta);
- Eventuali transiti rilevati durante un precedente stato di verifica violazioni (in velocità media o istantanea) non ancora inviati al Server Centrale;
- Informazioni circa l'ultimo transito inviato

  Inoltre, la UEL soddisfa le seguenti richieste ricevute dal Server Centrale:
- Comandi di amministrazione ASPI (avvio, arresto e reset delle URVs, richiesta di immagini di prova, etc.);
- Comando di avvio di un nuovo servizio di verifica violazioni (azione che provoca il passaggio di stato della UEL da disponibile a Verifica Violazioni (Media od Istantanea);
- Richiesta di immagini relativi a transiti già in possesso del Server Centrale;
- Richiesta di eliminazione di transiti (e relative immagini) da parte del Server Centrale (sia richieste puntuali che per data).
- Richiesta di cancellazione delle statistiche raccolte.
- Invio delle immagini mediante schedulazione.

5.6.3 Verifica Violazioni

La UEL viene a trovarsi nello stato di Verifica Violazioni a seguito di un comando di avvio servizio inviato da un operatore PS.

A seguito della ricezione di un comandi di avvio servizio di verifica violazioni, la UEL esegue le seguenti operazioni per iniziare una nuova fase di verifica violazioni:

- Invia alle URVs un messaggio per l'impostazione dei nuovi limiti di velocità per classe; tali limiti vengono utilizzati successivamente dalla UEL per filtrare i transiti da inviare al server nel caso di servizio di verifica violazioni in velocità istantanea
- Avvisa il servizio di cattura delle immagini di iniziare ad elaborare le immagini stesse non solo per fini statistici ma anche per fini di verifica violazioni. Le immagini vengono quindi salvate su percorso locale della UEL per poi poter essere inviate successivamente al server.

In questo stato operativo, la UEL esegue tutte le operazioni possibili nello stato Disponibile,

inoltre, come già detto, colleziona i transiti inviati dalle URVs. Saranno inibiti i comandi di amministrazione ASPI e di avvio servizi. I transiti raccolti verranno inviati al Server Centrale, dopo un'opportuna elaborazione e filtraggio degli stessi (controllo dei duplicati, controllo di integrità, firma, etc.).

5.7 URVManager

L'URVManager è il servizio che si occupa di dialogare con le URVs collegate alla UEL. Offre una interfaccia pubblica agli altri servizi per l'invio di comandi verso le URVs e si occupa di informare i servizi interessati dell'arrivo di un nuovo messaggio da parte di una delle URVs collegate. Inoltre svolge funzioni di controllo sul funzionamento delle URVs stesse.

L'URVManager è il primo servizio che viene avviato durante lo start della UEL stessa (che avviene, come già detto, attraverso il servizio principale UELManager) e resta attivo per tutto il funzionamento della UEL, svolgendo completamente le proprie funzioni, indipendentemente dallo stato operativo in cui si trova la UEL. In particolare, le funzioni svolte dal servizio URVManager sono le seguenti:

- Ricevere allarmi da parte delle URVs collegate;
- Ricevere messaggi di *trasferimento FTP completato* da parte delle URVs collegate;
- Permettere l'invio di comandi alle URVs collegate;
- Verificare periodicamente lo stato di funzionamento delle URVs collegate (messaggio di vita);
- Verificare il funzionamento del server NTP.

Ognuna delle seguenti funzionalità viene svolta da un componente dedicato, solitamente un EJB. La figura seguente descrive l'architettura del servizio URVManager in dettaglio, specificando i vari componenti utilizzati:

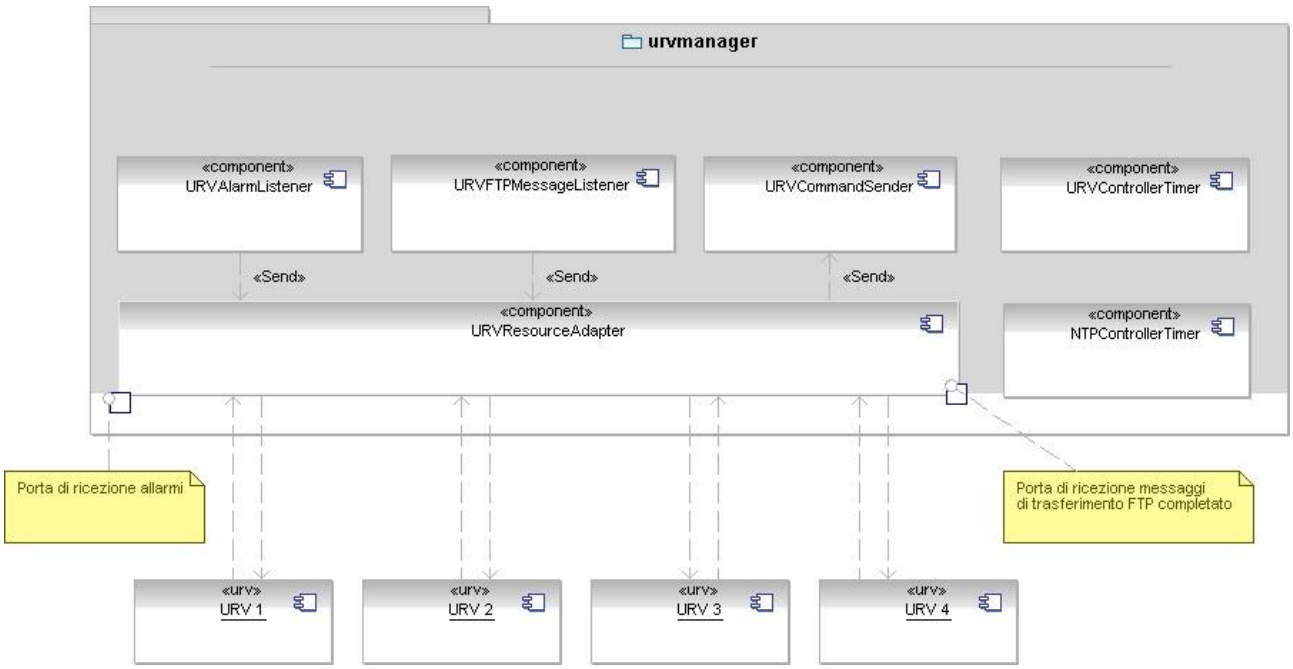

Figura 6 - URVManager, Diagramma dei componenti

La comunicazione tra UEL ed URVs avviene tramite un componente denominato URV Resource Adapter. L'URV Resource Adapter è l'implementazione di un componente specifico dell'architettura J2EE, il Resource Adapter, il quale si occupa di interfacciare l'applicazione con le sorgenti dati esterne (in questo caso le URVs). Il compito principale è quello di astrarre i dettagli della comunicazione con le sorgenti dati esterne, fornendo ai componenti che lo utilizzano un'interfaccia standard per l'accesso ai dati.

L'URV Resource Adapter permette la comunicazione con le URV in entrambi i sensi, sia per l'invio dei comandi che per la ricezione dei messaggi. Esistono due componenti distinti che si occupano della ricezione dei messaggi dalla URV, uno per ognuno dei due canali di comunicazione previsti (allarmi e messaggi di FTP completato). I due componenti utilizzati sono l'FTPMessageListener, che si occupa di gestire i messaggi di *trasferimento FTP completato* e l'AlarmListener, che si occupa di ricevere i messaggi di allarme. Nel seguito, ove necessario, verrà utilizzato il termine Listener per riferirsi indistintamente ad uno dei due componenti.

Entrambi i Listener sono degli MDB (message-driven-bean), quindi dei componenti che permettono la ricezione asincrona di messaggi. Il paragrafo successivo descrive il funzionamento del componente URV Resource Adapter.

5.7.1 URV Resource Adapter

La figura seguente descrive in dettaglio le classi che compongono l'URV Resource Adapter.

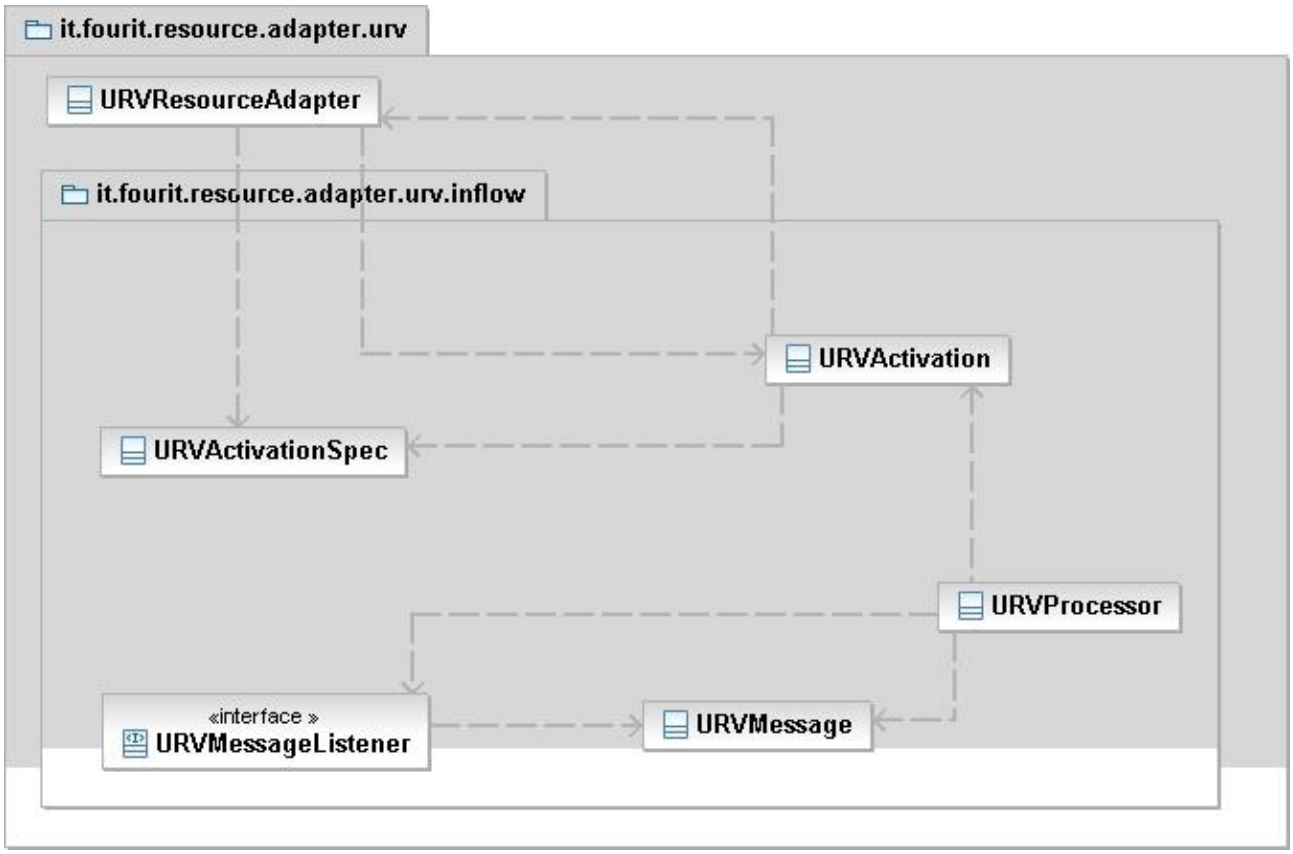

Figura 7 - Diagramma delle classi dell'URV Resource Adapter

I Listener, per poter utilizzare l'URV Resource Adapter, devono implementare l'interfaccia URVMessageListener, fornita dall'URV Resource Adapter stesso. Tale interfaccia definisce un metodo pubblico che verrà poi utilizzato dal Resource Adapter per recapitare i messaggi inviati dalle URV ai Listener stessi.

All'avvio della UEL, i due Listener si registrano presso il Resource Adapter. Durante la fase di registrazione viene inviata al Resource Adapter il numero di porta sulla quale si vogliono ricevere i messaggi; l'AlarmListener specificherà la porta riservata per gli allarmi, mentre l'FTPMessageListener specificherà la porta riservata per i messaggi di *trasferimento FTP completato*. Il Resource Adapter, per ognuno dei due ascoltatori, creerà uno specifico thread (URVActivation), il quale si occuperà di mettersi in ascolto di messaggi inviati dalle URV dalla specifica porta configurata.

La figura seguente descrive i passi eseguiti durante la fase di registrazione di un Listener (nello specifico, l'AlarmListener) presso il Resource Adapter. Identiche considerazioni possono essere fatte

per quel che riguarda lo FTPMessageListener.

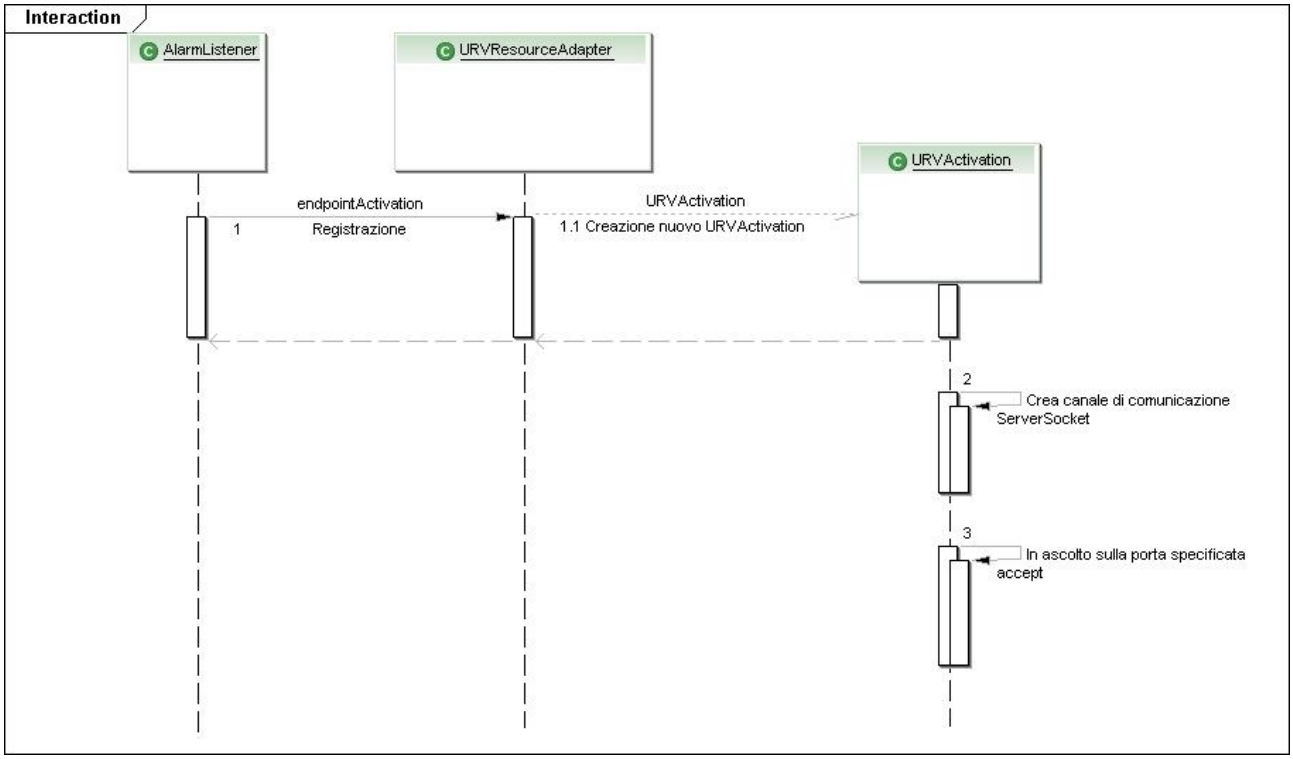

Figura 8 - Diagramma di sequenza della registrazione di un Listener presso l'URV Resource Adapter

Al momento della creazione dell'AlarmListener, l'Application Server si occupa di registrare lo stesso presso il Resource Adapter, specificando la porta dalla quale l'AlarmListener vuole ricevere messaggi di allarme. Quando il Resource Adapter riceve la richiesta di registrazione dell'AlarmListener, istanzia un nuovo thread (URVActivation) a cui delega le operazioni necessarie per la creazione del canale di comunicazione (TCP) e per la ricezione dei messaggi dalle URVs sulla porta specificata. L'URVActivation rimarrà quindi in ascolto sulla porta fino all'arrivo di un nuovo messaggio.

All'arrivo di un nuovo messaggio da parte di una URV, l'URVActivation in ascolto sulla porta di provenienza del messaggio stesso, si occuperà di creare un nuovo thread (URVProcessor) per la gestione del messaggio ricevuto. In questo modo, l'URVActivation sarà in grado di ricevere subito nuovi messaggi inviati dalle altre URVs.

L'URVProcessor si occupa realmente di ricevere il messaggio dalle URVs, di convertirlo nel formato concordato con i Listener e di inviarlo a questi ultimi. Una volta terminata l'elaborazione da parte del corrispettivo Listener, l'URVProcessor si occuperà di inviare la risposta generata da questo alla URV, per confermare l'avvenuta ricezione del messaggio.

La ricezione di un nuovo messaggio da parte di una delle URV scatena lo scenario descritto nella figura seguente.

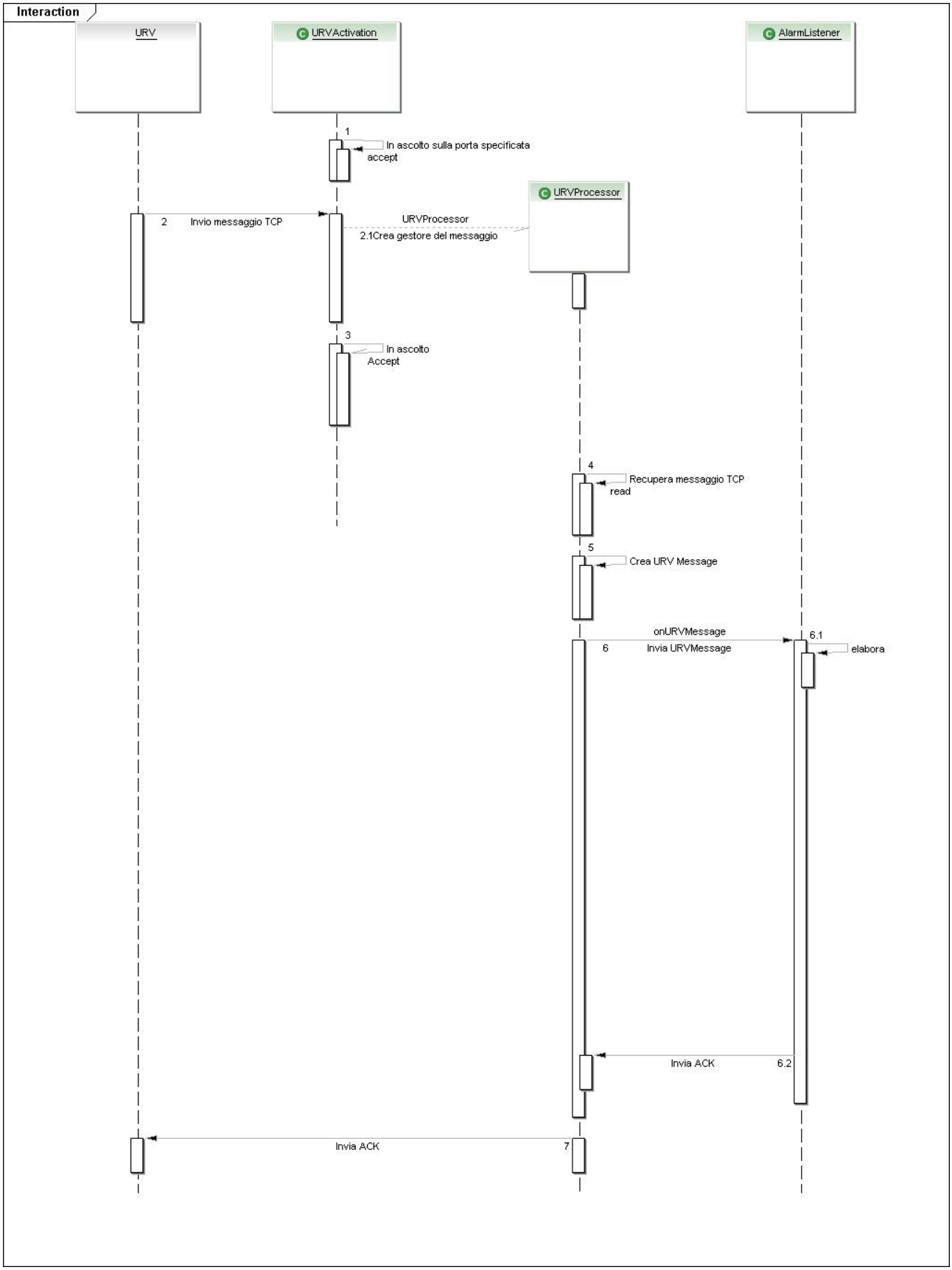

Figura 9 - Diagramma di sequenza relativo all'arrivo di un nuovo messaggio dalle URVs

All'arrivo di un nuovo messaggio, l'URVActivation, che si trova in ascolto sulla porta specifica, delega la ricezione del messaggio stesso ad un nuovo oggetto, l'URVProcessor; in questo modo può tornare subito disponibile per la ricezione di nuovi eventuali messaggi, inviati dalle altre URV collegate alla UEL.

L'URVProcessor è quindi il responsabile dell'elaborazione del messaggio ricevuto. Esso si occupa di elaborare il flusso dati inviato su canale TCP, impacchettare i dati in un oggetto URVMessage ed inviare il messaggio all'AlarmListener. Questi, dopo le dovute elaborazioni restituirà un messaggio di ACK per informare l'avvenuta elaborazione del messaggio. Sarà ancora l'URVProcessor ad incaricarsi di trasformare il messaggio di ACK in un flusso di dati per l'invio verso le URV tramite TCP. L'URVProcessor chiude quindi la connessione TCP aperta.

Nel caso si verifichi un errore durante l'elaborazione del messaggio da parte dell'AlarmListener, questi genererà un'eccezione per informare il Resource Adapter dell'errore occorso. Il Resource Adapter, tramite l'oggetto URVProcessor, chiuderà la connessione TCP aperta verso la URV senza inviare il messaggio di ACK. In questo caso, sarà la URV a rinviare il messaggio dopo non aver ricevuto il messaggio di ACK entro un tempo di timeout prefissato (dell'ordine dei millisecondi). Il fatto che il tempo di timeout sia molto basso, ha escluso la possibilità che la UEL inviasse messaggi di errore verso la URV.

Lo scenario sopra descritto è identico nel caso in cui si stiano ricevendo messaggi di *trasferimento FTP completato*. La differenza sostanziale sarà nell'elaborazione del messaggio effettuata dal FTPMessageListener, che saranno ovviamente differenti da quelle effettuate dall'AlarmListener, nonché della porta di ascolto. Nei paragrafi successivi vengono descritti i passi dell'elaborazione effettuati dai due Listener.

5.7.2 Ricezione di un allarme

Il componente che si occupa di elaborare i messaggi di allarme inviati dalle URVs è l'AlarmListener. Come già detto, esso utilizza l'URV Resource Adapter per ricevere i messaggi, presso il quale si registra specificando la porta da cui vuole ricevere i messaggi (1001, nel caso di seconda istanza 1002). L'AlarmListener riceve dall'URV Resource Adapter il messaggio d'allarme impacchettato in un opportuno oggetto (URVMessage), inoltre riceve l'indirizzo IP della URV che ha inviato il messaggio stesso. Tramite tale indirizzo IP è in grado di identificare univocamente la URV che ha generato l'allarme. Le operazioni svolte dall'AlarmListener sono le seguenti:

- Identificare, tramite l'indirizzo IP, la URV che ha generato l'allarme;
- Verificare se si tratta di un messaggio duplicato;
- Salvare l'allarme generato su database;
- Aggiornare lo stato della URV sulla base dell'allarme ricevuto;
- Nel caso la UEL sia stata attivata, inviare un messaggio al PRC Service che si occuperà di inviare l'allarme al Server Centrale (vedere paragrafi successivi).
- Restituire al Resource Adapter la risposta(ACK) da inviare alla URV per comunicare l'avvenuta ricezione ed elaborazione del messaggio stesso.

Sarà compito del Resource Adapter convertire ed inviare l'ACK di risposta alla URV.

L'AlarmListener si occupa quindi semplicemente di rendere persistente il messaggio ricevuto, di aggiornare lo stato della URV e quindi di delegare ad un altro servizio il compito di inviare l'allarme verso il Server Centrale, in modalità asincrona. Nel caso in cui si verifichi un errore durante l'elaborazione del messaggio, l'AlarmListener genererà un'eccezione che verrà inviata al sottostante Resource Adapter. Questi, come già visto, si occuperà di chiudere la connessione con la URV per permettere il rinvio del messaggio stesso.

L'AlarmListener controlla, per ogni messaggio ricevuto, che non si tratti di un messaggio duplicato (messaggio con identificativo identico); tale scenario può verificarsi nel caso in cui vada perso il messaggio di ACK inviato dall'AlarmListener verso la URV, perdita che causa il rinvio dello stesso da parte della URV.

5.7.3 Ricezione di un messaggio di trasferimento FTP completato

E' presente un componente dedicato che si occupa di elaborare i messaggi di *trasferimento FTP completato* inviati dalle URV per comunicare la presenza di una nuova immagine nell'alberatura del server FTP. Tale componente, denominato FTPMessageListener, utilizza, come l'AlarmListener, l'URV Resource Adapter per ricevere i messaggi inviati dalle URV; la differenza tra i due sta nella porta specificata durante la fase di registrazione dello FTPMessageListener (1003, nel caso di seconda istanza 1004). Per ogni messaggio di trasferimento FTP completato ricevuto, l'FTPMessageListener si occupa di eseguire le seguenti elaborazioni:

- Identificare, tramite l'indirizzo IP, la URV che ha inviato il messaggio;
- Verificare se si tratta di un messaggio duplicato (messaggio con identificativo identico);
- Recuperare il numero di sequenza contenuto nel messaggio stesso;

- Verificare che il messaggio ricevuto non sia fuori sequenza;
- Salvare il messaggio ricevuto in modo persistente sul database;
- Inviare un messaggio asincrono al PCaptureService, il quale si occuperà delle fasi successive dell'elaborazione del messaggio(transito) ricevuto.
- Restituire al Resource Adapter la risposta(ACK) da inviare alla URV per comunicare l'avvenuta ricezione ed elaborazione del messaggio stesso.

L'FTPMessageListener si occupa quindi di svolgere solamente delle semplici operazioni per il salvataggio del messaggio ricevuto e per il controllo del numero sequenza. Le elaborazioni successive, come ad esempio il parsing dell'header JPEG, il controllo dei duplicati, etc., vengono delegati ad un altro servizio (PCaptureService); il motivo risiede nel fatto che tali elaborazioni risultano pesanti dal punto di vista computazionale, quindi devono essere svolte successivamente alla ricezione del messaggio stesso, altrimenti si terrebbe sospesa la URV in attesa dell'ACK di risposta per un tempo inaccettabile. In tale modo, invece, vengono svolte operazioni veloci e viene subito inviato l'ACK alla URV. Il fatto di rendere persistente il messaggio ricevuto, garantisce che tale messaggio non possa andare più perso in alcun caso.

Il controllo del numero di sequenza permette di verificare che l'ordine con cui vengono inviati i messaggi dalle URVs verso la UEL, non venga alterato; inoltre permette di verificare che nessun messaggio vada perso. Nel caso in cui si riscontri un'anomalia nel numero di sequenza, viene generato un opportuno allarme per segnalare agli operatori ASPI l'errore occorso.

L'invio del messaggio verso il PCaptureService avviene in modalità asincrona ed utilizza una coda in memoria; all'interno del PCapture Service sarà presente un componente dedicato in ascolto su tale porta. In questo modo, il PCapture Service e l'URVManager, nello specifico l'FTPMessageListener, risultano disaccoppiati completamente.

5.7.4 Invio di un comando alle URVs

L'ultimo componente presente all'interno dell'URVManager che si occupa della comunicazione con le URVs è quello responsabile dell'invio di comandi verso le URVs. Tale componente (EJB di tipo Session), denominato CommandSender, utilizza ancora una volta l'URV Resource Adapter per comunicare con le URV, delegando a questo la trasformazione del comando in un messaggio atto alla trasmissione su canale TCP. Il CommandSender espone un'interfaccia pubblica per l'invio alle URV di tutti quei comandi previsti dalla specifica del protocollo Tattile, come ad esempio, l'avvio, l'arresto o il reset. Tali comandi saranno invocati dai servizi che ne avessero necessità.

Per ogni comando inviato, il CommandSender si occupa di aggiornare lo stato della URV o delle URVs interessate, quindi delega l'invio del messaggio stesso al Resource Adapter. A questo punto si mette in ascolto della risposta della URV. Alla ricezione della risposta, informa il chiamante della riuscita dell'invio del comando.

In caso non riceva risposta entro un periodo prefissato (timeout), verrà generata una eccezione che sarà propagata al servizio chiamante, che si occuperà di gestire l'errore occorso.

La figura seguente descrive la sequenza dei passi eseguiti durante l'invio di un comando ad una URV, cosi come descritto sopra.

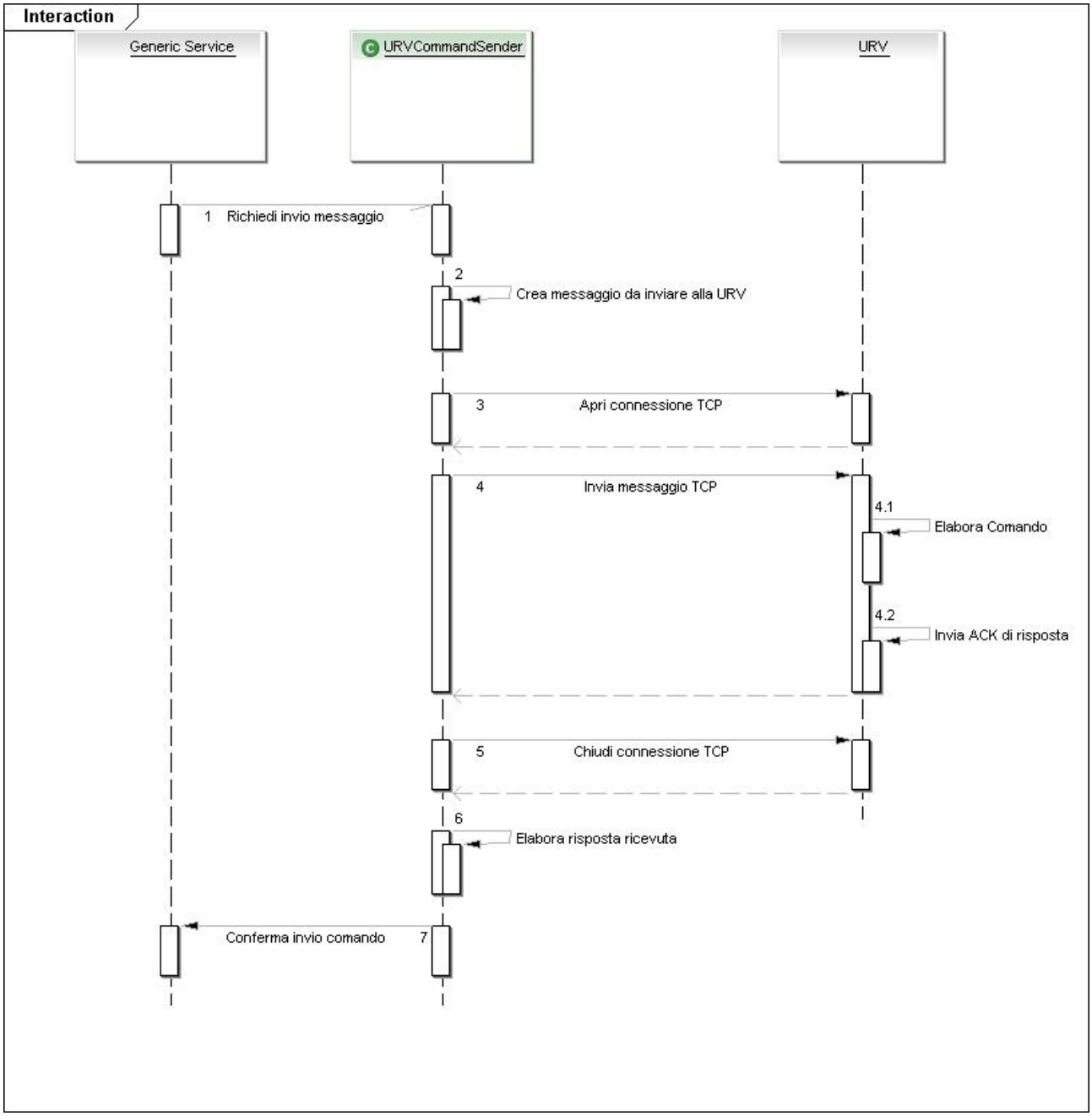

Figura 10 - Diagramma di sequenza dell'invio di un comando verso una URV

5.7.5 Verifica del collegamento con le URVs

All'interno del servizio URVManager è presente un componente specifico che si occupa di verificare lo stato del collegamento tra UEL ed URVs. Tale componente, denominato URVControllerTimer, verifica periodicamente, per ogni URV, l'intervallo di tempo trascorso tra l'ultimo messaggio ricevuto e l'ora attuale. Se tale intervallo risultasse maggiore di un valore

prefissato e configurabile, provvede a disabilitare la URV stessa tramite l'utilizzo di un firewall. In pratica la URV viene disabilitata se non invia per un periodo di tempo i messaggi di vita alla UEL.

Nel caso la UEL sia stata attivata, URVControllerTimer invia un messaggio al servizio di controllo remoto per permettere di segnalare la disabilitazione della URV al Server Centrale.

L'URVControllerTimer è un componente di tipo EJB Timer.

5.7.6 Verifica del funzionamento dell'NTP Server

L'ultimo componente che si trova all'interno dell'URVManager si occupa di verificare ciclicamente il corretto funzionamento dell'NTP Server. L'NTP Server si occupa di effettuare la sincronizzazione tra la UEL e le URVs collegate. Tale componente, denominato NTPControllerTimer, come il precedente, è un componente di tipo EJB Timer, quindi viene schedulato alla partenza del servizio URVManager.

Nel caso in cui l'NTPControllerTimer verifichi un malfunzionamento del server NTP, provvede ad inviare una allarme al servizio di controllo remoto, nel caso in cui la UEL risulti già attivata. L'errore occorso viene comunque tracciato su database in modo persistente.

Se il server NTP risulta non funzionante, i transiti raccolti dalla UEL non vengono collezionati per fini di verifica violazioni, in quanto non è possibile verificare con certezza l'orario del passaggio dei veicoli. Tale filtro viene eseguito anche qualora fosse attivo l'allarme di mancata sincronizzazione per la URV che ha inviato il transito.

5.8 PMManager

Il PMManager è il servizio che si occupa di dialogare con il demone EnDetector ospitato all'interno della UEL. Offre una interfaccia pubblica agli altri servizi per l'invio di comandi verso l'EnDetector e svolge funzioni di controllo sul funzionamento dell'EnDetector stesso.

Il PMManager è un servizio che viene avviato durante lo start della UEL stessa (che avviene, come già detto, attraverso il servizio principale UELManager) e resta attivo per tutto il funzionamento della UEL, svolgendo completamente le proprie funzioni, indipendentemente dallo stato operativo in cui si trova la UEL. In particolare, le funzioni svolte dal servizio sono le seguenti:

- Verificare lo stato di funzionamento del processo EnDetector;
- Inviare i transiti da elaborare a EnDetector e gestire le risposte;
- Permettere lo start e lo stop di EnDetector;
- Verificare la versione di EnDetector.

L'EnDetector è una applicazione esterna alla UEL, con la quale questa si interfaccia via HTTP. Tale applicazione riceve in input il nome file di un immagine e per ogni immagine tenta il riconoscimento delle stringhe identificative, che sono restituite in risposta.

Il PMManager, in caso di risposta positiva da parte di EnDetector, modifica l'header jpeg dell'immagine, in modo da aggiungerci la stringa PM ricavata e le altre informazioni a corredo, e salva la stessa informazione sul DB.

5.9 Comunicazione UEL – Server Centrale

Prima di descrivere i servizi che si occupano di dialogare con il Server Centrale per l'invio e la ricezione dei comandi, viene trattato in questo paragrafo il sottosistema di messaggistica utilizzato per lo scambio di tali messaggi.

La comunicazione tra UEL e Server Centrale avviene per mezzo del servizio JMS (Java Message Service), il quale fornisce un protocollo per lo scambio dei messaggi; tali messaggi vengono poi scambiati tramite connessioni TCP sicure. Il servizio JMS si pone dunque sopra i protocolli TCP/SSL, sfruttandone appieno le possibilità.

Lo scambio di messaggi nel servizio JMS avviene tramite l'utilizzo di code; i vari attori della comunicazione scrivono e leggono direttamente dalla coda, senza comunicazione diretta tra di loro. Le code risiedono in un componente denominato JMS Provider.

Il JMS Provider è il componente principale del sistema di messaggistica, il quale si occupa di gestire in memoria tutte le destinazioni previste, di ricevere i messaggi inviati dai diversi client e di notificare i messaggi stessi ai destinatari indicati. All'interno della tecnologia JMS, con il termine destinazione si indica una particolare coda di messaggi, all'interno della quale i client possono inserire i messaggi da inviare.

La grande potenzialità del servizio JMS sta nella possibilità di integrazione con la tecnologia EJB. Nella specifica EJB, è previsto un componente specifico, denominato message-driven-bean (MDB), che si occupa di ricevere i messaggi provenienti da una determinata coda. Un MDB, una volta avviato, si registra presso il JMSProvider su di una specifica coda, specificando eventualmente dei parametri di selezione dei messaggi che intende ricevere; ogni volta che un nuovo messaggio viene inviato sulla coda, il JMSProvider provvede ad inviare il messaggio ricevuto all'MDB; solitamente il JMSProvider e gli MDB risiedono sulla stessa macchina, quindi la consegna del messaggio non prevede l'instaurarsi di connessioni di rete.

Gli MDB, così come ogni EJB, vengono gestiti internamente dall'Application Server ove risiedono in uno specifico pool; quando un nuovo messaggio deve essere consegnato ad un MDB, l'Application Server seleziona un'istanza tra quelle libere presenti nel pool associato a quel particolare MDB, quindi provvede a consegnare il messaggio. In questo modo, non è necessario creare da zero il componente per ogni messaggio ricevuto. Nel caso ci fosse un aumento dei messaggi da scambiare, l'Application Server provvederà ad aumentare il numero di istanze presenti nel pool, per poter gestire il maggior carico derivato. Nel caso di ritorno ad attività normale, sempre l'Application Server provvederà ad eliminare le istanze in eccesso dal pool. Viene sottolineato che il pool si riferisce solamente alla gestione di istanze di componenti MDB, mentre non viene effettuato nessun pooling delle connessioni TCP.

L'invio di un messaggio viene solitamente riservato ad un componente EJB di tipo Session Stateless. Un EJB Session espone un insieme di metodi pubblici che possono essere utilizzati per svolgere particolari operazioni; anche questo tipo di componente viene gestito in un pool di EJB, per permettere l'esecuzione parallela delle stesse operazioni in caso di necessità. Un EJB Session Stateless, non mantiene informazioni sul proprio stato tra un'invocazione e l'altra; in questo caso, una volta terminata l'esecuzione di uno specifico metodo da parte di un EJB, questi torna immediatamente disponibile nel pool di istanze e può essere utilizzato da altri componenti per svolgere le proprie funzioni.

Nel caso di invio di un messaggio JMS, l'EJB Session Stateless si occupa di instaurare una

connessione TCP/SSL con il JMS Provider, quindi invia il messaggio JMS e rimane in attesa di un ACK di risposta. Alla ricezione dell'ACK chiude la connessione. Nei paragrafi successivi, dove verrà descritta l'infrastruttura utilizzata, verranno dati maggiori dettagli.

5.9.1 Infrastruttura di messaggistica

Per gestire la comunicazione UEL – Server Centrale, viene utilizzata un'infrastruttura di messaggistica che si pone l'obiettivo di mascherare ai servizi superiori i dettagli della comunicazione JMS, permettendo di concentrarsi solamente sulla logica del servizio stesso. L'infrastruttura di messaggistica non costituisce un layer aggiuntivo nella comunicazione, ma mette a disposizione delle classi astratte che i componenti atti alla comunicazione dovranno implementare; in questo modo si nascondono i dettagli della comunicazione senza aggiungere strati onerosi al sistema.

Ad un livello superiore, i servizi che si occupano della logica di business, dovranno utilizzare tali classi astratte per l'invio di un messaggio. Tale messaggio verrà poi convertito in un messaggio JMS atto all'invio sul canale di comunicazione, in maniera trasparente ai servizi superiori.

La figura seguente descrive le classi presenti nell'infrastruttura di messaggistica utilizzata:

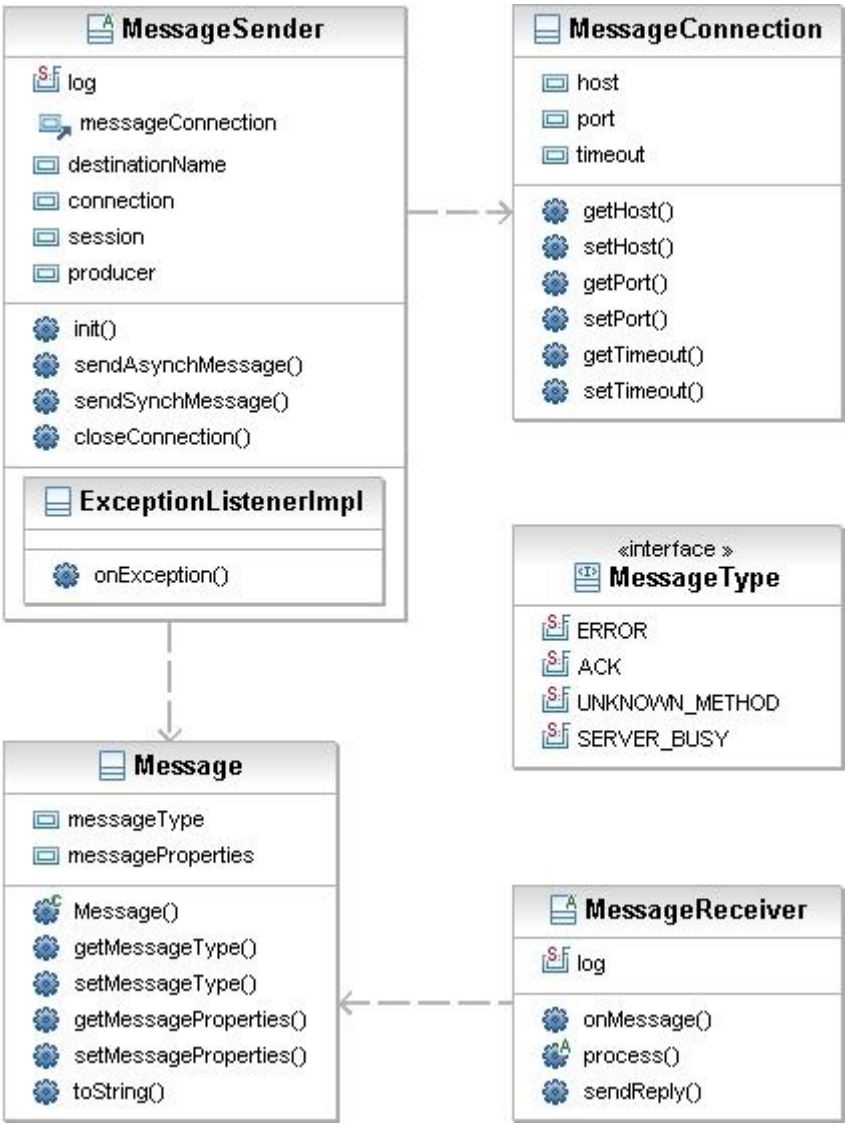

Figura 11 - Diagramma delle classi del sottosistema di messaggistica

La classe Message descrive il formato dei messaggi che vengono scambiati tra UEL e Server Centrale. Un generico messaggio è costituito da due campi differenti:

- il tipo del messaggio (messageType), indica il contenuto del messaggio stesso;
- un insieme di proprietà (messageProperties), definisce i parametri necessari allo specifico tipo di messaggio;

Ad esempio, nel caso la UEL debba inviare un messaggio di allarme verso il Server Centrale, creerà un oggetto Message con messageType uguale a SEND_ALERT, e nelle proprietà verranno definiti il tipo di allarme, la URV che lo ha generato, etc.

In questo modo, UEL e Server Centrale utilizzano sempre lo stesso oggetto per comunicare, senza dover condividere decine di oggetti differenti, uno per ogni tipo di messaggio; l'unica

informazione che dovranno conoscere UEL e Server Centrale è costituita dai tipi di messaggio che possono scambiarsi e delle relative proprietà associate ad ogni messaggio. Nel caso il destinatario del messaggio non trovi una delle proprietà previste per uno specifico messaggio, dovrà segnalare con un errore al mittente l'anomalia riscontrata. Ovviamente tali anomalie non saranno più presenti dopo le fasi di test del sistema (protocolli ben definiti tra UEL e Server Centrale).

Un generico scenario di comunicazione prevede sempre che uno tra UEL e Server Centrale inizi la comunicazione, mentre l'altro si occupa di elaborare la richiesta e di fornire eventualmente una risposta. All'interno di uno specifico canale di comunicazione (capture, rc, statistics, activation) vengono definiti i tipi di messaggio che la UEL può inviare e quelli che può ricevere; fare riferimento al documento dei requisiti per la definizione di tali protocolli di comunicazione. I tipi di messaggio vengono definiti all'interno di specifiche interfacce, le quali devono estendere l'interfaccia base MessageType, la quale definisce i tipi di messaggio comuni (ACK, ERROR, etc.) utilizzati durante la comunicazione.

Ogni servizio, inoltre, definirà due componenti, uno per l'invio di un messaggio ed uno per la ricezione. Il componente che si occuperà di inviare un messaggio, nel seguito ServiceSender, che sarà del tipo EJB Session, estenderà la classe MessageSender, la quale fornisce un'astrazione per l'invio di un messaggio JMS. La classe MessageSender definisce due metodi distinti per l'invio di un messaggio sincrono od asincrono. In base alle necessità dello specifico messaggio, potrà essere utilizzato uno dei due tipi di trasmissione.

La classe MessageSender si occupa di creare la connessione con il Provider JMS, utilizzando i parametri contenuti nella classe MessageConnection inizializzata dal ServiceSender. La classe MessageConnection contiene l'informazione relativa all'indirizzo IP del Provider JMS, della porta di ascolto , il timeout di connessione impostato e il numero di tentativi di rinvio del messaggio stesso in caso di errore. Tali parametri dipendono strettamente dal servizio, e nel caso della UEL, vengono inviati dal Server Centrale durante la fase di attivazione. Ogni servizio quindi avrà impostato i propri parametri di connessione.

La figura seguente mostra lo scenario relativo all'invio di un messaggio in modalità asincrona.

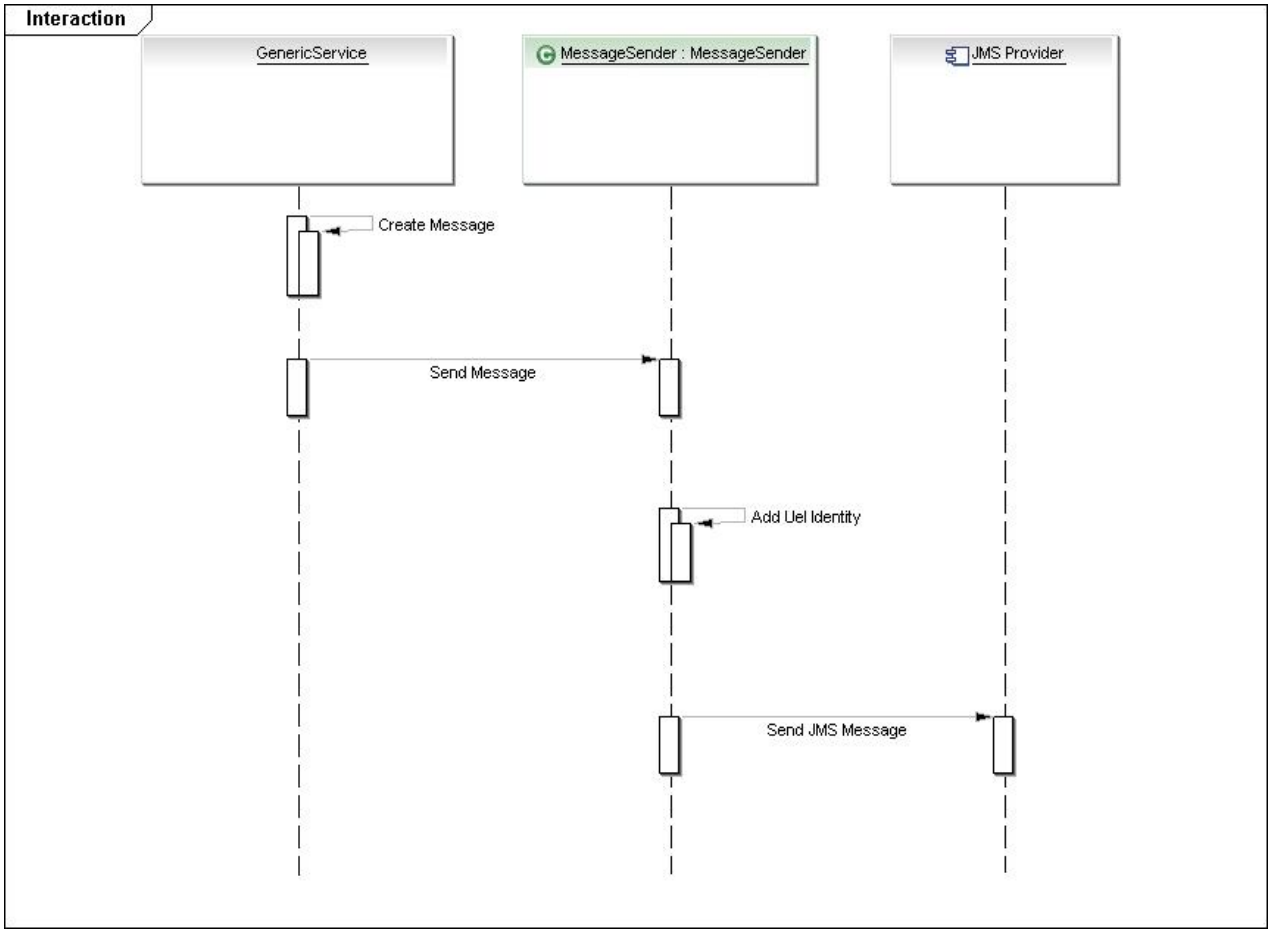

Figura 12 - Comunicazione asincrona

I passi eseguiti dal MessageSender sono i seguenti:

- Recupera i parametri della connessione;
- Apre una nuova connessione con il Provider JMS;
- Crea un messaggio JMS adatto alla trasmissione a partire dall'oggetto Message;
- Invia il messaggio JMS al Provider JMS;
- Chiude la connessione con il Provider JMS.

Nel caso in cui il MessageSender non sia in grado di inviare il messaggio, tenterà per un numero di volte pari al parametro contenuto nella classe MessageConnection. Nel caso fallisca per tutti gli n tentativi, viene lanciata un'eccezione. Sarà compito del servizio che sta utilizzando il sottosistema di messaggistica occuparsi di implementare delle politiche di rinvio del messaggio stesso. Tali politiche verranno descritte nei successivi paragrafi dove si entrerà nello specifico dei servizi stessi.

Lo scenario sincrono differisce dal precedente per il fatto che il MessageSender aspetta per un periodo di timeout, sempre contenuto nell'oggetto MessageConnection, una risposta dal destinatario

del messaggio. In caso contrario tenterà il rinvio come nello scenario asincrono. In caso di risposta, questa verrà inviata al servizio per la corretta elaborazione della stessa. La figura seguente descrive lo scenario sincrono.

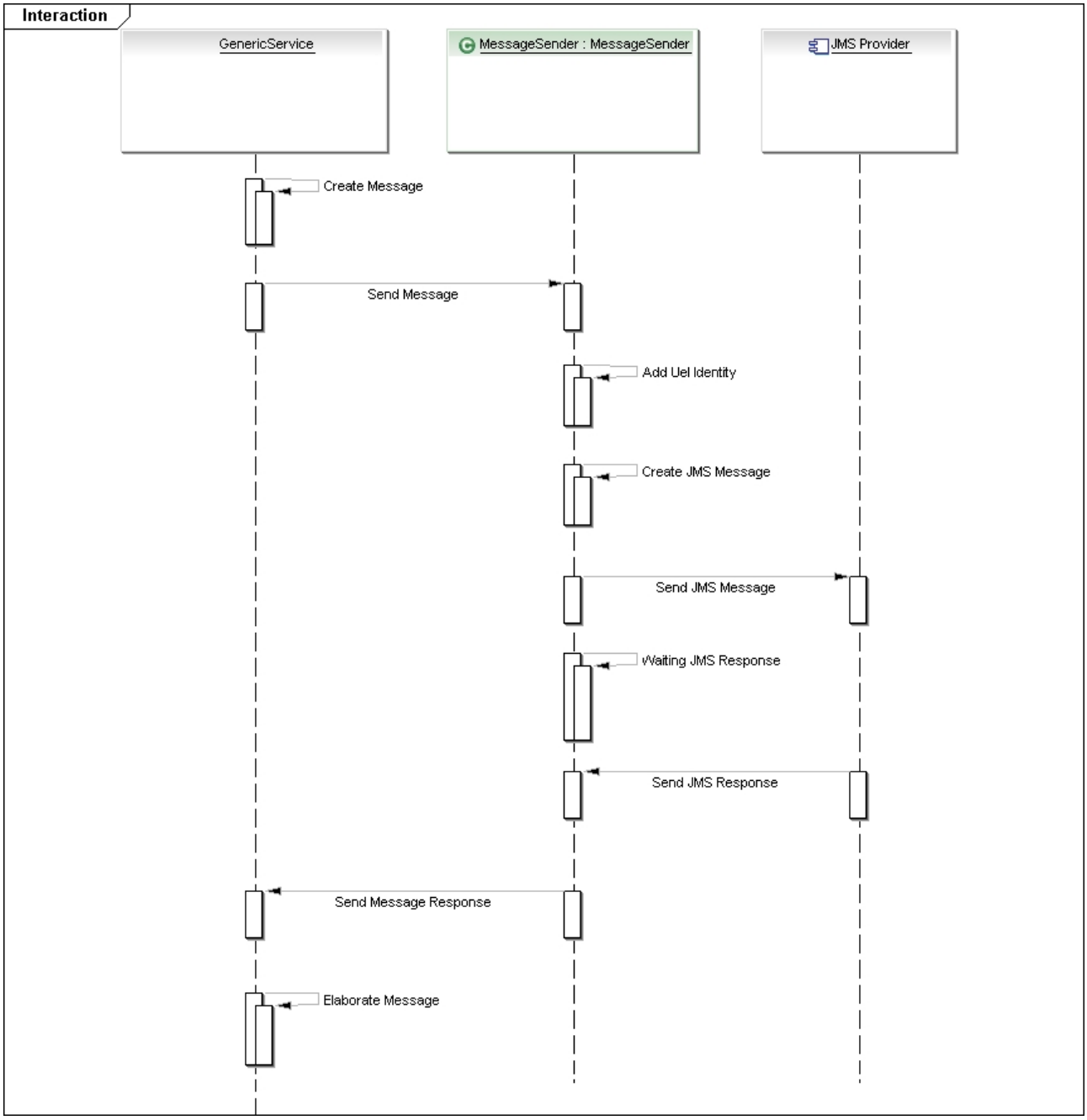

Figura 13 - Comunicazione sincrona

La ricezione dei messaggi è affidata al componente MessageReceiver. Tale componente, un MDB, si occupa di gestire la ricezione di un messaggio JMS in arrivo su una coda specifica. Il servizio che intenda ricevere messaggi, dovrà utilizzare un componente che estenda il MessageReceiver e che

implementi il metodo process; tale metodo sarà indipendente da JMS e si occuperà dell'elaborazione logica del messaggio ricevuto.

Nel caso in cui si verifichi un qualsiasi problema durante l'invio o la ricezione di un messaggio, il sottosistema di messaggistica lancerà un'eccezione di tipo MessageException per notificare l'errore avvenuto.

Il sottosistema di messaggistica farà riferimento al SecurityManager per recuperare il certificato di autenticazione da utilizzare per instaurare una connessione sicura con il JMS Provider.

Nei paragrafi successivi, verranno descritti i servizi che utilizzano il sottosistema di messaggistica, descrivendo le politiche di rinvio implementate.

5.10 Gestione delle transazioni di controllo remoto: PRC Service

Il PRC Service è il servizio che si occupa di gestire la comunicazione con il Server Centrale per quello che riguarda le transazioni di controllo remoto. Esso viene avviato solamente a seguito della fase di attivazione della UEL e svolge completamente le proprie funzionalità fino al momento in cui la UEL torna nello stato operativo "Statistiche in locale". Le operazioni svolte dal servizio sono le seguenti:

- Ricezione dei comandi di controllo remoto dal Server Centrale;
- Invio degli allarmi generati dalle URVs e dalla UEL verso il Server Centrale

Il PRC Service dialoga con il Server Centrale attraverso l'infrastruttura di messaggistica; sul Server Centrale è presente un componente duale del PRC Service che si occupa di interloquire con questo (SRC Service).

Il PRC Service comunica con gli altri servizi attraverso una coda locale, sulla quale gli altri servizi inviano gli allarmi generati che intendono inviare al Server Centrale.

La figura seguente descrive l'architettura logica interna del servizio PRC.

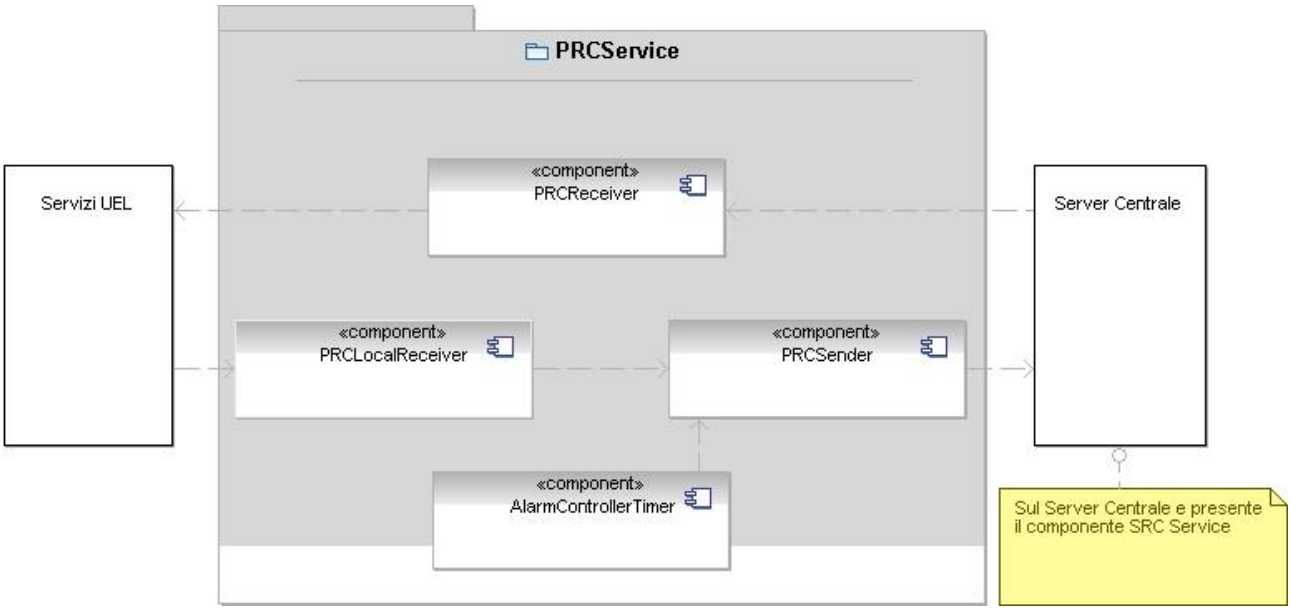

Figura 14 - Architettura logica del servizio PRC

Il PRC Service è composto essenzialmente da quattro distinti componenti, due dei quali utilizzati per la comunicazione con il Server Centrale (vedi paragrafo sul sistema di messaggistica), uno per ricevere le richieste inviate dagli altri servizi della UEL (come l'URV Manager) e l'ultimo per verificare la presenza di allarmi da rinviare in caso di fallimento precedente.

Il componente che si occupa della ricezione dei messaggi dal Server Centrale è il PRCReceiver. Tale componente estende la classe MessageReceiver ed una volta avviato, si pone in ascolto dei messaggi in arrivo sulla coda dedicata al servizio di controllo remoto. Su tale coda il Server Centrale invia i comandi di controllo remoto, come ad esempio lo start delle URV, il test della rete di campo, etc. Alla ricezione di un comando da parte del Server Centrale, il PRCReceiver si occupa di svolgere le attività legate al comando stesso, eventualmente richiamando il servizio URVManager. La figura seguente descrive i passi eseguiti durante la ricezione di un comando da parte del Server Centrale:

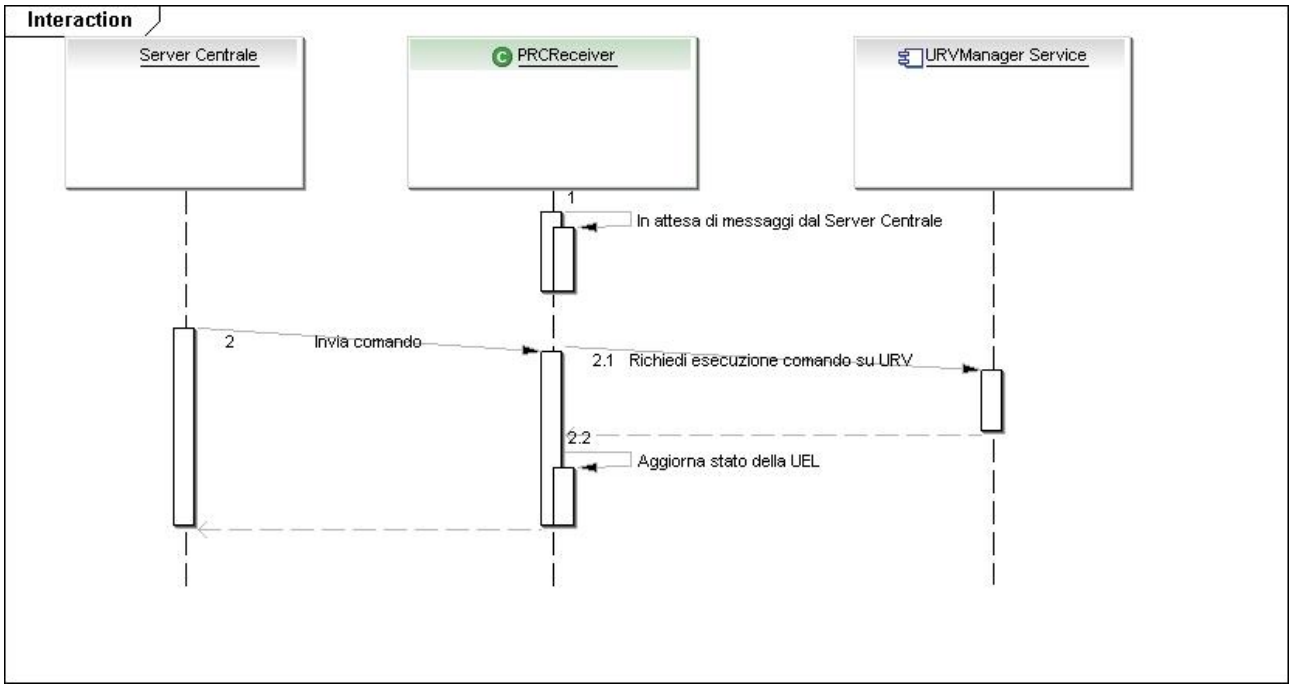

Figura 15 - Ricezione di un messaggio di controllo remoto, diagramma di sequenza

Quando il Server Centrale invia un messaggio di controllo remoto, questo viene consegnato al PRCReceiver, il quale si occupa di gestire l'elaborazione del messaggio stesso. In particolare, ove necessario, verrà richiesto al servizio URVManager di inviare un comando ad una o più URVs, verrà aggiornato lo stato dello UEL, ed infine verrà inviata la risposta al Server Centrale. Nei paragrafi successivi verranno descritti tutti i comandi inviati dal Server Centrale verso la UEL.

Il componente che si occupa di ricevere gli allarmi generati dalla UEL o dalle URVs dagli altri servizi è il PRCLocalReceiver. Tale componente è in ascolto su una coda locale (coda in memoria), sulla quale gli altri servizi inviano i vari allarmi generati per l'invio al Server Centrale. Il PRCLocalReceiver si occupa di ricevere tali messaggi, quindi delega al componente PRCSender il compito di inviarli al Server Centrale. Il PRCLocalReceiver si occupa quindi solamente di smistare gli allarmi in ingresso al PRC Service.

Il PRCSender si occupa effettivamente di inviare i messaggi di allarme al Server Centrale, estendendo la classe MessageSender. Il PRCSender recupera i parametri di connessione propri del PRCSender (inviati dal Server Centrale durante la fase di attivazione), quindi delega all'infrastruttura di messaggistica il compito della consegna del messaggio al Server Centrale. Una volta consegnato l'allarme, il corrispettivo record presente sulla base dati viene eliminato. La figura seguente descrive la sequenza dei passi eseguiti durante l'invio di un allarme.

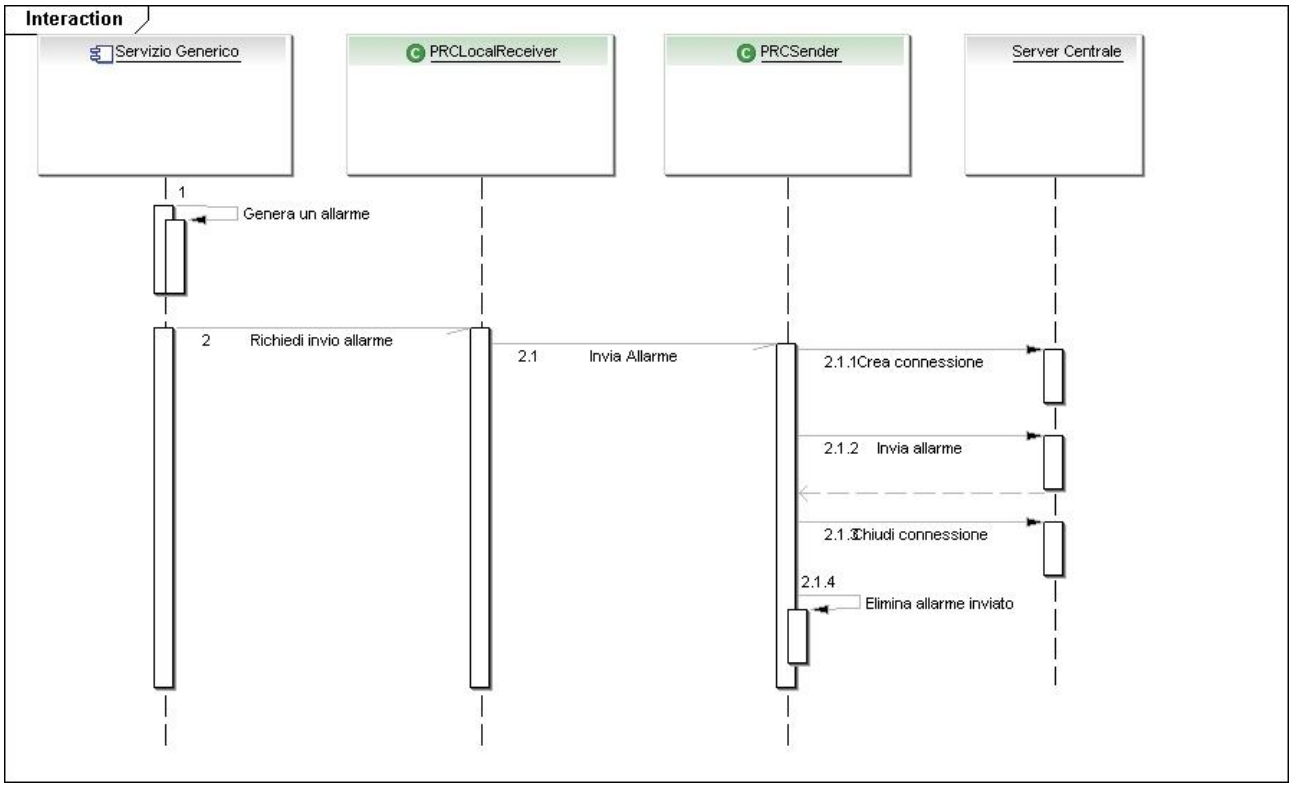

Figura 16 - Invio di un allarme, diagramma di sequenza

Nel caso si verifichi un qualunque errore che non permetta di inviare l'allarme al Server Centrale entro gli n tentativi previsti (vedi paragrafo messaggistica), il PRCSender marcherà l'allarme stesso per essere re-inviato successivamente. Tali allarmi verranno rinviati successivamente dopo essere stati recuperati dal componente AlarmControlTimer, componente che verifica periodicamente la presenza sul database di eventuali allarmi non ancora inviati al Server Centrale. In dettaglio vengono recuperati i primi n allarmi in ordine temporale, per tentarne il rinvio verso il Server Centrale.

Il valore n, impostato dal Server Centrale durante la fase di attivazione, evita il sovraccarico del Server in caso di ripristino a seguito di una qualsiasi interruzione della comunicazione. Tramite questo controllo, verranno inviati solo n allarmi per volta, schedulando quindi l'invio nel tempo. L'AlarmControlTimer, inoltre, permette di recuperare anche gli allarmi generati dagli altri servizi che

per qualunque motivo non sono arrivati precedentemente al PRC Service.

L'AlarmControlTimer si occupa inoltre, di re-inviare gli allarmi ancora attivi al Server Centrale, in maniera proporzionale al traffico ricevuto. In questo modo, viene data evidenza agli operatori ASPI degli allarmi ancora in corso sulla UEL e sulle URVs.

I paragrafi successivi descrivono in dettaglio le operazioni svolte per ogni comando di controllo remoto inviato dal Server Centrale.

5.10.1 Verifica dello stato della UEL

Attraverso questo comando, il Server Centrale richiede di verificare lo stato attuale della UEL inviando un opportuno messaggio di richiesta (UEL_GET_STATUS).

La UEL, alla ricezione del messaggio, esegue i passi descritti diagramma seguente:

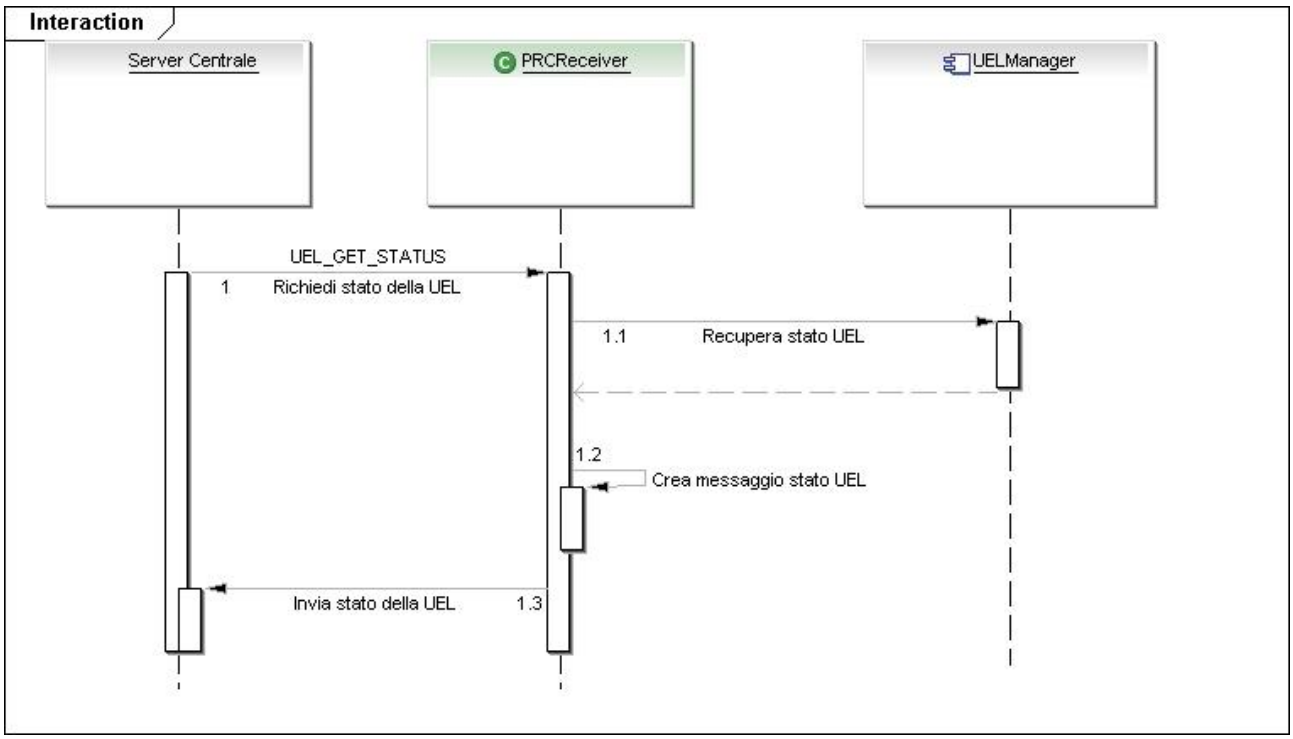

Figura 17 - Richiesta stato della UEL, diagramma di sequenza

- Il PRCReceiver riceve il messaggio contenete la richiesta di stato della UEL; una volta verificato che si tratta di un richiesta di stato, richiede al servizio UELManager lo stato della UEL stessa.
- L'UELManager restituisce lo stato della UEL, relativamente allo stato operativo in cui si trova, alla versione software della UEL e al funzionamento del server NTP; il PRCReceiver impacchetta i dati ricevuti all'interno di un messaggio, quindi invia al Server Centrale la

risposta contenente lo stato della UEL.

5.10.2 Test rete di campo

Attraverso questo comando, il Server Centrale richiede alla UEL di verificare il collegamento degli apparati sulla rete di campo. Il Server Centrale invia un opportuno messaggio di richiesta alla UEL (UEL_TEST_NETWORK) alla ricezione del quale, vengono eseguiti i passi descritti nel diagramma seguente:

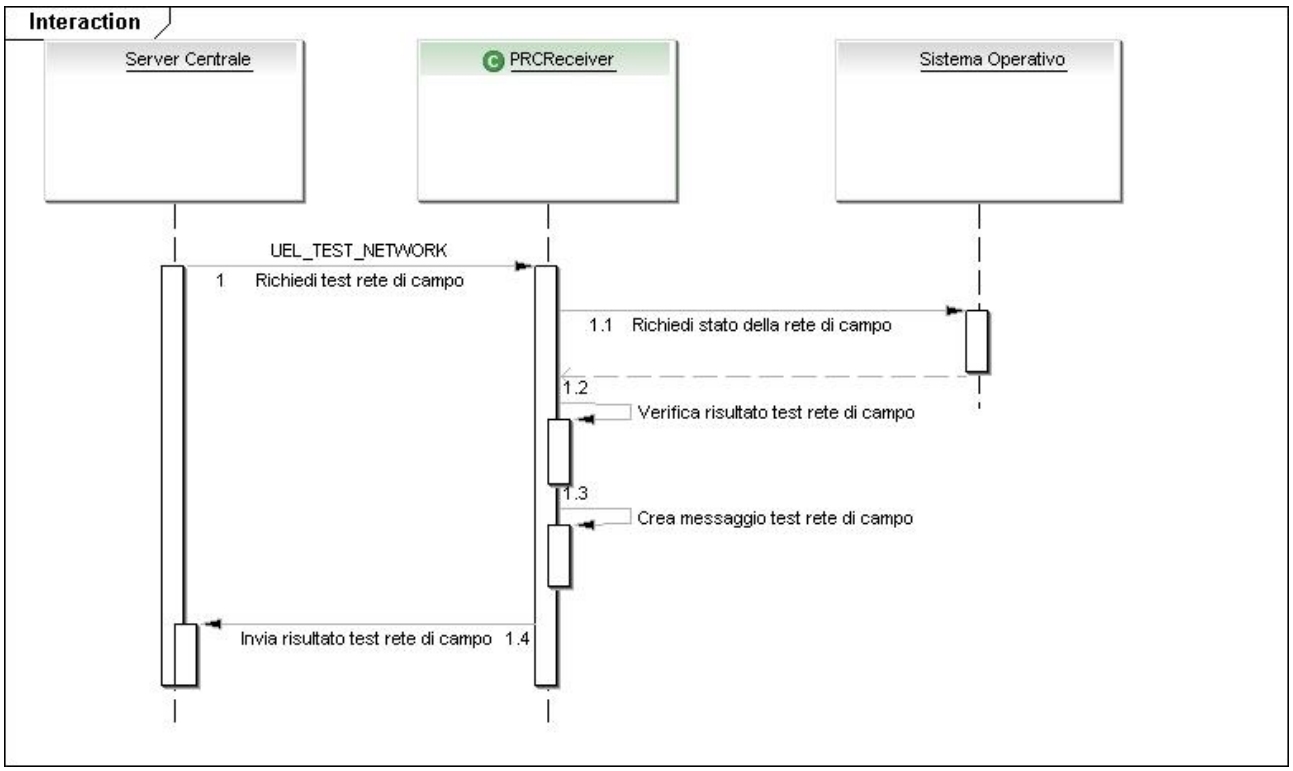

Figura 18 - Richiesta test rete di campo, diagramma di sequenza

Il PRCReceiver, ricevuto un comando di richiesta "test rete di campo", esegue i seguenti passi:

- Esegue il comando di sistema `nmap -sP [lista indirizzi IP URVs] -v -oX netstatus.xml`
- Esegue il parsing del file `netstatus.xml` (risultato del comando precedente) per verificare lo stato dei collegamenti
- Viene creato un messaggio di risposta contenente lo stato dei collegamenti;
- Viene inviato il messaggio di risposta al Server Centrale

5.10.3 Disattivazione della UEL

Attraverso questo comando, il Server Centrale richiede alla UEL di disattivarsi, eliminando dalla memoria le quantità di sicurezza precedentemente recuperate, di modificare il run-level del sistema operativo e di porsi nello stato statistiche in locale. L'esecuzione di tale comando agisce contemporaneamente su entrambe le istanze UEL presenti sul sistema periferico (ove queste fossero presenti). Tale restrizione è dovuta al fatto che la procedura di disattivazione modifica il run-level del sistema operativo e riavvia la macchina per fare in modo che tale modifica abbia effetto.

Il Server Centrale invia uno specifico comando per richiedere la disattivazione (UEL_DEACTIVATE) quindi la UEL svolge i passi indicati nel seguente diagramma:

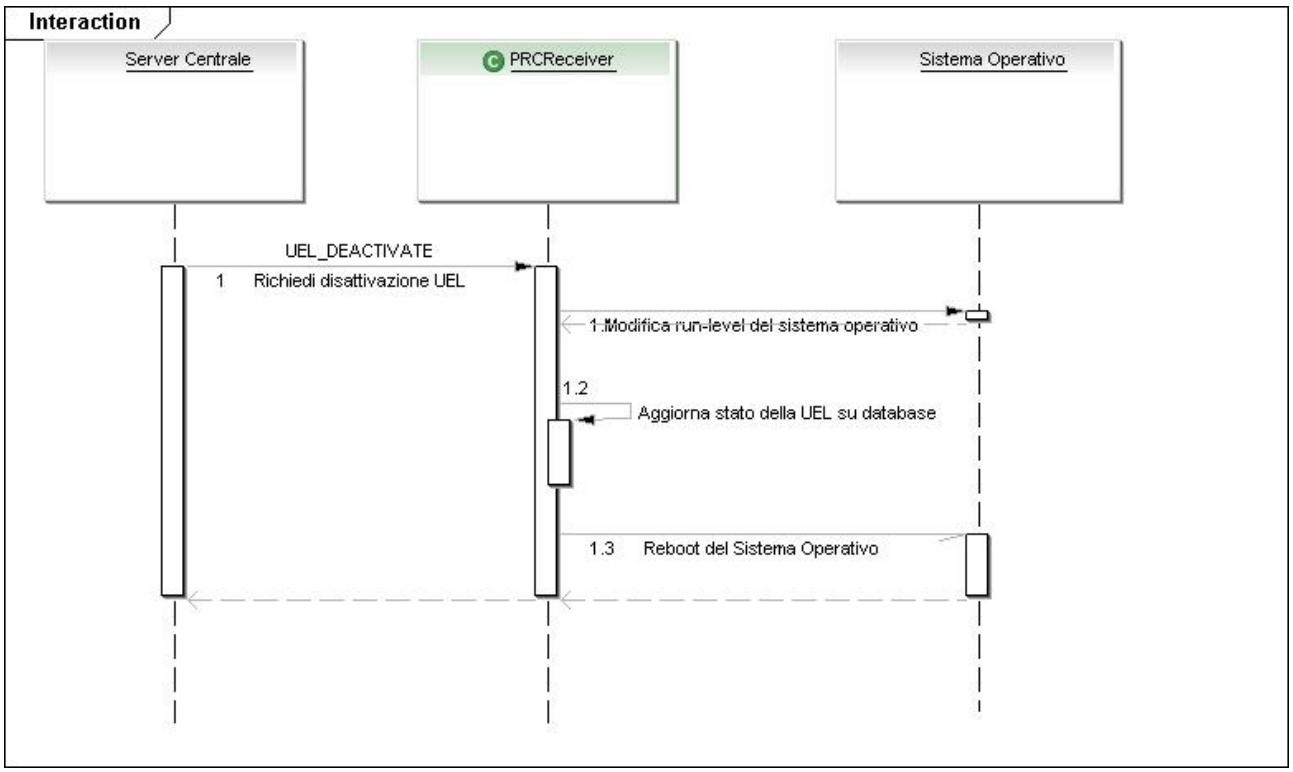

Figura 19 - Diasattivazione UEL, diagramma di sequenza

Alla ricezione del comando di disattivazione, il PRCReceiver svolge le seguenti operazioni:

- Elimina dalla memoria i certificati posseduti;
- Tramite script di sistema modifica il run-level del sistema operativo;
- Aggiorna lo stato operativo della UEL sul database in "Statistiche in locale";
- Tramite comando di sistema, effettua il reboot del sistema operativo;

Al termine di tale procedura, il sistema periferico verrà dunque riavviato, rendendo possibile

l'accesso agli utenti manutentore e attivatore. Lo stato operativo della UEL sarà "Statistiche in locale"; nel caso di doppia istanza, entrambe assumeranno tale stato.

Le quantità di sicurezza, poiché presenti in memoria volatile, verranno rimosse automaticamente al riavvio del sistema periferico.

5.10.4 Verifica dello stato delle URV

Attraverso tale comando, il Server Centrale richiede lo stato di una delle URV collegate alla UEL. La UEL, alla ricezione di tale comando (URV_GET_STATUS), esegue i passi descritti nella seguente figura:

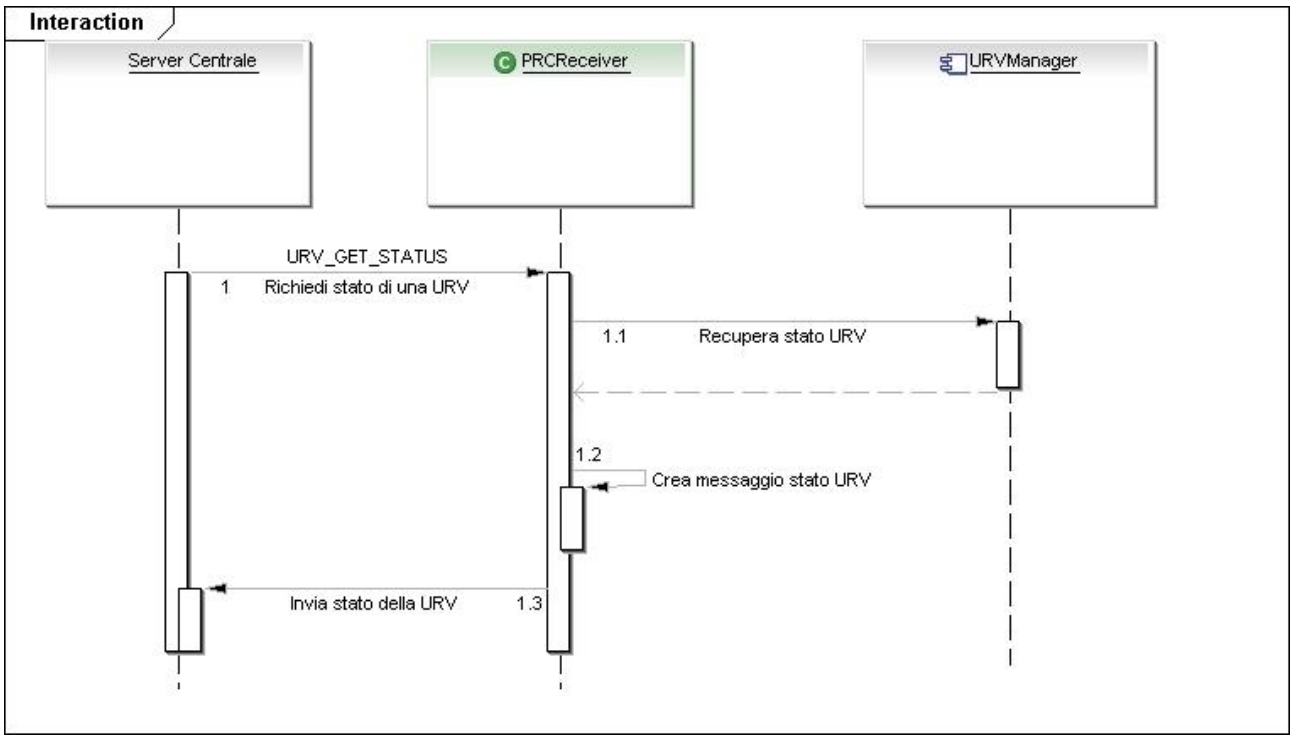

Figura 20 - Richiesta stato della URV, diagramma di sequenza

I passi eseguiti dal PRCReceiver sono i seguenti:

- Verifica che il comando ricevuto è la richiesta di stato di una della URV;
- Richiede al servizio URVManager lo stato della URV specifica;
- Crea un messaggio di risposta contenente lo stato della URV;
- Invia il messaggio al Server Centrale.

Come descritto precedentemente, l'URVManager, attraverso uno specifico componente, interroga periodicamente le URVs per recuperarne lo stato attraverso opportuno comando; quando viene richiesto all'URVManager lo stato di una URV, questi invia l'ultimo stato recuperato senza dover inviare il comando di richiesta stato alla URV.

5.10.5 Verifica dello stato EnDetector

Attraverso tale comando, il Server Centrale richiede lo stato dell'EnDetector che si trova

all'interno della UEL. La UEL, alla ricezione di tale comando (EDT_GET_STATUS), esegue i passi seguenti:

- Verifica che il comando ricevuto sia la verifica dello stato dell'EnDetector;
- Richiede al servizio PMManager lo stato dell'EnDetector;
- Crea un messaggio di risposta contenente lo stato dell'EnDetector;
- Invia il messaggio al Server Centrale.

Come descritto precedentemente, il PMManager, interroga periodicamente l'EnDetector per recuperarne lo stato attraverso opportuno comando; quando viene richiesto lo stato al PMManager, questi invia l'ultimo stato recuperato senza dover inviare il comando di richiesta stato.

5.10.6 Stop di una URV

Attraverso questo comando, il Server Centrale chiede di arrestare una delle URVs collegate alla UEL. La ricezione di tale comando (URV_STOP), scatena i passi descritti nel diagramma di sequenza seguente:

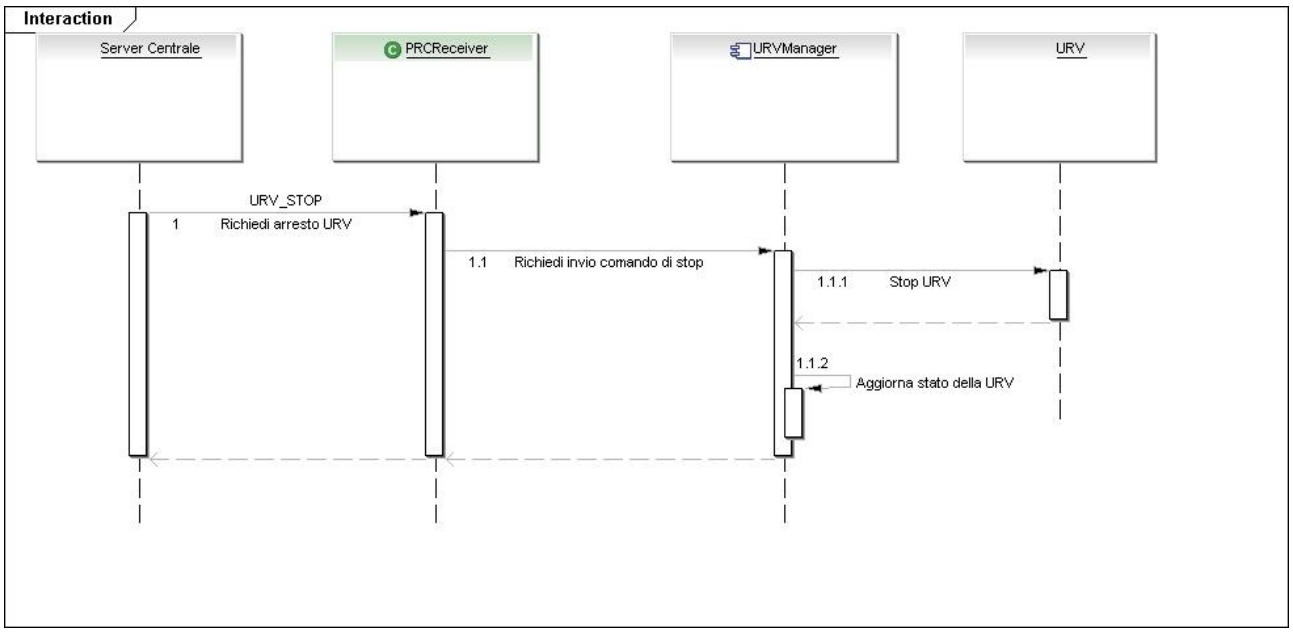

Figura 21 - Stop di una URV, diagramma di sequenza

Alla ricezione del messaggio di stop, il PRCReceiver esegue le seguenti operazioni:

- Richiede all'URVManager di inviare un comando di stop alla URV;
- Abilita il firewall relativamente alla URV interessata
- Invia un messaggio di conferma al Server Centrale.

5.10.7 Start di una URV

Attraverso questo comando, il Server Centrale chiede di avviare una delle URVs collegate alla UEL. La ricezione di tale comando (URV_START), scatena i passi descritti nel diagramma di sequenza seguente:

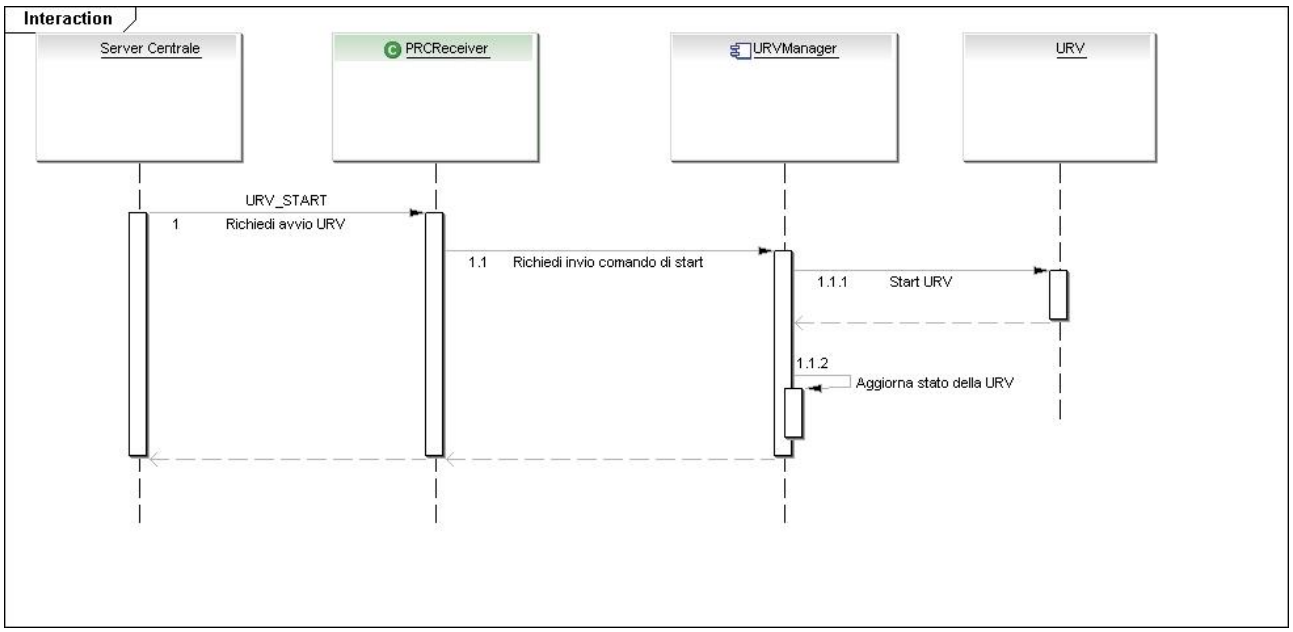

Figura 22 - Start di una URV, diagramma di sequenza

Alla ricezione del messaggio di start, il PRCReceiver esegue le seguenti operazioni:

- Disabilita il firewall relativo alla URV interessata;
- Richiede all'URVManager di inviare un comando di start alla URVs;
- Invia un messaggio di conferma al Server Centrale.

5.10.8 Start/Stop dell'EnDetector

Attraverso questo comando, il Server Centrale chiede di avviare (o stoppare) l'enDetector presente all'interno della UEL. La ricezione di tale comando (EDT_START/EDT_STOP), scatena i seguenti passi:

Alla ricezione del messaggio di start, il PRCReceiver esegue le seguenti operazioni:

- Richiede al PMManager di inviare un comando di start (o di stop) all'enDetector;
- Invia un messaggio di conferma al Server Centrale.

5.10.9 Reset di una URV

Attraverso questo comando, il Server Centrale chiede di resettare una delle URVs collegate alla UEL. La ricezione di tale comando (URV_RESET), scatena i passi descritti nel diagramma di sequenza seguente:

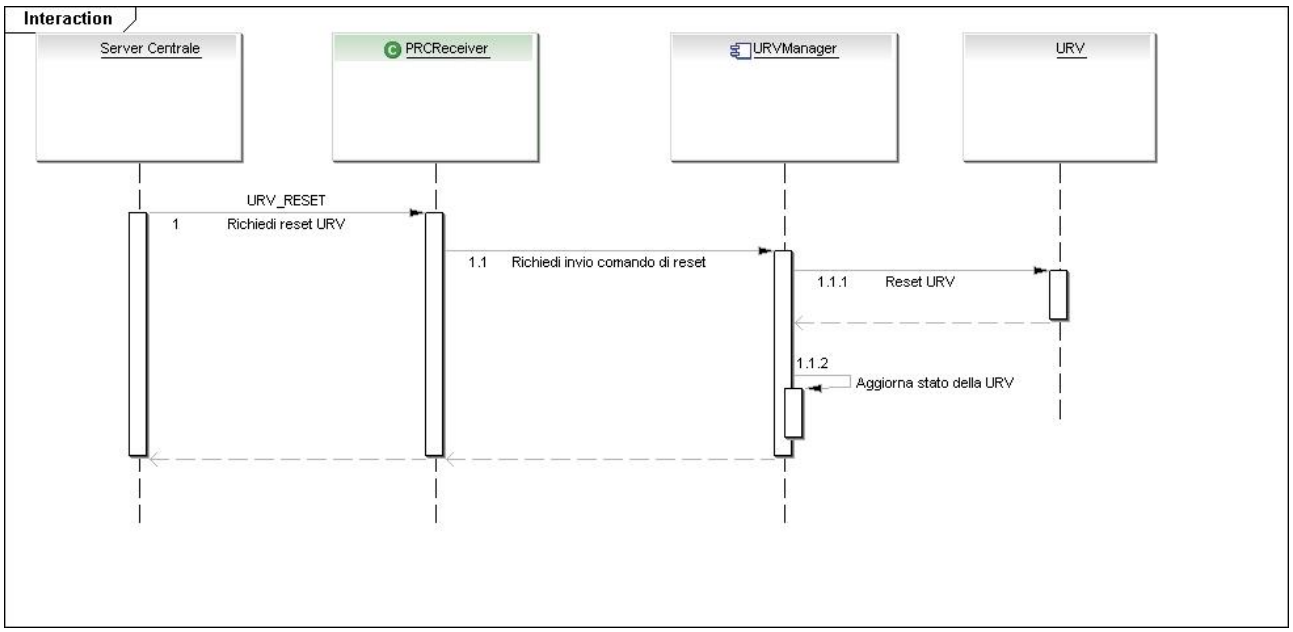

Figura 23 - Reset di una URV, diagramma di sequenza

Alla ricezione del messaggio di reset, il PRCReceiver esegue le seguenti operazioni:

- Disabilita il firewall relativo alla URV interessata;
- Richiede all'URVManager di inviare un comando di reset alla URVs;
- Invia un messaggio di conferma al Server Centrale.

Il messaggio di conferma inviato al Server Centrale serve ad informare quest'ultimo che la fase di reset è stata avviata. Il termine di tale fase dipende dal funzionamento della URV stessa.

5.10.10 Avvio fase di learn

Attraverso questo comando, il Server Centrale chiede di avviare una fase di learn dei rilevatori di velocità e classe su una delle URVs collegate alla UEL. Questa funzionalità permette di rifasare i rilevatori e viene attivata tipicamente dopo l'emissione da parte di una URV di un allarme specifico, tale allarme avverte che una serie consecutiva di transiti hanno i bit di qualità della rilevazione diversi da 0000.

La ricezione di tale comando (URV_LEARN), scatena i passi descritti nel diagramma di sequenza seguente:

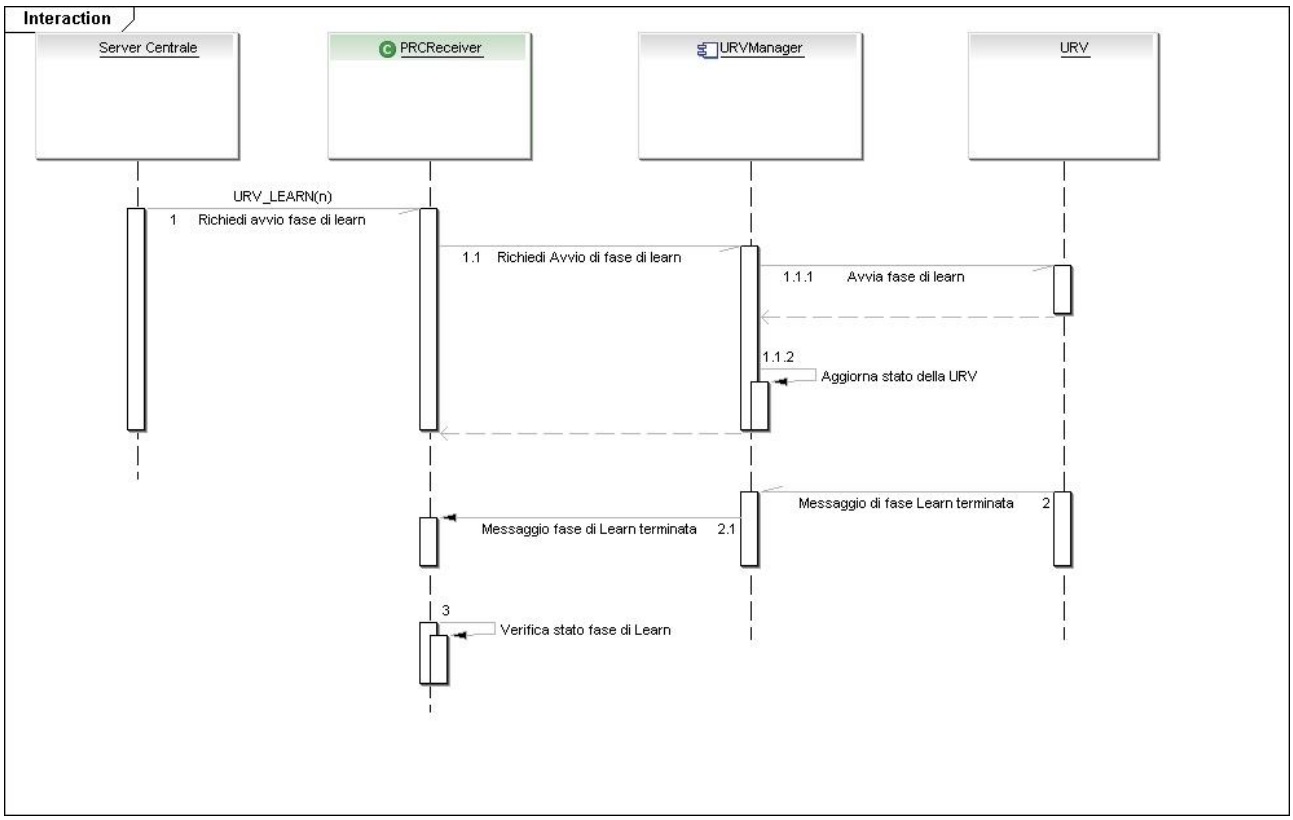

Figura 24 - Fase di learn, diagramma di sequenza

Alla ricezione di tale comando, il PRCReceiver esegue le seguenti operazioni:

- Invia un messaggio di conferma ricezione al Server Centrale del messaggio (Fase di learn gestita in maniera asincrona)
- Richiede all'URVManager di arrestare la URV;
- Richiede all'URVManager di avviare una fase di learn sulla URV;
- L'URVManager invia alla URV il comando di avvio fase di learn.

- Aggiorna in maniera opportuna lo stato della URV
- Al termine della fase di learn, in modalità asincrona, la URV invia un allarme per notificare l'evento, specificando se la fase stessa è stata terminata con successo o con errore.
- Viene aggiornato lo stato della URV (Learn terminato con successo/errore)
- In caso di fase di learn terminata con successo, il PRC Service richiede all'URVManager di inviare un comando di start alla URV interessata.

5.10.11 Richiesta immagini di prova

Con tale comando, il Server Centrale richiede alla UEL di inviare, relativamente ad una delle URV collegate, delle immagini di prova per valutare la qualità delle stesse.

Alla ricezione di tale comando, il PRC Service svolge le seguenti operazioni:

- Richiede all'URVManager di inviare un comando di immagini di prova alla URV;
- Si pone in attesa su una coda locale delle 5 immagini di prova.

La URV invierà le immagini di prova al servizio URVManager; tali immagini avranno il numero di sequenza pari a 0.

L'URVManager, riconosciute le immagini come immagini di prova, le invia sulla coda locale dove si trova in ascolto il PRC Service.

Alla ricezione della quinta immagine di prova, il PRC Service impacchetta le immagini recuperate e provvede ad inviarle al Server Centrale.

5.10.12 Architettura lato Server Centrale: SRC Service

Il componente duale del PRC Service lato Server Centrale viene denominato SRC Service. Esso si occupa sia di inviare tutti i comandi di controllo remoto verso le UEL, sia di ricevere i messaggi di allarme inviati dalle UEL. La figura seguente mostra l'architettura interna del servizio SRC.

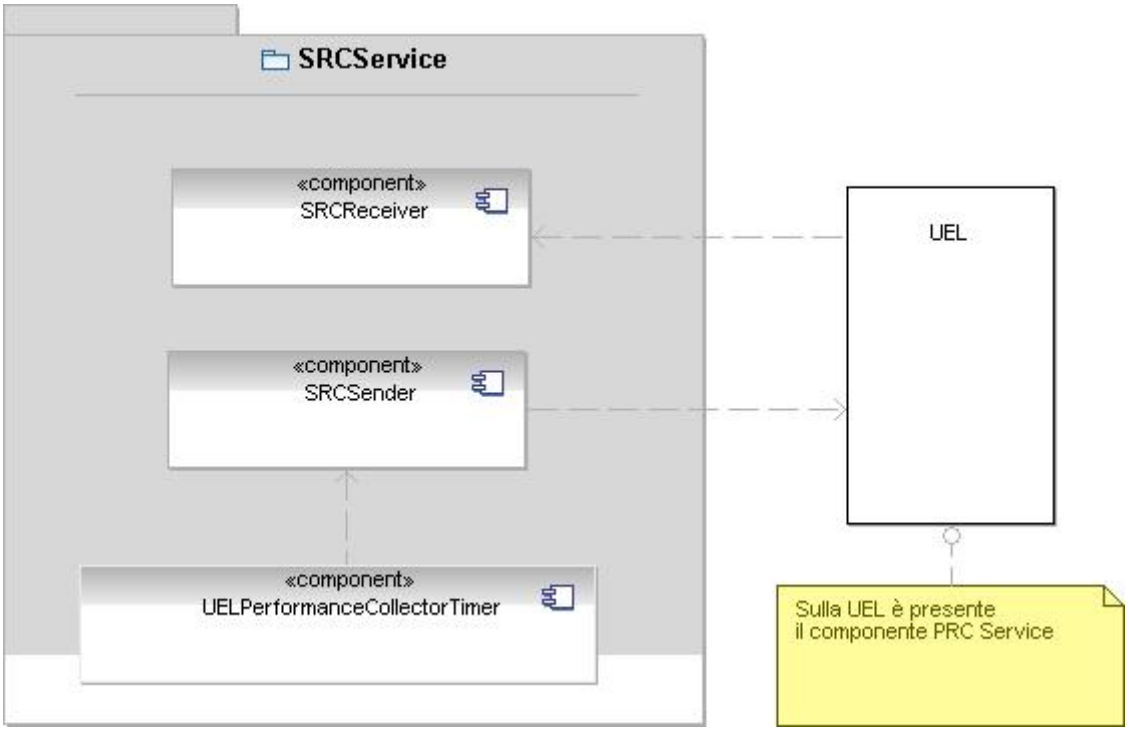

Figura 25 - Architettura del servizio SRC (Server Centrale)

Il servizio SRC è costituito da tre componenti principali, oltre al servizio stesso:

- SRCReceiver si occupa di ricevere gli allarmi generati dalle UEL e di inserirli nel database;
- SRCSender si occupa di inviare i comandi si controllo remoto alle UEL, invocati tramite applicazione Web Admin ASPI;
- UELPerformanceCollectorTimer si occupa di recuperare periodicamente le statistiche di funzionamento delle UEL collegate al Server Centrale.

Il servizio SRC espone una interfaccia pubblica che verrà utilizzata dall'applicazione Web di amministrazione per eseguire i comandi di controllo remoto; l'invio del comando alla UEL verrà eseguito dal componente SRCSender. La figura seguente mostra i passi eseguiti durante l'invio di un comando.

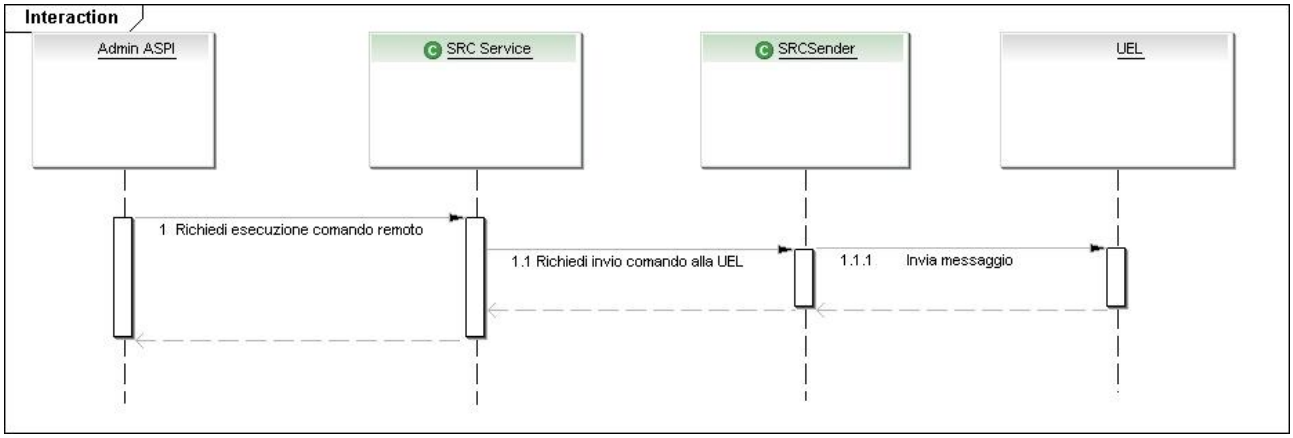

Figura 26 - Invio di un comando di controllo remoto

Come descritto precedentemente, l'applicazione Web utilizzerà l'interfaccia del servizio SRC, il quale delegherà al componente SRCSender l'invio del messaggio.

La figura seguente mostra i passi eseguiti dal servizio SRC alla ricezione di un nuovo allarme da una UEL.

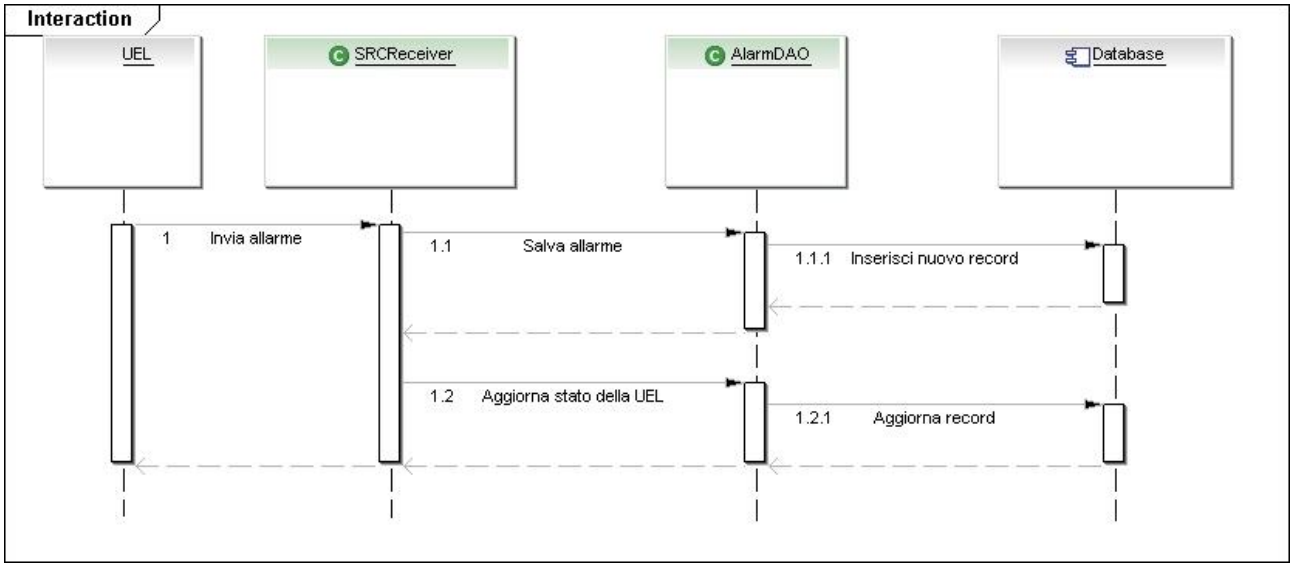

Figura 27 - Ricezione di un allarme (Server Centrale)

Alla ricezione di un allarme, l'SRCReceiver si occupa di salvare l'allarme ricevuto, di aggiornare lo stato della UEL sul database e di inviare alla UEL la conferma dell'avvenuta ricezione. La UEL provvede ad eliminare l'allarme.

L'ultimo componente presente all'interno del servizio SRC, l'UELPerformanceCollectorTimer, si occupa, periodicamente, di interrogare tutte le UEL collegate, al fine di recuperare le statistiche di funzionamento delle macchine. Le statistiche collezionate sono le seguenti:

- Carico della CPU;

- Utilizzo della memoria;
- Utilizzo dello spazio disco;

Nel caso in cui venga rilevato un sovraccarico delle risorse, l'UELPerformanceCollectorTimer genera un opportuno allarme per informare l'amministratore ASPI del problema rilevato. La figura seguente mostra i passi eseguiti dall' l'UELPerformanceCollectorTimer.

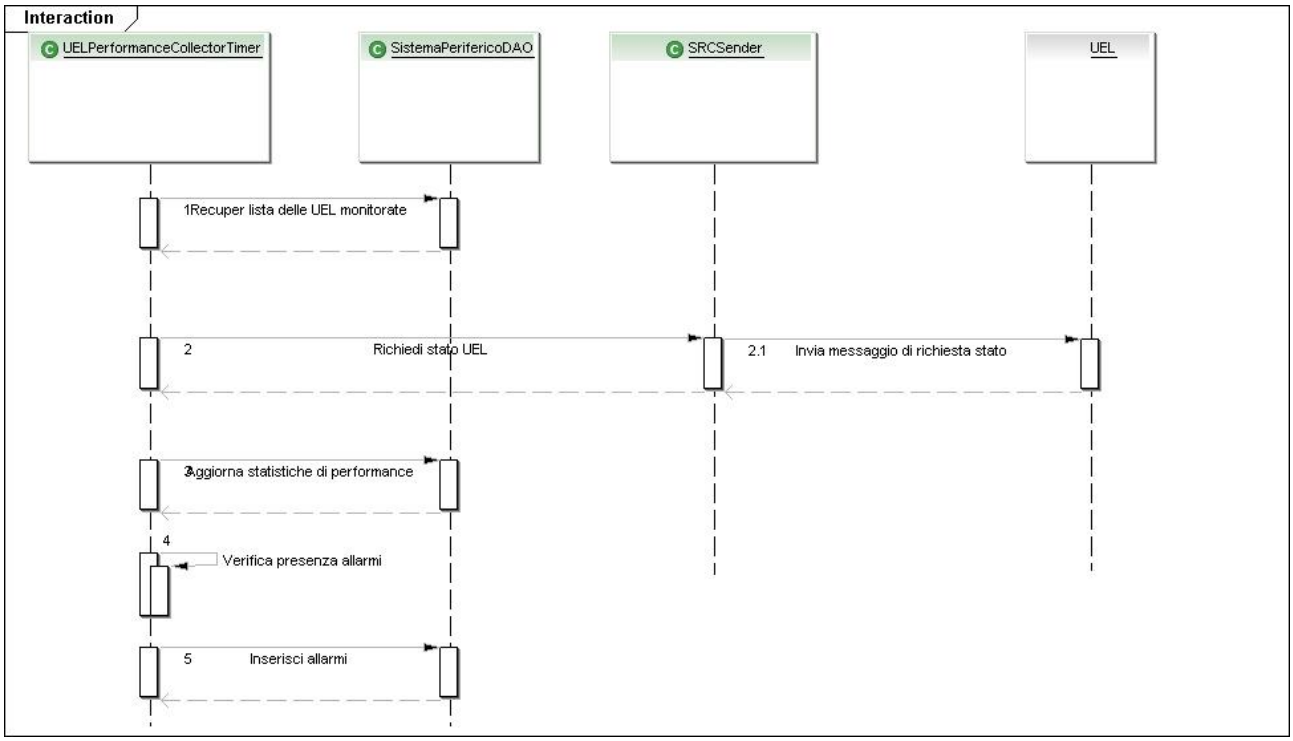

Figura 28 - Raccolta dei dati di performance (Server Centrale)

Non sono presenti altri componenti, all'infuori dell'applicazione web Admin ASPI, che facciano uso del servizio SRC.

5.11 Gestione dei dati di transito: PCapture Service

Il servizio PCapture si occupa di elaborare i transiti rilevati dalle URVs e di inviare il risultato di tale elaborazione verso il Server Centrale. Il servizio PCapture rappresenta il cuore dell'applicazione UEL, occupandosi di svolgere le operazioni computazionalmente più onerose.

L'architettura del servizio PCapture viene descritta dalla seguente figura.

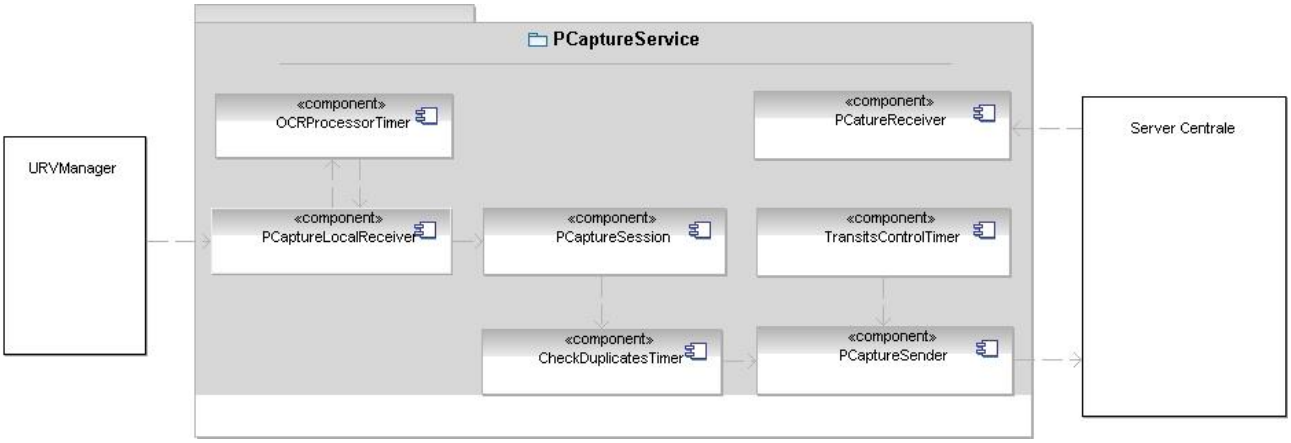

Figura 29 - Architettura del servizio PCapture

Le funzionalità svolte dal servizio PCapture sono strettamente dipendenti dallo stato operativo in cui si trova la UEL stessa. Ogni volta che la UEL transita da uno stato operativo all'altro, l'UELManager informa il servizio PCapture della transizione di stato occorsa; sarà quindi il servizio PCapture stesso ad eseguire le funzionalità richieste per quel particolare stato ed ad inibire le eventuali funzionalità non richieste. Nel seguito del capitolo, per ogni componente descritto, verranno indicate le funzionalità svolte e gli stati operativi in cui vengono richieste.

Il servizio PCapture viene alimentato direttamente dal servizio URVManager, il quale, attraverso il componente FTPMessageListener, invia i messaggi di trasferimento FTP completato inviati dalle URVs collegate. Il componente che si occupa di ricevere i messaggi di *trasferimento FTP completato* è il PCaptureLocalReceiver. Le funzionalità svolte da tale componente vengono descritte nel paragrafo seguente.

5.11.1 Ricezione dei messaggi di trasferimento FTP completato

Quando il servizio URVManager riceve un messaggio di *trasferimento FTP completato*, lo rende persistente quindi lo invia al servizio PCapture per avviare l'elaborazione dello stesso. All'interno del PCapture è presente un componente specifico, il PCaptureLocalReceiver, che si occupa di ricevere

tali messaggi dall'URVManager. Tale componente viene avviato allo start-up della UEL ed opera in tutti gli stati operativi in cui si trova la UEL.

Alla ricezione del messaggio, il PCaptureLocalReceiver delega la fase di elaborazione del transito ad un altro componete, il PCaptureProcessTransit, che esegue i seguenti passi:

- Recupera l'header Jpeg dal messaggio ricevuto;
- Effettua la decodifica l'header Jpeg ;
- Effettua un controllo sulla qualità del header decodificato
- Se necessario confronta header Jpeg presente sul messaggio con quello presente sul file
- Se necessario effettua la copia del file immagine ricevuto
- Se la uel si trova in verifica violazione controlla se la data del transito è antecedente a quella del servizio
- Controlla la validità temporale dell'immagine (NTP e allarme sincronizzazione della URV);
- Nel caso la UEL sia con tecnologia OCR, verifica se il transito è una targa letta, in caso contrario avvia l'OCR di secondo livello, marcando il record in attesa di OCR (vedi paragrafo successivo);
- Nel caso la UEL sia con tecnologia PlateMatching, avvia l'elaborazione del transito mediante enDetector, marcando il record in attesa di EDT (vedi paragrafo successivo);
- In caso di istantanea, verifica la validità dei parametri Weiss (accuratezza velocità)
- In caso di istantanea, verifica se la velocità del transito è inferiore al limite impostato, se ciò è vero il transito non sarà salvato
- Inserisce nel database i dati statistici relativi al transito (dati puntuali);
  Recupera il path e il filename dal messaggio;

Terminate le operazioni precedenti, il PCaptureProcessTransit verifica lo stato operativo della UEL, e sulla base di questo continua il flusso dell'elaborazione:

- *Statistiche in locale*: elimina il file dalla directory FTP ed elimina il messaggio FTP dal database;
- *Disponibile*: se abilitato alla raccolta di dati statistici in media, salva i dati del transito per l'invio successivo delle statistiche in media al Server Centrale, quindi elimina il file immagine ed il messaggio FTP dal database;
- *Verifica Violazioni*: viene avviato il processo di salvataggio del transito e della relativa immagine; viene eliminato il messaggio FTP dal database.

5.11.2 Salvataggio di un transito

Nel caso in cui la UEL stia operando in verifica violazioni, il PCaptureProcessTransit, terminate le proprie operazioni, avvia la fase di salvataggio del transito su database e della relativa immagine associata e delega tutta la fase di salvataggio ad un altro componente il PCaptureSaveTransit.

I passi eseguiti dal PCaptureSaveTransit sono i seguenti:

- Salva i dati del transito su database locale;
- Recupera il file immagine dalla directory del server FTP;
- Firma e cripta il file immagine
    - firma il file utilizzando il certificato di firma (PKCS#7);
    - genera una chiave casuale a 128 bit;
    - cripta il file firmato con l'algoritmo DES utilizzando la chiave generata precedentemente;
    - sostituisce il file sorgente con il file PKCS#7;
    - cripta con l'algoritmo RSA la chiave generata utilizzando la chiave pubblica del certificato di firma;
    - salva, sul database, la chiave casuale criptata;
    - cripta con l'algoritmo RSA la targa utilizzando la chiave pubblica del certificato di firma;
    - salva, sul database, la targa criptata;

Al termine di questa fase il record relativo al transito rimane in attesa di essere sottoposto al controllo duplicati prima di essere inviato al Server Centrale.

5.11.3 Verifica transiti duplicati

Il controllo dei duplicati viene affidato ad un componente temporizzato che si occupa, periodicamente, di verificare quali transiti possono essere inviati al Server Centrale. Tale componente, denominato CheckDuplicatesTimer, implementa l'algoritmo di verifica duplicati descritto nel documento dei requisiti. Tale componente è attivo soltanto con la UEL abilitata con tecnologia OCR, mentre non è attivo con la UEL PlateMatching.

Un transito duplicato può verificarsi nel caso un veicolo transiti a cavalo di due corsie, ed entrambe le URV effettuino il rilevamento del veicolo stesso.

Nel caso venga verificata la presenza di un transito duplicato, relativo ad un servizio in istantanea

la scelta del transito da mantenere viene fatta sulla base dei seguenti criteri:

- Byte Qualifier del dispositivo Weiss;
- Minore velocità del veicolo;
- Qualità della lettura OCR

Nel caso venga verificata la presenza di un transito duplicato, relativo ad un servizio in media la scelta del transito da mantenere viene fatta sulla base dei seguenti criteri:

- Qualità della lettura OCR;
- Classificazione del dispositivo Weiss;
- Orario di rilevazione più vecchio.

Al termina della verifica, il transito viene inviato al Server Centrale, attraverso l'infrastruttura di messaggistica utilizzando un componente dedicato il PCaptureSender(descritto più avanti). Per maggiori dettagli sull'algoritmo di controllo dei duplicati fare riferimento al documento [2]

5.11.4 Riconoscimento OCR secondo livello

Il servizio PCapture, per tutti i transiti con targa non letta o non rilevata, avvia un processo OCR che tenta di riconoscere la targa all'interno dell'immagine stessa. Il riconoscimento OCR viene fatto da una applicazione esterna alla UEL, richiamata attraverso esecuzione di sistema. Tale applicazione può ricevere in input il nome file di un immagine, oppure una directory; per ogni immagine tenta il riconoscimento della targa, aggiorna l'header Jpeg dell'immagine e salva il nuovo file con estensione modificata.

All'interno del PCapture esiste uno specifico componente, denominato OCRProcessorTimer, che si occupa di lanciare l'applicazione OCR per tutti i transiti che si trovano in attesa. Le operazioni svolte da tale componente sono le seguenti:

- Recupera dal database la lista dei transiti in attesa di lettura OCR;
- Controlla se esistono nella cartella di lavoro dei file che sono rimasti appesi durante una precedente sessione di lavoro, se ciò è vero li rinomina e l'inserisce nella lista precedentemente creata
- Per ogni transito, sposta la relativa immagine presente sotto l'alberatura del server FTP in una propria cartella di lavoro;

- Lancia l'applicazione OCR passandogli come parametro la directory di lavoro;
- Attende che l'applicazione OCR termini l'elaborazione ed attende il valore di ritorno restituito;
- Controlla se il processo Plates ha restituito dei messaggi di errore, se ciò è vero traccia l'errore nei file di log ed cancella i dati del transito sul File System e sul database
- Per ogni immagine, viene recuperato il nuovo header dal file elaborato;
- Aggiorna i messaggi FTP sul database (lettura OCR e nuovo Header Jpeg);
- Sposta il file elaborato nella directory di lavoro della UEL;
- Invia, per ogni transito, un messaggio al PCaptureLocalReceiver per permetterne l'elaborazione.

Il PCaptureLocalReceiver eseguirà nuovamente le elaborazioni sul transito, inserendo solamente le statistiche relative alla lettura OCR.

Nel caso del l'applicazione OCR non sia in grado di leggere la targa dall'immagine, il transito verrà trattato come una targa non letta/non rilevata;

5.11.5 Rilevazione stringhe PM EnDetector

Il servizio PCapture, per tutti i transiti marcati come in attesa EDT (attesa EnDetector), avvia un processo che ricava le stringhe PM identificative del veicolo. Tali stringhe sono ricavate inoltrando l'immagine abbinata al transito al PMManager, che a sua volta si interfaccia con EnDetector.

All'interno del PCapture esiste uno specifico task che si occupa di inoltrare le immagini al PMManager per tutti i transiti che si trovano in attesa. Le operazioni svolte da tale componente sono le seguenti:

- Recupera dal database (Tabella ENDETECTOR_FILES) la lista dei transiti in attesa EDT;
- Controlla se esistono nella cartella di lavoro dei file che sono rimasti appesi durante una precedente sessione di lavoro, se ciò è vero li rinomina e li inserisce nella lista precedentemente creata
- Per ogni transito, sposta la relativa immagine presente sotto l'alberatura del server FTP in una propria cartella di lavoro;
- Invoca il PMManager passandogli come parametro la directory di lavoro;
- Attende che PMManager termini l'elaborazione ed attende il valore di ritorno restituito;
- Se PMManager ha restituito dei messaggi di errore, traccia l'errore nei file di log ed cancella i

dati del transito sul File System e sul database

- Per ogni immagine, viene recuperato il nuovo header dal file elaborato;
- Sposta il file elaborato nella directory di lavoro della UEL;
- Invia, per ogni transito, un messaggio al PCaptureLocalReceiver per permetterne l'elaborazione.

Il PCaptureLocalReceiver eseguirà nuovamente le elaborazioni sul transito, inserendo le informazioni relative alla lettura della stringa PM nella tabella SICVE_FILES.

Nel caso del l'applicazione EDT non sia in grado di leggere la targa dall'immagine, il transito verrà mantenuto fino a retention

5.11.6 Invio transiti al Server Centrale

Per l'invio dei transiti verso il Server Centrale, viene utilizzato uno specifico componente, denominato PCaptureSender, che sarà attivo solo nel caso in cui la UEL si trovi in stato operativo di verifica violazioni.

Nel caso sia attivo un servizio di verifica violazioni in velocità istantanea, viene verificato che la velocità del transito sia maggiore del limite di classe impostato. In particolare:

- se la classe è stata individuata: viene confrontata la velocità rilevata con quella del limite di classe;
- se la classe non è stata individuata: viene confrontata la velocità con quella della classe che ha la velocità più bassa impostata tra le varie classi;

Nel caso in cui si verifichi che la velocità è minore, il transito viene eliminato e l'elaborazione termina.

In caso contrario, viene decisa la modalità di invio sulla base del tipo di servizio avviato (istantanea/media, pre-assegnato, post-assegnato) e sul tipo targa (letta/non letta). A seconda di questi parametri, viene deciso se inviare solo i dati di transito oppure anche l'immagine relativa. La tabella seguente descrive le casistiche possibili:

Tipo Servizio	**Assegnazione**	**Lettura Targa**	**Modalità invio**
Istantanea	Pre-Assegnato	Letta	Invia dati e immagine
Istantanea	Pre-Assegnato	Non Letta	Invia dati e immagine
Istantanea	Post-Assegnato	Letta	Invia dati

Istantanea	Post-Assegnato	Non Letta	Invia dati
Media	Pre-Assegnato	Letta	Invia dati
Media	Pre-Assegnato	Non Letta	Invia dati
Media	Post-Assegnato	Letta	Invia dati
Media	Post-Assegnato	Non Letta	Invia dati

Tabella 1 - Modalità di invio transiti

Nel caso in cui vengano inviati solo i dati del transito, viene tracciato l'invio del transito al Server Centrale, ma non vengono eliminati i dati fino a quando non verrà richiesta l'immagine. La figura seguente mostra i passi relativi all'invio dei dati del transito:

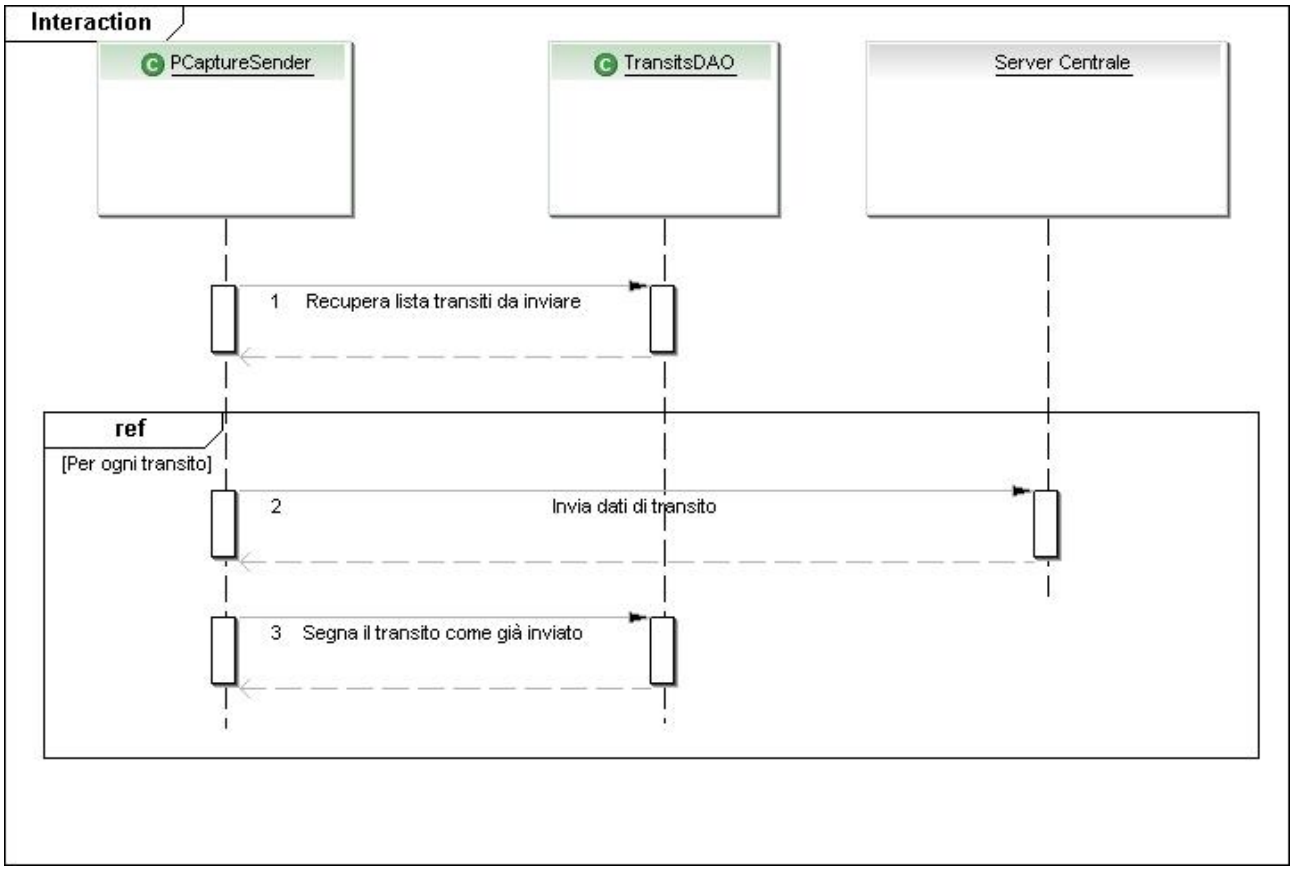

Tabella 2 - Invio dei dati di un transito al Server Centrale

Nel caso in cui vengano inviati sia i dati di transito che l'immagine associata, dopo aver ricevuto la conferma dal Server Centrale dell'avvenuta ricezione, la UEL elimina logicamente il transito (l'eliminazione fisica viene effettuata successivamente da una altro componente). La figura seguente mostra i passi relativi all'invio dei dati di transito e dell'immagine associata:

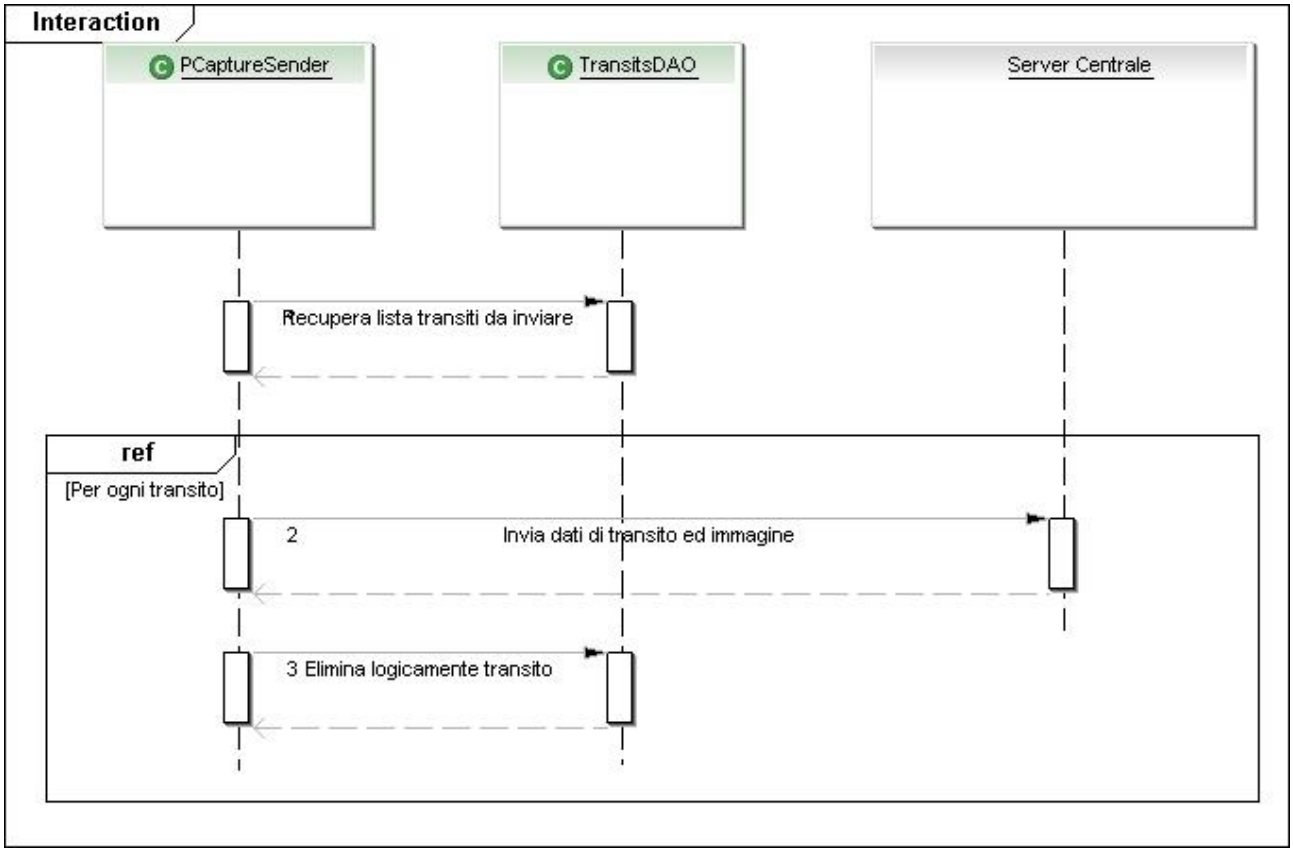

Tabella 3 - Invio di dati di transito ed immagine al Server Centrale

Nel caso in cui non sia possibile inviare il messaggio contenente il transito entro il numero di tentativi configurati (vedi infrastruttura di messaggistica),nel caso tutti i tentativi fallisco lo stesso componete proverà a rinviare i dati nella successiva iterazione.

Il sistema di invio è realizzato creando un unico messaggio contenete una lista di oggetti.

Ogni oggetto contiene le informazioni relative ad un transito da inviare al server..

Anche in questo caso, come per gli allarmi, sarà previsto un numero massimo di transiti da recuperare per l'invio per evitare un sovraccarico del Server Centrale in caso di caduta del collegamento.

5.11.7 Invio status transiti al Server Centrale

La UEL deve periodicamente notificare al server Centrale il timestamp al quale è relativo l'ultimo transito che ha inviato. Questo messaggio permette al Server Centrale di fare assunzioni su quale sia l'istante temporale più recente sul quale può fare assunzioni relative ai transiti ricevuti. La ricezione di questo messaggio infatti, permette al Server Cetnrale di sapere fino a quando sono stati ricevuti tutti i transiti, senza il rischio di perdere informazioni.

Per l'invio di questa informazione verso il Server Centrale, così come per l'invio dei transiti, viene utilizzato uno specifico componente, denominato PCaptureSender, che sarà attivo solo nel caso in cui la UEL si trovi in stato operativo di verifica violazioni.

Il componente periodicamente invia un messaggio contenente il timestamp di invio del messaggio e il timestamp associato all'ultimo transito inviato al Server Centrale.

Nel caso in cui non sia possibile inviare il messaggio contenente il transito entro il numero di tentativi configurati (vedi infrastruttura di messaggistica),nel caso tutti i tentativi fallisco lo stesso componete proverà a rinviare i dati nella successiva iterazione.

5.11.8 Richiesta immagini

Il Server Centrale può richiedere alla UEL le immagini associate ad un transito di cui la UEL aveva inviato solo i dati. Il componente che si occupa di ricevere tali richieste è il PCaptureReceiver. Tale componente è in ascolto sulla coda riservata al canale di capturing, e gestisce tutte le richieste inviate dal Server Centrale. La figura seguente mostra i passi eseguiti nello scenario di richiesta immagini:

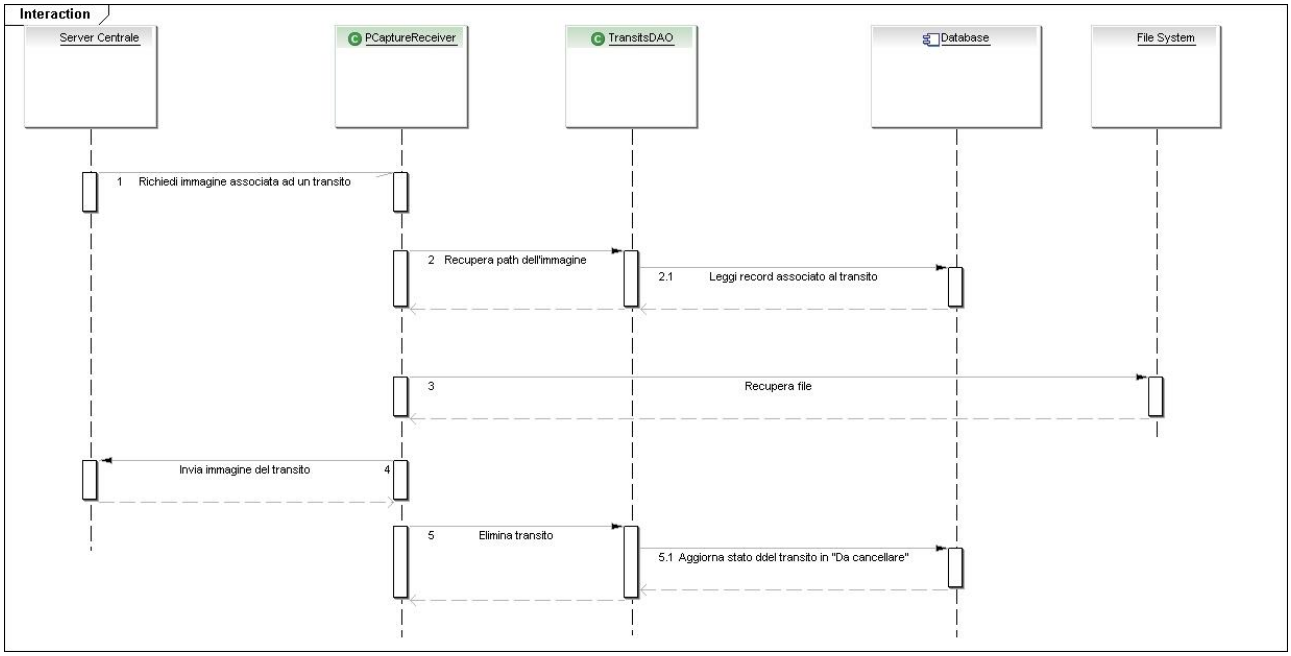

Figura 30 - Richiesta di un'immagine, diagramma di sequenza

Alla ricezione del messaggio richiesta di un'immagine, il PCaptureLocalReceiver esegue i seguenti passi:

- Recupera dal database il percorso e il nome file del transito;

- Recupera da file system l'immagine associata al transito;
- Decifra l'immagine utilizzando il certificato di firma e la chiave casuale memorizzata nel database;
- Invia un messaggio contenente l'immagine al Server Centrale;
- Riceve la conferma dell'avvenuta ricezione dal Server
- Elimina logicamente il transito sul database.

5.11.9 Cancellazione fisica dei transiti e delle immagini

All'interno del PCapture, è presente un componente temporizzato, denominato DeleteTransitsTimer, che si occupa della cancellazione fisica dei transiti e dei relativi file immagine. Come descritto nei paragrafi precedenti, tutte le operazioni di cancellazione vengono eseguite logicamente, impostando lo stato del transito in "da cancellare". Il motivo di tale scelta risiede nel fatto che le cancellazioni vengono sempre eseguite all'interno di uno scenario di comunicazione con il Server Centrale (messaggio di cancellazione, invio immagine, etc.), dove è necessario, per non mantenere troppo tempo la connessione in attesa, svolgere operazioni veloci ed inviare subito un messaggio di conferma. La cancellazione fisica di un file, infatti, è una operazione onerosa per il sistema, se confrontata con il resto delle attività.

Per ovviare a questo problema, la cancellazione viene eseguita mediante impostazione dello stato del transito in "da cancellare"; da questo momento in poi, nessun altro componente utilizzerà tale transito, al di fuori del DeleteTransitsTimer.

La figura seguente descrive i passi eseguiti dal DeleteTransitsTimer:

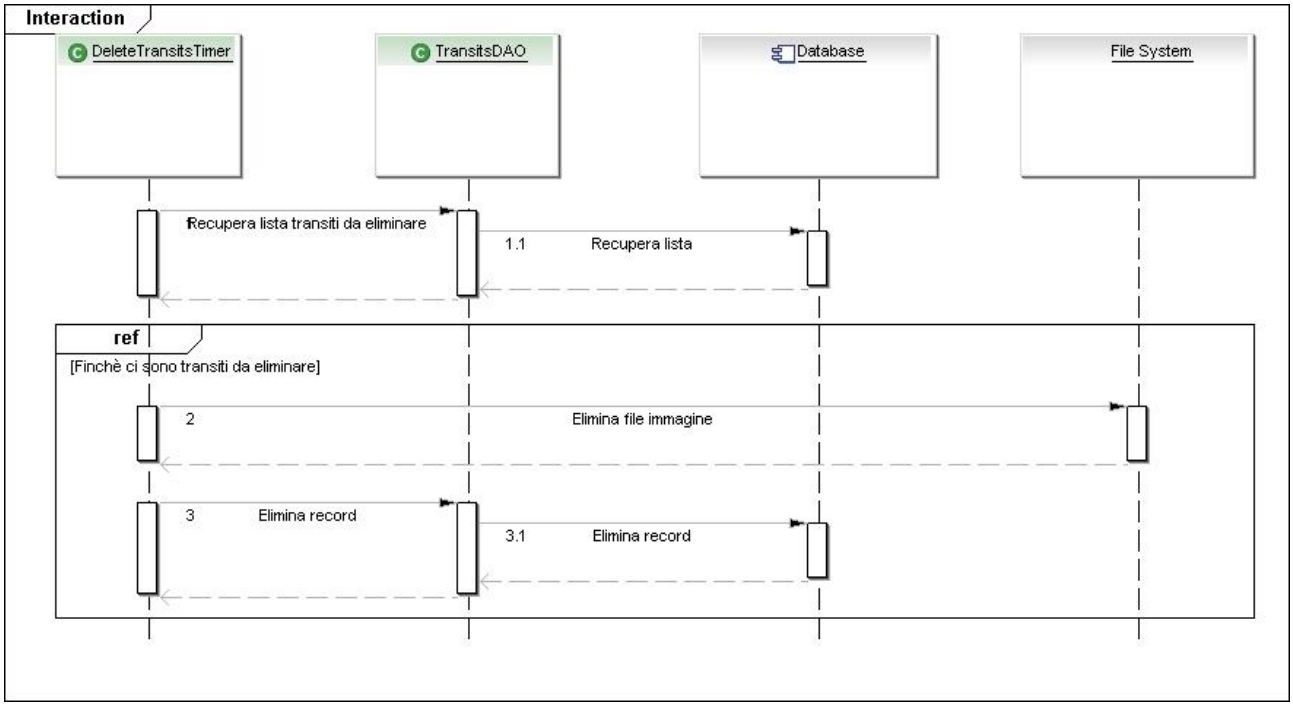

Figura 31 - Cancellazione fisica dei transiti, diagramma di sequenza

Per ogni esecuzione, il DeleteTransitsTimer recupera la lista dei transiti da eliminare, quindi per ognuno di essi esegue i seguenti passi:

- Elimina il file presente su file-system;
- Elimina il relativo record sul database.

5.11.10 Cancellazione dei transiti

Il Server Centrale può richiedere alla UEL l'eliminazione di un transito e della relativa immagine; tale richiesta può essere fatta per scadenza del periodo di retention (vedi paragrafo successivo) oppure quando, in seguito all'accoppiamento, viene riscontrato che un transito non costituisce infrazione.

E' previsto uno specifico messaggio inviato dal Server Centrale per richiedere la cancellazione di un transito. La figura seguente mostra lo scenario di interazione tra Server Centrale ed UEL.

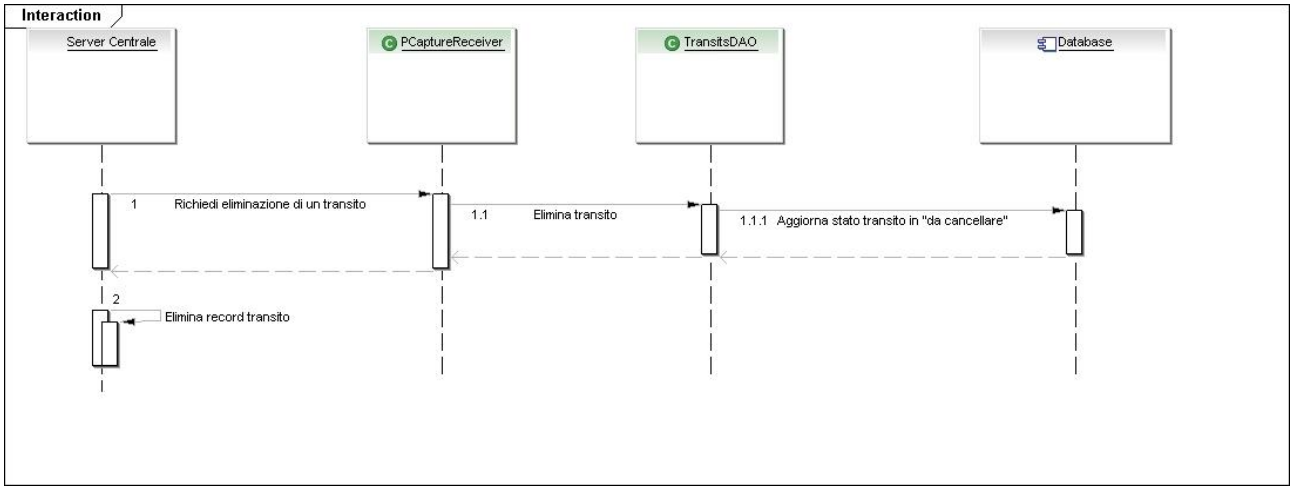

Figura 32 - Richiesta di eliminazione di un transito, diagramma di sequenza

Alla ricezione di un messaggio di richiesta cancellazione, la UEL aggiorna il record del database relativo al transito impostando lo stato a da cancellare;

5.11.11 Cancellazione dei transiti per data

Il sistema centrale può richiedere la cancellazione di tutti i transiti e delle relative immagini che siano appartenenti a un determianto servizio e antecedenti a una certa data. La UEL, ricevuto il messaggio, provvede alla cancellazione logica dei transiti che soddisfano la i parametri contenuti nella richiesta, quindi invia un messaggio di conferma al Server Centrale, il quale, provvederà egli stesso alla cancellazione del transito in suo possesso.

5.11.12 Cancellazione dei transiti per scadenza periodo di retention

Il periodo di retention è il periodo massimo di tempo che un transito non costituente presunta infrazione, può essere mantenuto su file-system; scaduto tale tempo, il transito va eliminato. Possono essere eliminati per retention solamente i transiti che si trovano nei seguenti stati:

- Transiti in velocità media, targa non letta o non rilevata;
- Transiti in velocità media, targa letta ma non ancora accoppiata.

La verifica dei tempi di scadenza della retention viene effettuata dal Server Centrale. Periodicamente, viene verificata la presenza di transiti scaduti, quindi, per tutti quelli di cui non è stata scaricata l'immagine, viene inviato un messaggio alla UEL per richiederne la cancellazione. La UEL, ricevuto il messaggio, provvede alla cancellazione logica del transito, quindi invia un messaggio

di conferma al Server Centrale, il quale, provvederà egli stesso alla cancellazione del transito in suo possesso.

Nel caso l'immagine associata ad un transito scaduto sia già presente sul Server Centrale, non verrà inviato nessun messaggio alla UEL; in questo caso, infatti, il file sarà stato già eliminato durante lo scenario di invio dell'immagine.

La figura seguente descrive i passi della cancellazione per retention da parte del Server Centrale.

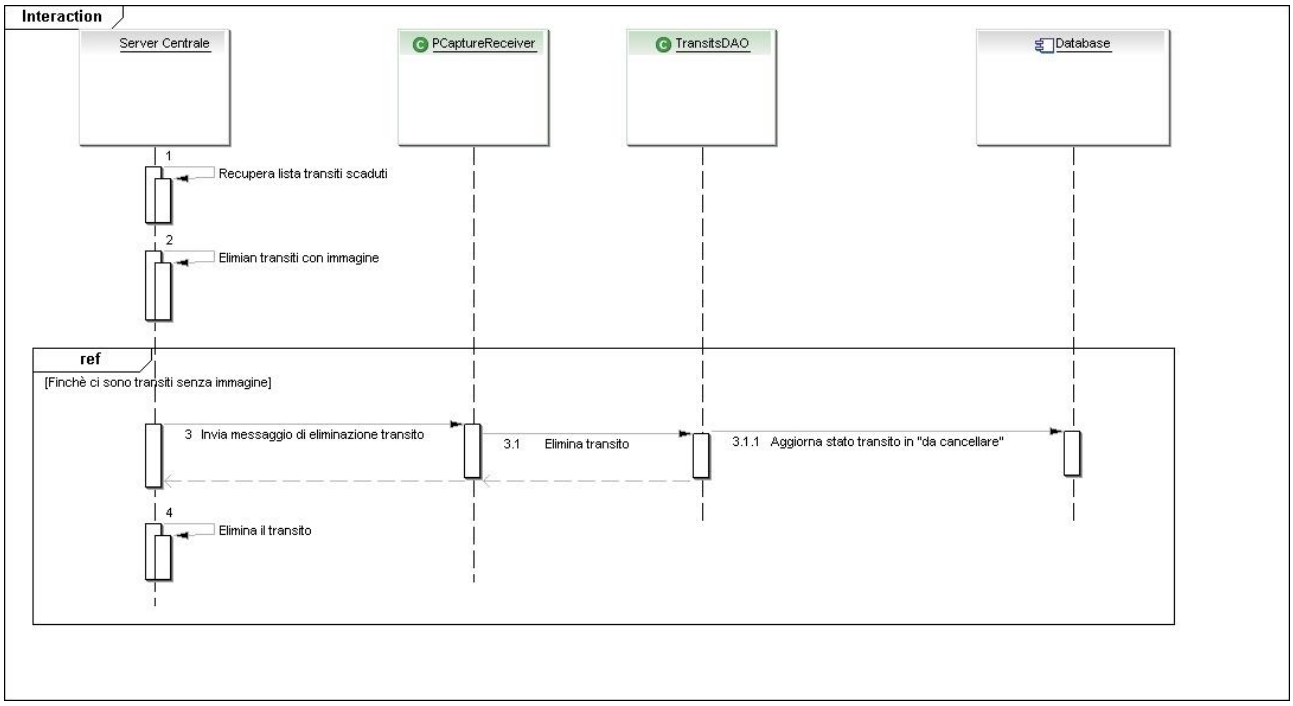

Figura 33 - Cancellazione per retention, diagramma di sequenza

La cancellazione di un transito avviene sempre a seguito di un opportuno comando del Server Centrale. L'unica eccezione è costituita dal componente CheckRetentionTimer, che si occupa di eliminare tutti quei transiti in velocità media, non ancora inviati al Server Centrale, che si trovano sulla UEL da un tempo maggiore a quello di retention. Ovviamente ciò si verifica nel caso in cui ci sia una caduta di collegamento con il Server Centrale.

In questo caso, i transiti non possono essere mantenuti per un periodo maggiore a quello di retention, quindi vengono eliminati. I transiti relativi a servizi di verifica violazioni in velocità istantanea non vengono mai eliminati, poiché costituiscono già una presunta infrazione.

5.11.13 Cancellazione delle cartelle FTP

Il servizio PCapture ad intervalli regolari effettua il controllo delle cartelle FTP . Se tali cartelle sono più vecchie di un certo valore configurabile allora viene cancellata con tutto il suo contenuto.

Sottolineando che nel caso esistono dei file su tali cartelle essi saranno cancellati ed un messaggio di warning scritto nei log.

5.11.14 Cancellazione dei servizi

Il servizio PCapture ad intervalli regolari effettua dei controlli se esistono dei servizi che devono essere eliminati, ciò è possibile solo nei seguenti casi:

- È stato inviato dal server un messaggio di rollback start alla UEL
- Il servizio è stato stoppato e tutti i transiti ad esso associati sono stati inviati al server e/o sono stati cancellati .

Se le precedenti condizioni sono vere il record del servizio verrà cancellato.

5.11.15 Architettura lato Server Centrale: Servizio SCapture

La comunicazione di messaggi e dati relativi ai transiti rilevati dalle URVs viene gestita lato Server Centrale da uno specifico servizio, denominato SCapture. Tale servizio dialoga con il duale presente nella UEL, il PCapture, attraverso lo scambio di messaggi JMS ed in particolare utilizzando l'infrastruttura di messaggi descritta nei capitoli precedenti.

Il servizio SCapture viene avviato durante lo start-up del Server Centrale, e rimane in esecuzione per tutto il tempo in cui il Server è attivo. Esso si occupa di ricevere i messaggi di transito inviati dalla UEL, e di inviare alla UEL stessa messaggi di richiesta immagini o di cancellazione transiti. Inoltre si occupa di inviare i messaggi di avvio e arresto di un servizio di verifica violazioni(manca il collegamento con le WA nella figura seguente). La figura seguente descrive l'architettura logica interna del servizio SCapture.

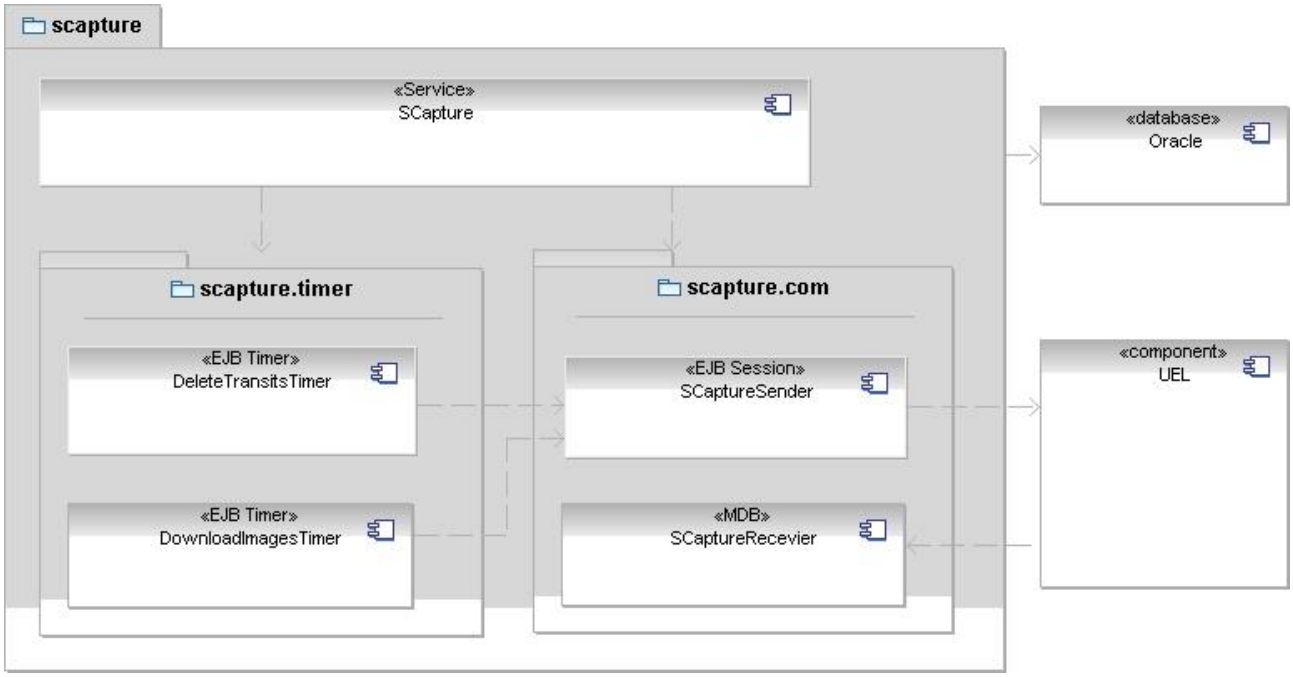

Figura 34 - Servizio SCapture, diagramma dei componenti

Il servizio SCapture è costituito dai seguenti componenti:

- SCapture, è il servizio vero e proprio, gestisce gli altri componenti;
- SCaptureSender (EJB Session), si occupa di inviare messaggi di richiesta alla UEL;
- SCaptureReceiver (MDB), si occupa di ricevere messaggi dalla UEL;
- DeleteTransitsTimer (EJB Timer), si occupa di eliminare dalle UEL i transiti in attesa di cancellazione;
- DownloadImagesTimer (EJB Timer), si occupa di scaricare dalle UEL le immagini richieste da altri servizi.
- RollbackServiceTimer (EJB Timer), si occupa di effettuare lo stop dei servizi, che durante la fase di stop sono andati in errore

Tutti i componenti sopra descritti effettuano delle operazioni di lettura e scrittura su database, attraverso l'utilizzo del pattern DAO. Nei paragrafi successivi vengono descritti i componenti di cui sopra e le funzionalità da essi svolte.

5.11.15.1 Avvio del servizio SCapture

Il servizio SCapture viene avviato durante la fase di start-up del Server Centrale, e rimane attivo per tutta l'esecuzione del Server. Il servizio SCapture si occupa di gestire tutti i componenti utilizzati dal servizio stesso, inoltre espone attraverso la sua interfaccia pubblica due metodi utilizzati per

l'avvio e l'arresto di un servizio di verifica violazioni. La figura successiva descrive il diagramma delle classi del servizio SCapture.

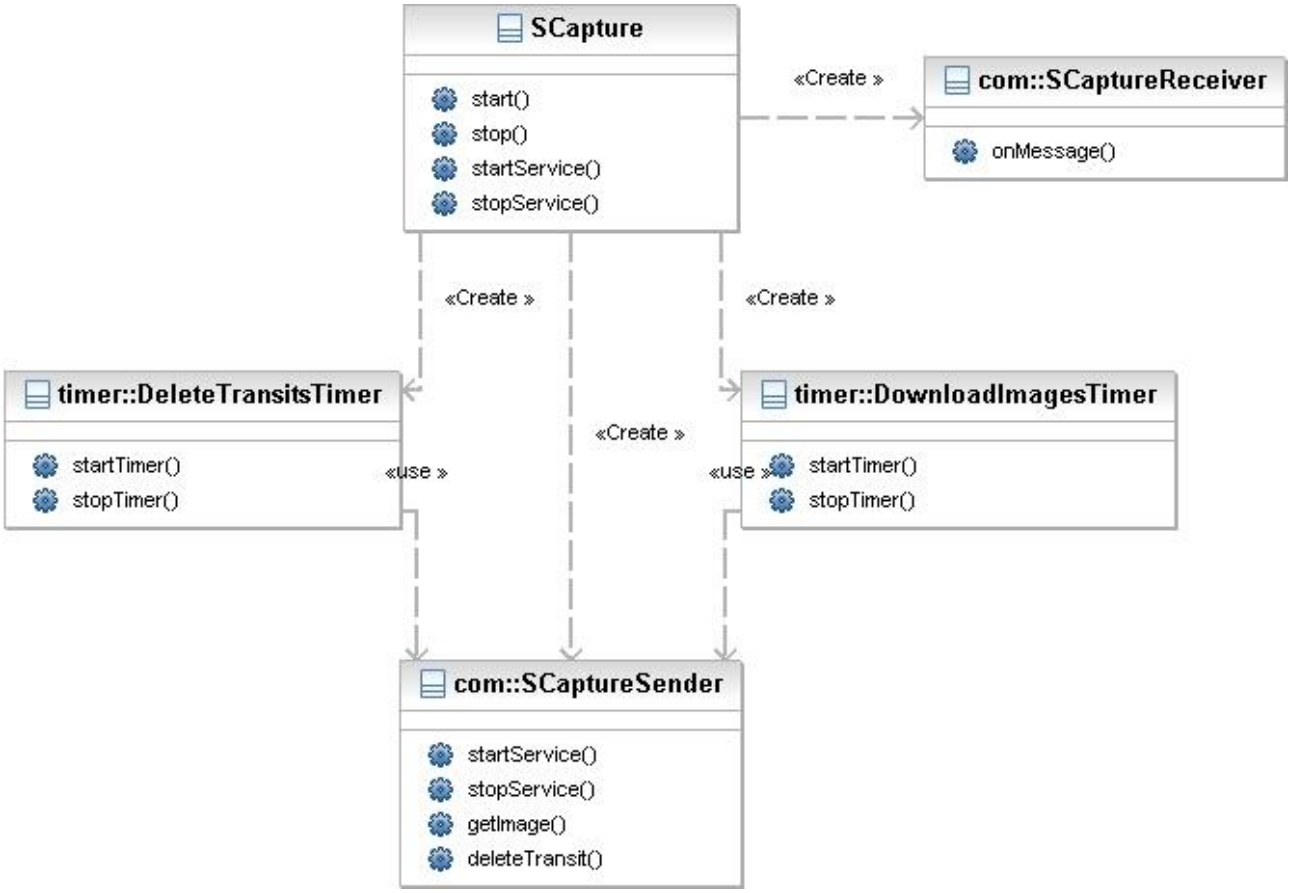

Figura 35 - Servizio SCapture, diagramma delle classi

Il componente SCapture si occupa di creare tutti gli altri componenti, in particolare avvia la schedulazione dei componenti timer. Attraverso la sua interfaccia pubblica, permette agli altri servizi del Server Centrale di avviare ed arrestare un servizio di verifica violazioni.

L'avvio di un servizio di verifica violazioni differisce nel caso in cui venga avviato in modalità istantanea o media; questo secondo caso merita particolare attenzione poiché coinvolge il Server Centrale e due differenti UEL. I paragrafi successivi descrivono i due possibili casi.

5.11.15.2 Servizio di verifica violazioni in velocità istantanea

L'avvio di un servizio di verifica violazioni viene richiesto da un operatore PS attraverso l'applicazione Web di amministrazione dedicata. Per l'avvio del servizio, l'operatore PS dovrà indicare per quale sistema periferico intende avviare il servizio, ed impostare i nuovi limiti di velocità da associare alle singole categorie.

Il server invierà la richiesta al sistema periferico specificando i limiti di velocità settati dall'operatore, il tipo di servizio (pre o post assegnato) e il valore della retention. La figura seguente mostra i passi eseguiti durante l'avvio di un servizio di verifica violazioni in istantanea.

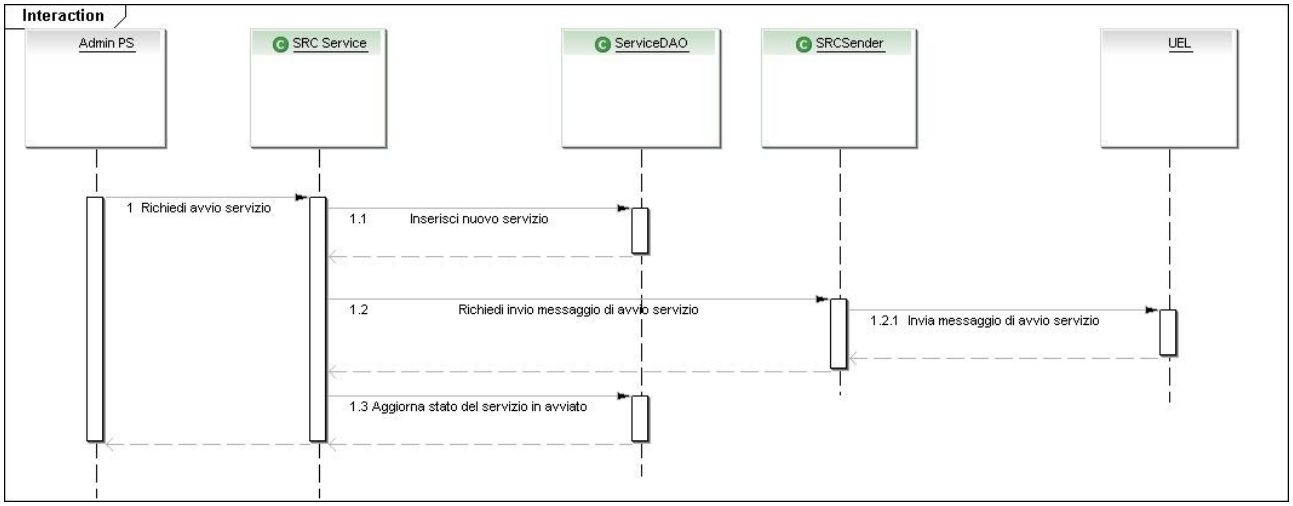

Figura 36 - Avvio di un servizio in istantanea

La UEL si occuperà di settare i limiti di velocità per le URVs, quindi aggiornerà il suo stato operativo. In caso di successo invierà un messaggio di conferma al Server Centrale.

Lo stop di un servizio di verifica violazioni in istantanea viene descritto dalla figura seguente.

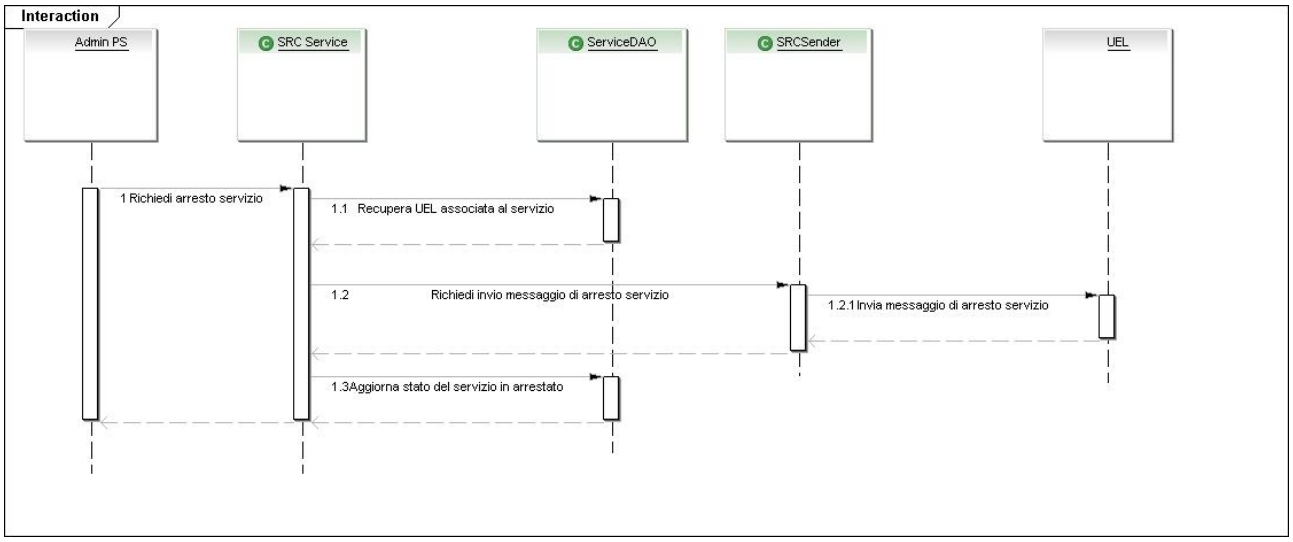

Figura 37 - Arresto di un servizio di verifica violazioni in istantanea

Alla richiesta di arresto di un servizio di verifica violazioni, lo SCapture Service invia un messaggio alla UEL. La UEL aggiornerà il proprio stato, informando il relativo PCapture Service di non collezionare più i transiti inviati dalle URVs.

Ricevuto il messaggio di conferma dalla UEL, lo SCapture aggiornerà lo stato del servizio di

verifica violazioni in "arrestato".

5.11.15.3 Cancellazione dei transiti (lato Server Centrale)

Come descritto nei paragrafi precedenti, il Server Centrale può richiedere la cancellazione di immagini alle UEL. Il compito di recuperare la lista delle immagini da cancellare viene svolto dal componente DeleteTransitsTimer. Così come per la UEL, la cancellazione di un transito sul Server Centrale avviene in modo logico. La cancellazione fisica viene effettuata proprio dal componente DeleteTransitsTimer.

Tale componente, periodicamente, recupera la lista dei transiti da cancellare per ogni UEL, quindi invia, un messaggio alla UEL contenente la lista degli identificativi dei transiti su cui richiedere la cancellazione. Dopo aver avuto un messaggio di conferma, elimina in maniera logica il transito dalla base dati del Server Centrale. La figura seguente mostra i passi eseguiti dal DeleteTransitsTimer.

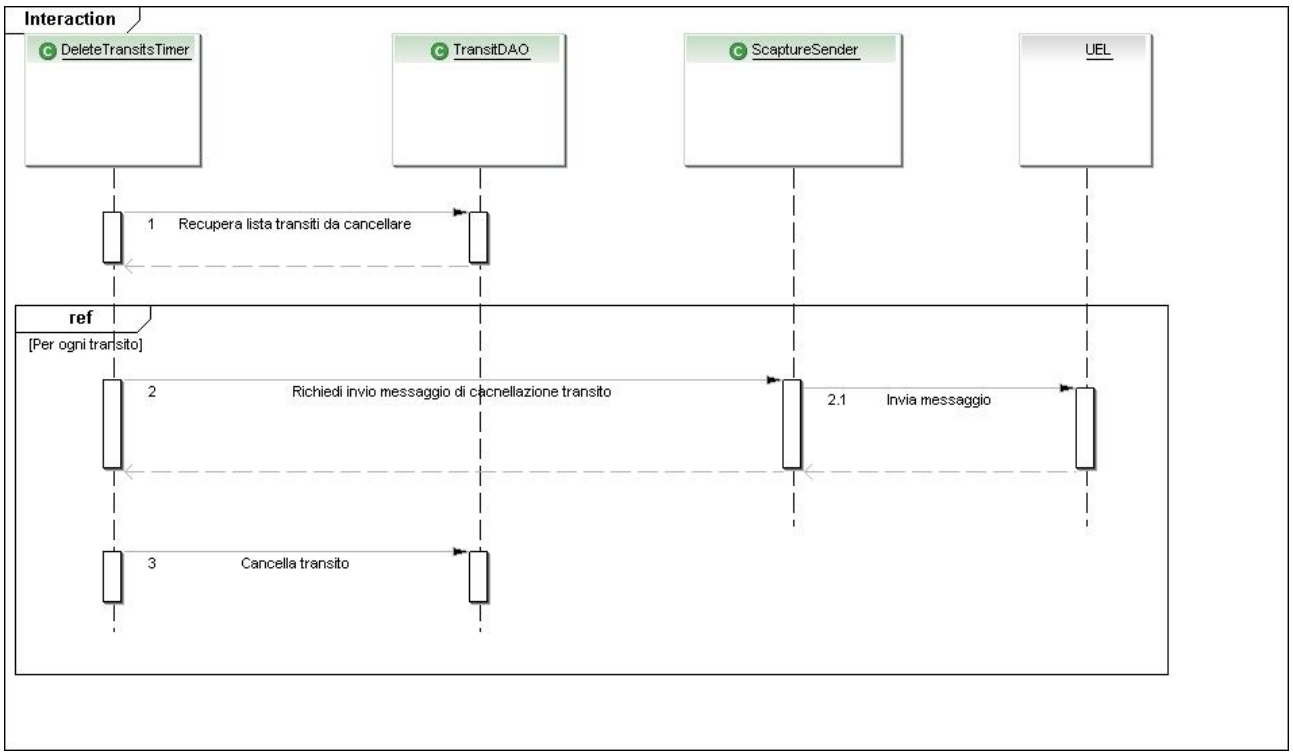

Figura 38 - Cancellazione dei transiti (Server Centrale)

5.11.15.4 Download delle immagini

Il servizio SCapture si occupa del download delle immagini dalla UEL, a seguito

dell'accoppiamento dei transiti in velocità media.

E' presente uno specifico componente, il DownloadImagesTimer, che si occupa per ogni UEL di recuperare la lista dei transiti in attesa di download, crea un messaggio contenente tale lista ed lo invia alla UEL. La figura seguente mostra le operazioni svolte dal componente DownloadImagesTimer..

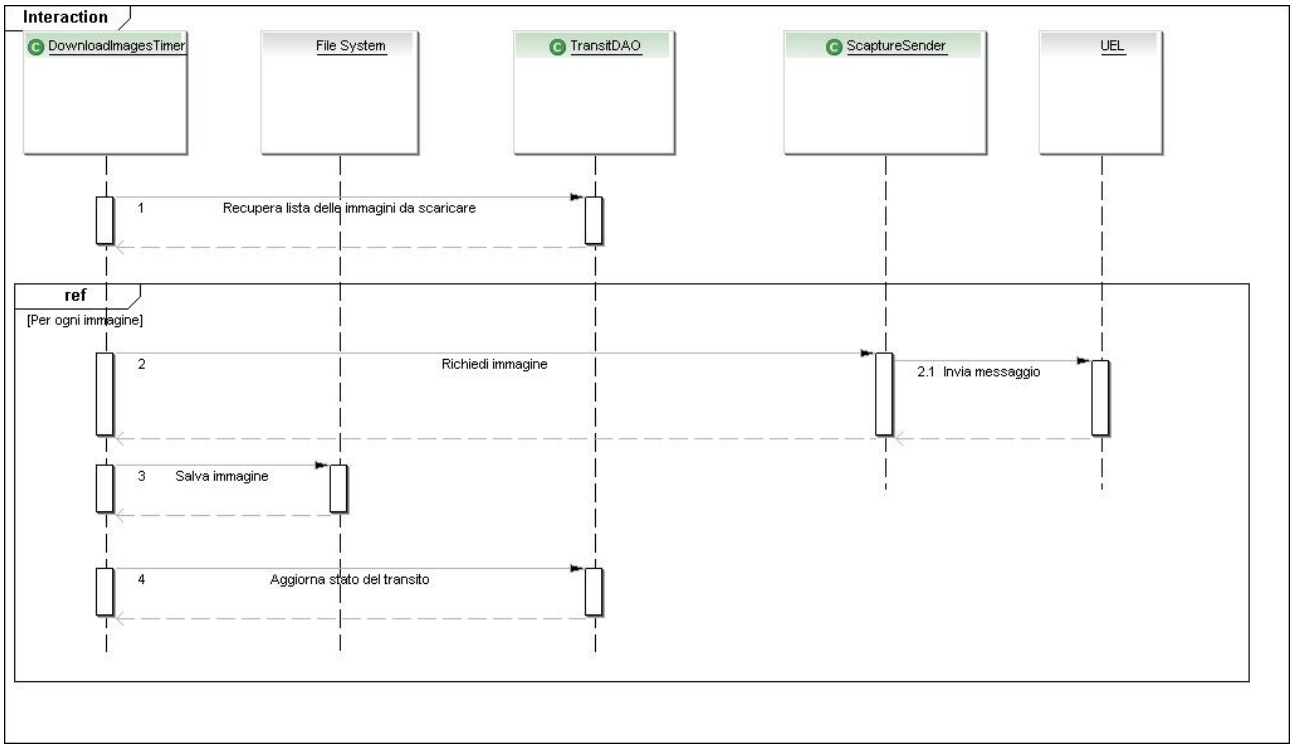

Figura 39 - Download delle immagini (Server Centrale)

5.11.15.5 Download delle immagini su richiesta operatore PS

Gli operatori di PS possono richiedere, durante la lavorazione, il download di immagini relative a transiti con targa non letta/non rilevata. Una volta scaricate le immagini, queste vengono visualizzate direttamente all'interno dell'applicazione Web.

E' prevista una modifica all'applicazione Web Ticket PS, per permettere, in caso di immagini richieste relative ad UEL re ingegnerizzate, il download delle immagini stesse. Le richieste verranno sempre effettuate al PCapture Service, e da questi delegate al componente PCaptureSender. Per i dettagli del flusso dei messaggi, fare riferimento al paragrafo precedente.

5.11.15.6 Stop dei servizi in caso di UEL non raggiungibile

Quando un operatore PS richiede lo stop di un servizio mediante la Web Application PS la UEL potrebbe essere non raggiungibile. In questo caso lo stato del servizio viene settato in "error stopping".

Nel servizio SCapture è presente un componente che ad intervalli regolari controlla se esistono servizi in tale stato. Se ciò è vero invia un messaggio di "rollback stop" alla UEL associata a tale servizio contenete anche la data di stop del servizio. La UEL appena riceve tale messaggio pone lo stato del servizio in "stopped" ed eliminata tutti i transiti non inviati al server antecedente alla data di stop.

5.12 Gestione dei dati statistici: PStatistics Service

Il PStatistics Service è il servizio che si occupa di aggregare le statistiche puntuali su base temporale di 5 minuti, un'ora e un giorno, e di inviare i dati aggregati al Server Centrale.

Il PStatistics Service dialoga con il Server Centrale attraverso l'infrastruttura di messaggistica; sul Server Centrale è presente un componente duale del PStatistics Service che si occupa di interloquire con questo (SStatistics Service).

La figura seguente mostra l'architettura logica interna del servizio PStatistics.

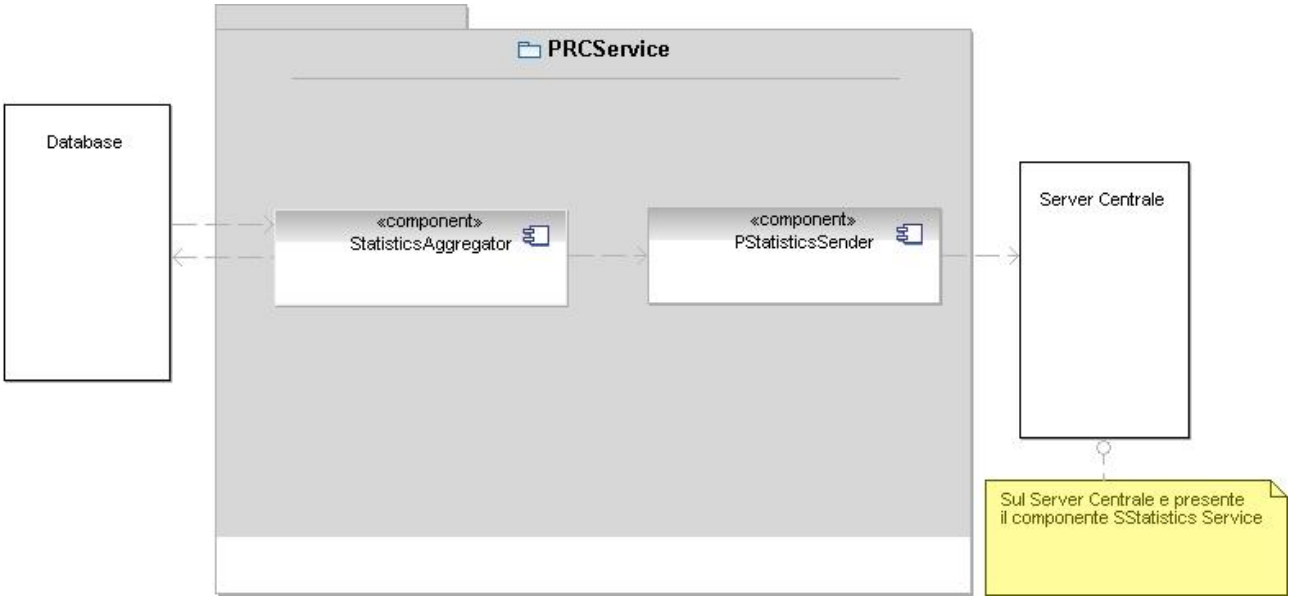

Figura 40 - Architettura del servizio PStatistics

Come è possibile verificare dalla figura precedente, sono previsti solamente due componenti:

- StatisticsAggregatorTimer, si occupa di aggregare i dati statistici puntuali, per intervalli temporali di 5 minuti, un'ora e un giorno.
- PStatisticsSender, invia i dati statistici aggregati al Server Centrale.

Lo StatisticsAggregatorTimer, ogni 5 minuti, preleva i dati statistici di dettaglio e crea le relative statistiche aggregate, quindi richiede al componente PStatisticsSender di inviarle al Server Centrale.

L'invio dei dati statistici di dettaglio per il calcolo delle statistiche in velocità media non è più previsto. Tali informazioni vengono inviate dal servizio PCapture, sia che la UEL sia in Verifica Violazioni o in stato Disponibile. Il Server Centrale, sulla base dell'identificativo di servizio specificato da tali dati, sarà in grado di capire se si tratta di dati da utilizzare per fini statistici o anche per fini di verifica violazioni. Fare riferimento All'appendice A per informazioni relative alle

statistiche collezionate.

6 DEPLOYMENT VIEW

E' prevista una fase in cui sarà realizzata una doppia gestione delle UEL disponibili, suddivise tra quelle che utilizzano la vecchia release e quelle che utilizzano la nuova versione (UEL 2.0). Poiché la reingegnerizzazione della UEL prevede che una parte del Server Centrale sia oggetto di modifiche, nella fase di doppia gestione coesisteranno la versione originale del Server Centrale ed una nuova versione che si occuperà della sola comunicazione con le UEL 2.0.

A livello di deployment, tutto ciò si traduce nell'aggiunta di un nuovo server, contenete la nuova versione adibita al colloquio con le UEL 2.0 (denominato server di pre-produzione), affiancato al preesistente Server Centrale (denominato server di produzione), con il quale condividerà il database. Le UEL 2.0 comunicheranno con il nuovo server, mentre le UEL con versione del software precedente colloquieranno con il server originale. Questo ultimo, inoltre, continuerà a svolgere tutte quelle operazioni non strettamente correlate alla comunicazione con le UEL (flusso violazioni, MCTC, export PS). Il database del Server Centrale resterà unico e verrà condiviso sia dal vecchio server che dal nuovo.

L'architettura di rete sarà divisa nei seguenti nodi:

1. Server di produzione: il server fisico contenente l'intera versione originale del sistema (server centrale ed applicazioni web). Tale server, già presente in ambiente di produzione, sarà responsabile della comunicazione con le vecchie UEL e della gestione di tutti quei processi, estranei alla comunicazione con le UEL, responsabili dell'elaborazione dei dati presenti sul database. Su tale macchina, inoltre, resterà presente il database centralizzato che sarà utilizzato con centro di scambio di dati tra il nodo di produzione e quello di pre-produzione. Ad esso verranno collegate le vecchie UEL;
2. Server di pre-produzione: il server fisico dove verrà installata la versione, del nuovo server, necessaria per la comunicazione con le nuove UEL. Su tale macchina sarà dunque installato l'Application Server JBoss, contenente la nuova versione del server, ed il relativo provider JMS. Inoltre, come già detto, sarà collegato al database presente nel server di produzione. Ad esso saranno collegate le nuove UEL;
3. UEL: Server fisici dove resterà installata la versione originale delle UEL, con database locale MySql 3. Tali macchine colloquieranno con il server di produzione;
4. UEL 2.0: Server fisici dove verrà installata la nuova versione della software della UEL, con database locale MySql 5. Tali macchine colloquieranno con il server di pre-produzione;

5. Client Web: Personal Computer utilizzati dagli utenti del sistema per collegarsi alle applicazioni web presenti sul server di produzione.

La figura seguente mostra l'architettura appena descritta:

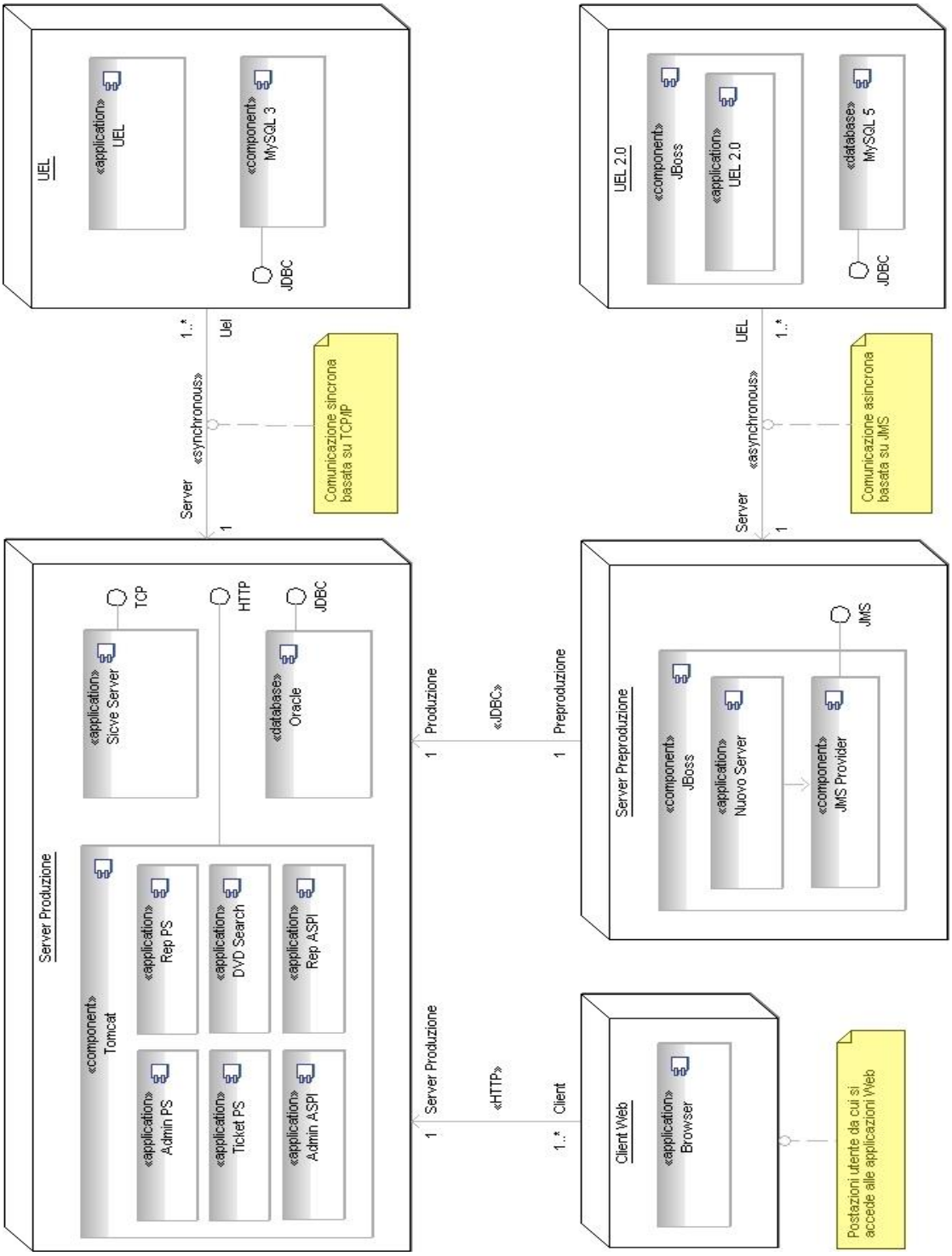

Figura 41 - Diagramma di deployment

6.1 Adattamento del Server e delle applicazioni Web

Come specificato precedentemente, il vecchio Server e le applicazioni Web necessitano di un adattamento per permettere l'interazione degli utenti con le nuove UEL. In particolare verranno eseguiti i seguenti aggiornamenti:

Server Centrale → Verrà modificato il thread che si occupa di scaricare le immagini dalle UEL, al fine di evitare che tali richieste vengano effettuate anche alle nuove UEL;

Applicazioni Web → Verranno modificate delle pagine JSP per permettere alle applicazioni Web di interagire con le nuove UEL, oltre che con le vecchie UEL.

DB → E' prevista la creazione di una nuova tabella sul database Oracle, che permetta di distinguere tra UEL nuove ed UEL vecchie.

Le modifiche sopra descritte prevedono la sostituzione delle pagine JSP all'interno delle applicazioni Web. Per quanto riguarda il Server Centrale, verrà sostituito l'archivio(sicve.server.jar) contenete i file compilati.

7 DATA VIEW

L'accesso al database viene effettuato dall'applicazione attraverso uno strato applicativo (layer) dedicato. Tale strato (strato di persistenza), è stato progettato applicando un semplice quanto efficace pattern denominato DAO (Data Acces Object), il quale permette di disaccoppiare la logica di business dalla logica di accesso ai dati.

I vantaggi principali offerti da tale pattern sono rappresentati dalla semplicità della lettura del codice e dalla robustezza che implicitamente ne deriva; il tutto si traduce, inevitabilmente, in un aumento di manutenibilità e scalabilità del codice stesso.

Una realizzazione tipicamente a componenti come quella prevista dal DAO consente inoltre la rapida integrazione con il pool di connessioni alla base dati, come spiegato nel seguito.

I due diagrammi a seguire, per quanto schematici, sono esaustivi nel definire il pattern utilizzato, e rendono chiare le interazioni previste fra i vari componenti del sistema, esaltando nel contempo le caratteristiche di semplicità implicite nel pattern stesso. In particolare, l'immagine sotto riportata mostra le classi previste dal pattern DAO dove ad un BusinessObject (un oggetto che si occupa di eseguire operazioni di logica di business) non è consentito l'accesso diretto ai dati, ma dovrà affidarsi ad un oggetto responsabile delle operazioni da compiere sui dati, consentendo il disaccoppiamento delle responsabilità.

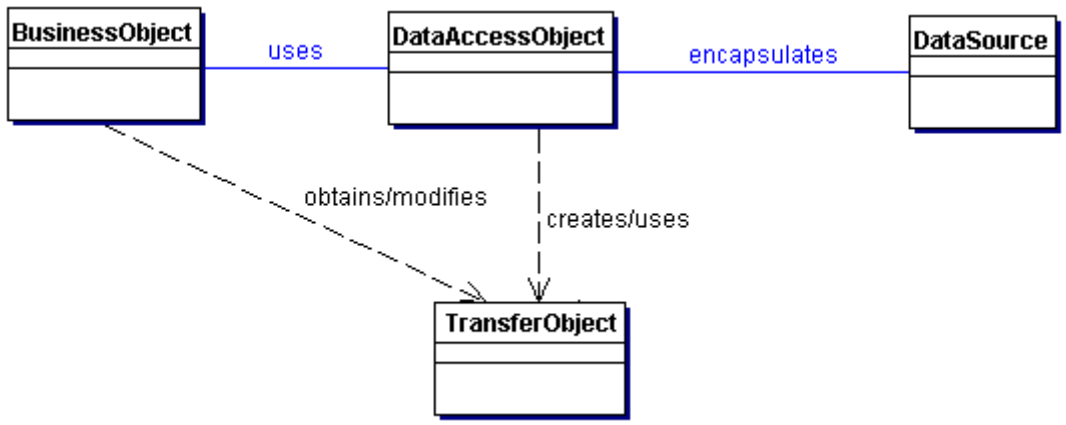

Figura 42 - Pattern DAO, Class Diagram

Il diagramma di sequenza sotto riportato mostra invece le tipiche interazioni previste fra le varie classi definite nel diagramma precedente.

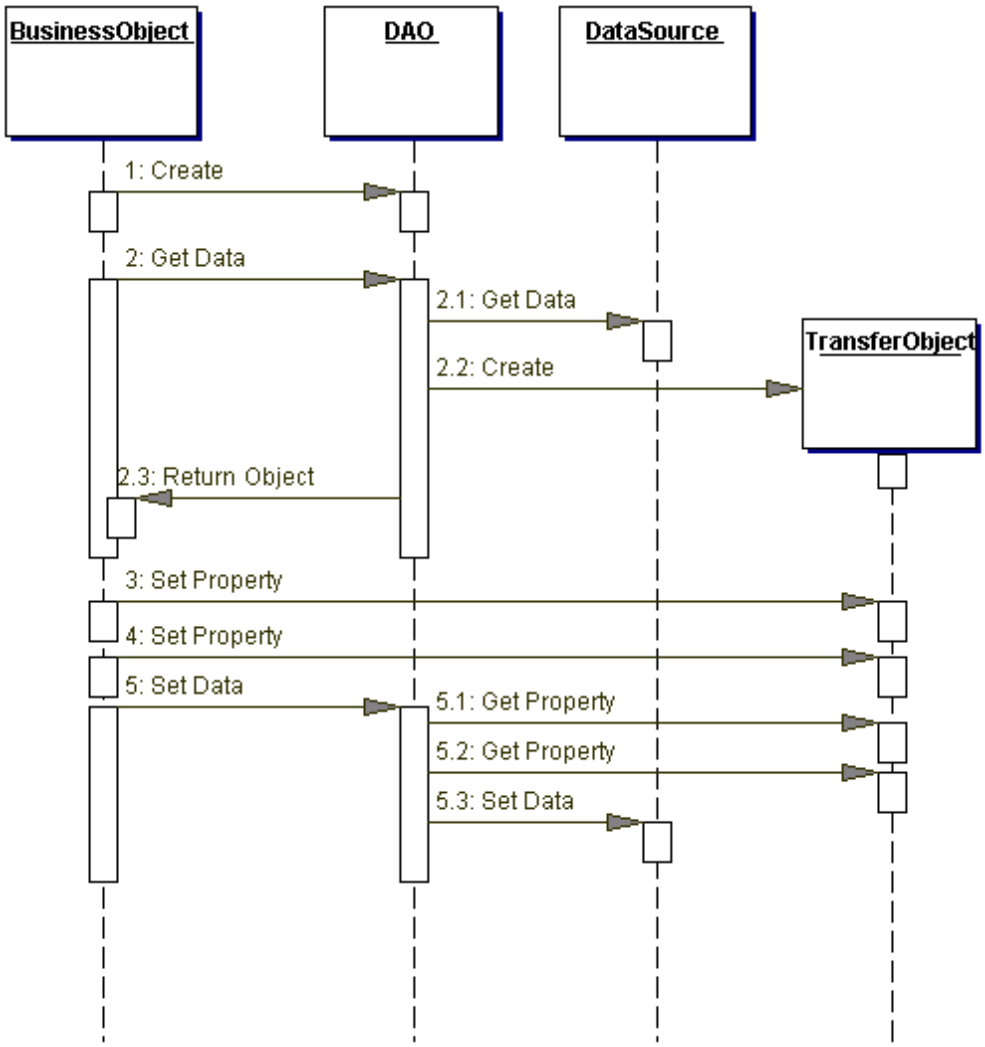

Figura 43 - Pattern DAO, Sequence Diagram

In particolare, la reingegnerizzazione della UEL prevede la definizione, per ogni tabella presente nel database, la realizzazione di un oggetto DAO e delle relative classi di supporto. Ciò permette di centralizzare l'accesso ai dati di una singola tabella all'interno di una singola classe (classe DAO), in modo tale che due differenti oggetti di business, che necessitino di effettuare la stessa operazione sui dati, condividano lo stesso modo di accedere ai dati stessi.

Per quanto riguarda il pool di connessioni, ricordiamo che la reingegnerizzazione della UEL prevede la sostituzione dell'attuale pool di connessione proprietario con il pool integrato nell'Application Server utilizzato (JBoss). Tale scelta si riflette nella garanzia di utilizzare un prodotto affidabile e maturo, caratteristiche essenziali per un sistema che prevede un uso intenso del database stesso.

Il database che verrà utilizzato è MySql, in particolare la release 5 (ultima release stabile rilasciata

alla data attuale), la quale coniuga le ben note e riconosciute caratteristiche di affidabilità, robustezza ed alte prestazioni del database stesso, con il supporto alle operazioni transazionali e con la disponibilità di una versione open source che offre le stesse caratteristiche e potenzialità della versione Enterprise. Ricordiamo che l'attuale database in uso sulle UEL, MySql release 3, non fornisce supporto alla transazioni, il che determina, seppur casualmente, perdita di dati durante le operazioni di accesso ai dati, con conseguenze facilmente immaginabili.

Fare riferimento al modello dei dati della UEL [3] per informazioni di dettaglio sullo schema del database della UEL.

8 SECURITY VIEW

La comunicazione tra Server Centrale ed UEL necessita dell'utilizzo di alcuni standard di sicurezza, al fine di evitare l'intromissione di terze parti sul canale di comunicazione stesso e per permettere a Server Centrale ed UEL di verificare vicendevolmente la propria identità. Nel seguito vengo descritti i protocolli e gli standard utilizzati, nonché il supporto fornito dall'Application Server (JBoss) per l'implementazione degli stessi

8.1 Standard X.509

Lo scopo principale dei protocolli di autenticazione è permettere alle parti coinvolte in una qualsiasi comunicazione di verificare vicendevolmente le proprie identità e di scambiarsi le chiavi di sessione. Nel caso di comunicazione tra UEL e Server, si rende necessario permettere, ad entrambi i sistemi, di verificare l'identità del proprio interlocutore e di cifrare il canale di comunicazione utilizzato.

La comunicazione tra UEL e Server viene resa sicura attraverso l'implementazione dello standard X.509 (Raccomandazione ITU-T X.509). Scopo di tale standard è quello di fornire servizi di autenticazione tramite la directory X.500; tali argomenti esulano comunque dal contesto del sistema. Quello che interessa dello standard X.509 è la definizione di protocolli di autenticazione basati sull'uso di certificati a chiave pubblica.

X.509 è uno standard importante, in quanto la struttura dei certificati e i protocolli definiti in X. 509 vengono utilizzati in vari contesti, quali ad esempio l'autenticazione di utenti sul Web, la protezione della posta elettronica (Secure/Multipurpose Internet Mail Extensions - S/MIME), la protezione IP (IP Security), nei protocolli Secure SocketsLayer/Transaction Layer Security (SSL/TLS) e, per ultimo ma non meno importante, nella firma digitale. L'ultima versione disponibile dello standard X.509 è la versione 3, rilasciata nel 1995 riveduta nel 2000.

Come già detto, X.509 si basa sull'uso della crittografia a chiave pubblica e delle firme digitali. Lo standard non prevede l'uso di un algoritmo determinato, ma viene comunque consigliato l'impiego di RSA.

8.1.1 Certificati a chiave pubblica

Il cuore del meccanismo X.509 è il certificato a chiave pubblica di ogni utente, dove, qui e nel

seguito, per utente si intende sia una persona fisica che un qualsiasi dispositivo o applicativo. Tali certificati vengono creati da una qualche autorità di certificazione, la quale ha inoltre il compito della distribuzione dei certificati stessi. La struttura di un certificato digitale X.509 v3 è la seguente:

1. Certificato
    1. Versione
    2. Numero di serie
    3. Identificatore algoritmo di firma
    4. Nome emettitore
    5. Periodo di validità
        1. Non prima
        2. Non dopo
    6. Nome soggetto
    7. Informazioni sulla chiave pubblica del soggetto
        1. Algoritmo per l'utilizzo della chiave pubblica
        2. Chiave pubblica
    8. Identificatore univoco dell'emettitore (facoltativo)
    9. Identificatore univoco del soggetto (facoltativo)
    10. Estensioni (facoltativo)
        1. .
2. Algoritmo di firma del certificato
3. Firma del certificato

L'autorità di certificazione firma il certificato con la propria chiave privata. Se l'utente conosce la chiave pubblica corrispondente, può verificare che il certificato firmato dall'autorità di certificazione è valido.

8.2 Procedura di autenticazione UEL – Server Centrale

Come già detto, la comunicazione tra UEL e Server centrale utilizza lo standard X.509 per effettuare l'autenticazione delle parti e per la cifratura del canale di comunicazione stesso. Il protocollo di autenticazione utilizzato prevede una autenticazione forte (mutua), cioè sia la UEL che il Server centrale inviano richiedono al proprio interlocutore di dimostrare la propria identità, attraverso l'invio del proprio certificato. Tale procedura di autenticazione viene utilizzata durante le fasi di handshake necessarie all'instaurarsi di una connessione SSL tra UEL e Server centrale.

L'implementazione della connessione SSL e delle relative procedure di autenticazione viene fornita come funzionalità integrata dell'Application Server (JBoss), attraverso l'utilizzo di JSSE (Java Secure Socket Extension) per quanto riguarda il protocollo SSL, ed attraverso lo standard JAAS (Java Authentication and Authorization Service) per la verifica delle credenziali.

In pratica, attraverso la comunicazione SSL si verifica l'attendibilità delle parti e si cifra il canale di comunicazione, attraverso il servizio JAAS si verificano le responsabilità del proprio interlocutore; in pratica viene verificato quali operazioni ha diritto ad eseguire il proprio interlocutore.

Lo standard JSSE prevede l'utilizzo di un particolare oggetto, il keystore, il quale può essere pensato come un raccoglitore di quantità di sicurezza (certificati digitali). Attraverso il keystore, un utente può definire, tramite lo standard JSSE, informazioni inerenti la propria identità (keystore vero e proprio) e definire la lista dei certificati ritenuti attendibili (truststore). Appare chiaro che, durante le fasi di handshake per l'instaurarsi della connessione SSL, la UEL viene autenticato correttamente se la propria identità (certificato) è contenuta all'interno del truststore del Server centrale, e viceversa. Inoltre è possibile ritenere attendibile un utente se l'autorità di certificazione che ha firmato il suo certificato è la stessa che ha firmato il proprio certificato.

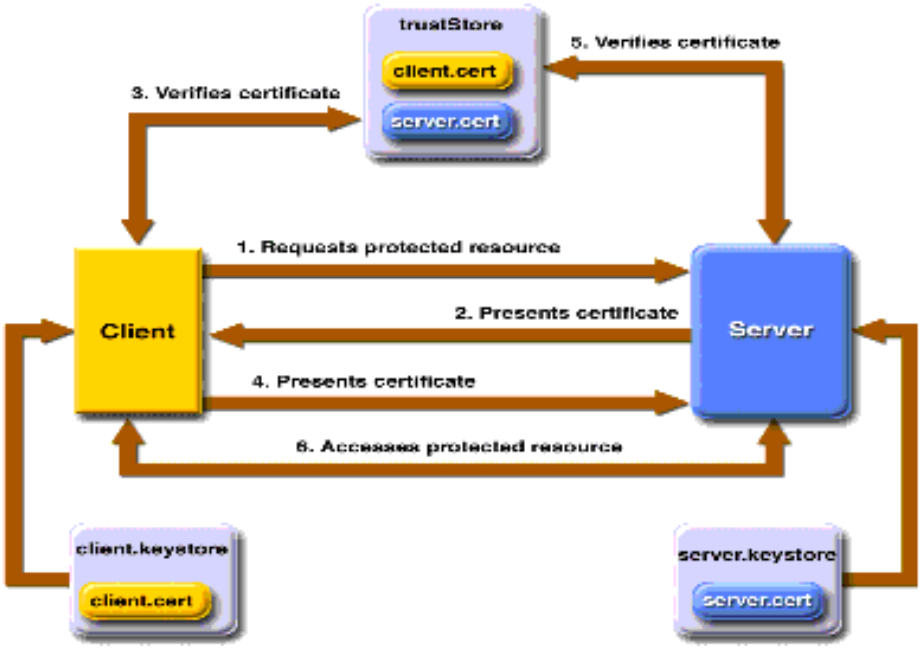

Figura 44 - Mutua autenticazione basata sui certificati

Una volta instaurata la connessione SSL, attraverso il servizio JAAS, viene effettuata una verifica sulle credenziali della UEL; viene verificato, tramite il common name del soggetto, che la UEL sia conosciuta al server centrale, che l'indirizzo IP sia lo stesso della macchina client da cui si effettua la connessione, e per ultimo, si verifica la tipologia del certificato. Infatti non sarà possibile per una

UEL, se ci si connette con un certificato di attivazione, eseguire altre operazioni all'infuori di quelle proprie dell'attivazione stessa; stessa cosa avverrà per il certificato di connect, il quale permetterà di eseguire le operazioni rimanenti.

8.3 Procedura di firma e cifratura dei dati sensibili

Le immagini raccolte dalla UEL possono essere mantenute su File-System solamente se opportunamente cifrate. Inoltre è previsto che ogni immagine inviata dalla UEL al Server Centrale sia firmata, per permettere al Server Centrale di verificare l'autenticità dell'immagine stessa.

E' previsto un opportuno certificato di firma, differente da quello utilizzato per l'autenticazione, che viene utilizzato per la firma delle immagini e per la cifratura.

Le operazioni di firma e cifratura di una immagine vengono svolte nel momento in cui la UEL, recuperata l'immagine stessa dal Server FTP, salva l'immagine all'interno della propria directory di lavoro. Le operazioni svolte dalla UEL sono le seguenti:

- firma il file utilizzando il certificato di firma (standard PKCS#7);
- genera una chiave casuale a 128 bit;
- cripta il file firmato con l'algoritmo DES utilizzando la chiave generata precedentemente;
- sostituisce il file sorgente con il file PKCS#7 cifrato;
- cripta con l'algoritmo RSA la chiave generata utilizzando la chiave pubblica del certificato di firma;
- salva, sul database, la chiave casuale criptata;
- cripta con l'algoritmo RSA la targa utilizzando la chiave pubblica del certificato di firma;
- salva, sul database, la targa criptata.

Quando l'immagine deve essere inviata al Server Centrale, vengono svolte le seguenti operazioni:

- Recupera dal database il percorso e il nome file dell'immagine;
- Recupera da file system l'immagine;
- Decifra la chiave casuale memorizzata sul database utilizzando la chiave privata del certificato di firma
- Decifra l'immagine la chiave casuale;
- Invia al Server Centrale l'immagine firmata sul canale di comunicazione sicuro;
- Il Server Centrale verifica, attraverso la firma dell'immagine, la validità della stessa

8.4 Librerie di terza parti

Per la gestione della sicurezza, vengono utilizzate due librerie di terze parti, rilasciate come open source. In particolare, le librerie utilizzate sono le seguenti:

- IAIK Provider → Utilizzato per la gestione della Smart-card (Standard PKCS#11);
- Bouncy Castle Provider → Fornisce delle utility per la firma dei file (Standard PKCS#7).

9 APPENDICE A – STATISTICHE AGGREGATE

Vengono descritte in questa appendice le statistiche aggregate calcolate dalla UEL.

Le statistiche generate dall'applicazione UEL si dividono in due tipi, quelle generate mentre l'applicazione è in modalità locale e quelle generate quando applicazione è in modalità disponibile o verifica violazioni.

Le statistiche generate in modalità locale vengono avviate quando il sistema non è in comunicazione con il Server. Questo tipo di statistiche non vengono mai inviate al Server stesso e sono consultabili solo dall'utente Manutentore.

Le statistiche generate in modalità disponibile vengono avviate quando il sistema è in comunicazione con il Server. Questo tipo di statistiche viene inviato ogni 5 minuti al Server e scritte su un'apposita tabella.

9.1 Statistiche locali

Tali dati statistici, come già detto, vengono raccolti quando la UEL si trova in modalità Statistiche in locale. Attraverso i dati statistici raccolti, l'operatore manutentore potrà verificare il corretto funzionamento della UEL stessa, utilizzando l'apposita applicazione riservata ad esso.

I dati statistici raccolti in questa fase, e memorizzati nelle tabelle SICVE_ST_GROUPED e SICVE_ST_GROUPED_CLASSES sono i seguenti:

- Numero di veicoli transitati per corsia all'interno di un determinato slot di tempo (memorizzati in SICVE_ST_GROUPED);
- Tempo medio interveicolo per corsia all'interno di un determinato slot di tempo (memorizzati in SICVE_ST_GROUPED);
- Numero di veicoli transitati per corsia e per classe all'interno di un determinato slot di tempo (memorizzati in SICVE_ST_GROUPED_CLASSES);
- Velocità media dei transiti per corsia e per classe all'interno di un determinato slot di tempo (memorizzati in SICVE_ST_GROUPED_CLASSES).

Le statistiche sopra elencate vengono generate sempre ad intervalli prefissati che sono:

- 5 minuti: questo tipo di statistica aggrega i dati puntuali nell'intervallo di 5 minuti precedente;
- 1 ora: questo tipo di statistica aggrega i dati delle statistiche di 5 minuti contenuti nell'ora

precedente;

- 1 giorno: questo tipo di statistica aggrega i dati delle statistiche di 1 ora appartenenti all'intero giorno. Infatti la statistica viene generata allo scoccare della mezzanotte ed è riferita al giorno precedente.

Tutti i dati raccolti in questa fase non vengono mai inviati al Server Centrale, ma vengono cancellati quando la UEL raggiunge lo stato operativo Disponibile, a seguito della fase di attivazione. Come già detto, tali dati servono solo per permettere all'utente manutentore di verificare il corretto funzionamento della UEL stessa.

9.2 Statistiche di traffico

Quando la UEL passa in modalità Disponibile o Verifica Violazioni, vengono prodotte delle statistiche aggregate differenti da quelle prodotte in modalità locale; tali statistiche vengono inoltre inviate al Server Centrale.

Le statistiche generate quando l'applicazione UEL si trova in modalità Disponibile o Verifica Violazioni vengono memorizzate nella tabella SICVE_STAT, in cui sono anche contenuti i dati puntuali dei transiti.

Ogni tipo di statistica viene identificata con un codice univoco condiviso con il Server Centrale. Nei paragrafi successivi descritte le statistiche previste dalla UEL. Tali statistiche vengono create a partire da una tabella di configurazione (SICVE_STAT_TYPE), presente sulla UEL, che può essere modificata per generare altri tipi di statistiche eventualmente necessarie.

9.2.1 Statistica 21 (transiti per corsia, classe e lettura OCR)

La statistica 21 si occupa di:

- Ogni 5 min contare il numero di transiti per corsia, classe e risultato della lettura OCR;
- Ogni ora, unitamente alla scadenza precedente, sommare le statistiche calcolate ogni 5 minuti relative all'ora trascorsa;
- Ed infine giornalmente, in corrispondenza del cambio di data, sommare le statistiche orarie rilevate durante l'arco della giornata

9.2.2 Statistica 22 (transiti per corsia con targa letta)

La statistica 22 si occupa di:

- Ogni 5 min contare il numero di transiti per corsia che hanno risultato della lettura OCR uguale a 0 (Targhe lette);
- Ogni ora, unitamente alla scadenza precedente, sommare le statistiche calcolate ogni 5 minuti relative all'ora trascorsa;
- Ed infine giornalmente, in corrispondenza del cambio di data, sommare le statistiche orarie rilevate durante l'arco della giornata.

9.2.3 Statistica 23 (transiti per corsia elaborati dall'OCR di secondo livello)

La statistica 23 si occupa di:

- Ogni 5 min contare il numero di transiti per corsia elaborati dall' OCR di 2° livello;
- Ogni ora, unitamente alla scadenza precedente, sommare le statistiche calcolate ogni 5 minuti relative all'ora trascorsa;
- Ed infine giornalmente, in corrispondenza del cambio di data, sommare le statistiche orarie rilevate durante l'arco della giornata.

9.2.4 Statistica 24 (transiti per corsia riconosciuti dall'OCR di secondo livello)

La statistica 24 si occupa di:

- Ogni 5 min contare il numero di transiti per corsia che hanno risultato della lettura dell'OCR uguale a 0 (Targhe lette) elaborati dall' OCR di 2° livello;
- Ogni ora, unitamente alla scadenza precedente, sommare le statistiche calcolate ogni 5 minuti relative all'ora trascorsa;
- Ed infine giornalmente, in corrispondenza del cambio di data, sommare le statistiche orarie rilevate durante l'arco della giornata.

9.2.5 Statistica 26 (transiti per corsia con targa non rilevata)

La statistica 26 si occupa di:

- Ogni 5 min contare il numero di transiti per corsia che hanno risultato della lettura OCR uguale a 1 (Targhe non rilevate);

- Ogni ora, unitamente alla scadenza precedente, sommare le statistiche calcolate ogni 5 minuti relative all'ora trascorsa;
- Ed infine giornalmente, in corrispondenza del cambio di data, sommare le statistiche orarie rilevate durante l'arco della giornata.

9.2.6 Statistica 28(transiti per corsia con targa non letta)

La statistica 28 si occupa di:

- Ogni 5 min contare il numero di transiti per corsia che hanno risultato della lettura OCR uguale a 2 (Targhe non lette);
- Ogni ora, unitamente alla scadenza precedente, sommare le statistiche calcolate ogni 5 minuti relative all'ora trascorsa;
- Ed infine giornalmente, in corrispondenza del cambio di data, sommare le statistiche orarie rilevate durante l'arco della giornata.

9.2.7 Statistica 30 (transiti per corsia e classe)

La statistica 30 si occupa di:

- Ogni 5 min contare il numero di transiti per corsia e classe;
- Ogni ora, unitamente alla scadenza precedente, sommare le statistiche calcolate ogni 5 minuti relative all'ora trascorsa;
- Ed infine giornalmente, in corrispondenza del cambio di data, sommare le statistiche orarie rilevate durante l'arco della giornata.

9.2.8 Statistica 31 (transiti scaduti per retention sulla UEL)

La statistica 31 si occupa di:

- Ogni 5 min contare il numero di transiti scaduti;
- Ogni ora, unitamente alla scadenza precedente, sommare le statistiche calcolate ogni 5 minuti relative all'ora trascorsa;
- Ed infine giornalmente, in corrispondenza del cambio di data, sommare le statistiche orarie rilevate durante l'arco della giornata.

9.2.9 Statistica 33 (velocità media per corsia e classe)

La statistica 33 si occupa di calcolare ogni 5 minuti la media della velocità per corsia e classe in cui la velocità sia maggiore di 0.

9.2.10 Statistica 34 (tempo interveicolo per corsia e risultato lettura OCR)

La statistica 34 si occupa di calcolare ogni 5 minuti il tempo interveicolo per corsia e risultato della lettura dell'OCR.

9.2.11 Statistica 35 (transiti riconosciuti dall'OCR di secondo livello)

La statistica 35 si occupa di:

- Ogni 5 min contare il numero di transiti riconosciuti dall'OCR di 2° livello;
- Ogni ora, unitamente alla scadenza precedente, sommare le statistiche calcolate ogni 5 minuti relative all'ora trascorsa;
- Ed infine giornalmente, in corrispondenza del cambio di data, sommare le statistiche orarie rilevate durante l'arco della giornata.

9.2.12 Statistica 36 (numero di transiti)

La statistica 36 si occupa di:

- Ogni 5 min contare il numero di transiti;
- Ogni ora, unitamente alla scadenza precedente, sommare le statistiche calcolate ogni 5 minuti relative all'ora trascorsa;
- Ed infine giornalmente, in corrispondenza del cambio di data, sommare le statistiche orarie rilevate durante l'arco della giornata.

9.2.13 Statistica 37 (transiti per corsia con targa non rilevata)

La statistica 37 si occupa di:

- Ogni 5 min contare il numero di transiti che hanno risultato della lettura OCR uguale a 1 (Targhe non rilevate);
- Ogni ora, unitamente alla scadenza precedente, sommare le statistiche calcolate ogni 5 minuti relative all'ora trascorsa;

- Ed infine giornalmente, in corrispondenza del cambio di data, sommare le statistiche orarie rilevate durante l'arco della giornata.

9.2.14 Statistica 38 (transiti per corsia con targa non letta)

La statistica 38 si occupa di:

- Ogni 5 min contare il numero di transiti che hanno risultato della lettura OCR uguale a 2 (Targhe non lette);
- Ogni ora, unitamente alla scadenza precedente, sommare le statistiche calcolate ogni 5 minuti relative all'ora trascorsa;
- Ed infine giornalmente, in corrispondenza del cambio di data, sommare le statistiche orarie rilevate durante l'arco della giornata.

9.2.15 Statistica 39 (numero violazioni in istantanea per classe)

La statistica 39 si occupa di:

- Ogni 5 min contare il numero di transiti per classe che sono in violazione;
- Ogni ora, unitamente alla scadenza precedente, sommare le statistiche calcolate ogni 5 minuti relative all'ora trascorsa;
- Ed infine giornalmente, in corrispondenza del cambio di data, sommare le statistiche orarie rilevate durante l'arco della giornata.

9.3 Cancellazione statistiche in modalità Disponibile

Tutti i dati raccolti in questa fase vengono inviati ogni 5 minuti al Server Centrale, e vengono cancellati quando sono stati aggregati dalla statistica successiva.

La cancellazione dei dati avviene secondo le seguenti regole:

- I dati puntuali dei transiti vengo cancellati ogni volta che vengono utilizzati per generare statistiche di 5 minuti.
- Le statistiche di 5 minuti vengono cancellate quando sono stata inviate al Server e quando sono state aggregate dalle statistiche orarie.
- Le statistiche orarie vengono cancellate quando sono stata inviate al Server e quando sono

state aggregate dalle statistiche giornaliere.

- Le statistiche giornaliere vengono cancellate quando sono state inviate al Server.

Per capire quando una statistica è stata inviata al Server o aggregata dalla statistiche successiva sono presenti due attributi di tipo booleano nella tabella SICVE_STAT, che sono:

- Sent: per capire se è stata inviata al Server;
- Aggregated: per capire se il dato è stato aggregato.

9.4 Query eseguite per l'aggregazione delle statistiche

Nelle tabelle successive vengono riportate le query eseguite per l'aggregazione delle statistiche. Tali query, come già detto, vengono create dinamicamente dalla tabella SICVE_STAT_TYPE.

9.4.1 Query eseguite per le statistiche di 5 minuti

ID	Query Eseguita
21	SELECT corsia, ocr_result, classe,COUNT(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=1 AND ID_STAT_AGG_PERIOD='0' GROUP BY corsia,ocr_result,classe
22	SELECT corsia,COUNT(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=1 AND ID_STAT_AGG_PERIOD='0' AND ocr_result=0 GROUP BY corsia
23	SELECT corsia,COUNT(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=2 AND ID_STAT_AGG_PERIOD='0' GROUP BY corsia
24	SELECT corsia,COUNT(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=2 AND ID_STAT_AGG_PERIOD='0' AND ocr_result=0 GROUP BY corsia
26	SELECT corsia,COUNT(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=1 AND ID_STAT_AGG_PERIOD='0' AND ocr_result=1 GROUP BY corsia
28	SELECT corsia,COUNT(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=1 AND ID_STAT_AGG_PERIOD='0' AND ocr_result=2 GROUP BY corsia
30	SELECT corsia, classe,COUNT(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=1 AND ID_STAT_AGG_PERIOD='0' GROUP BY corsia,classe
31	SELECT COUNT(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=4 AND ID_STAT_AGG_PERIOD='0'
33	SELECT corsia, classe,AVG(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=1 AND ID_STAT_AGG_PERIOD='0' AND value > 0 GROUP BY corsia,classe

34	SELECT corsia, ocr_result, ROUND((TIME_TO_SEC(MAX(DATA)) - TIME_TO_SEC(MIN(DATA))) / COUNT(VALUE)) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=1 AND ID_STAT_AGG_PERIOD='0' GROUP BY corsia,ocr_result
35	SELECT COUNT(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=2 AND ID_STAT_AGG_PERIOD='0' AND ocr_result=0
36	SELECT COUNT(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=1 AND ID_STAT_AGG_PERIOD='0'
37	SELECT COUNT(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=1 AND ID_STAT_AGG_PERIOD='0' AND ocr_result=1
38	SELECT COUNT(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=1 AND ID_STAT_AGG_PERIOD='0' AND ocr_result=2
39	SELECT classe,COUNT(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=3 AND ID_STAT_AGG_PERIOD='0' GROUP BY classe

9.4.1 Query eseguite per le statistiche orarie

ID	Query Eseguita
21	SELECT corsia, ocr_result, classe,SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=21 AND ID_STAT_AGG_PERIOD='5' GROUP BY corsia,ocr_result,classe
22	SELECT corsia, ocr_result,SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=22 AND ID_STAT_AGG_PERIOD='5' AND ocr_result=0 GROUP BY corsia,ocr_result
23	SELECT corsia,SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=23 AND ID_STAT_AGG_PERIOD='5' GROUP BY corsia
24	SELECT corsia, ocr_result,SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=24 AND ID_STAT_AGG_PERIOD='5' AND ocr_result=0 GROUP BY corsia,ocr_result

26	SELECT corsia, ocr_result,SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=26 AND ID_STAT_AGG_PERIOD='5' AND ocr_result=1 GROUP BY corsia,ocr_result
28	SELECT corsia, ocr_result,SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=28 AND ID_STAT_AGG_PERIOD='5' AND ocr_result=2 GROUP BY corsia,ocr_result
30	SELECT corsia, classe,SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=30 AND ID_STAT_AGG_PERIOD='5' GROUP BY corsia,classe
31	SELECT SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=31 AND ID_STAT_AGG_PERIOD='5'
35	SELECT ocr_result,SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=35 AND ID_STAT_AGG_PERIOD='5' AND ocr_result=0 GROUP BY ocr_result
36	SELECT SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=36 AND ID_STAT_AGG_PERIOD='5'
37	SELECT ocr_result,SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=37 AND ID_STAT_AGG_PERIOD='5' AND ocr_result=1 GROUP BY ocr_result
38	SELECT ocr_result,SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=38 AND ID_STAT_AGG_PERIOD='5' AND ocr_result=2 GROUP BY ocr_result
39	SELECT SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=39 AND ID_STAT_AGG_PERIOD='5'

9.4.2 Query eseguite per le statistiche giornaliere

ID	Query Eseguita
21	SELECT corsia, ocr_result, classe,SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=21 AND ID_STAT_AGG_PERIOD='H' GROUP BY corsia,ocr_result,classe
22	SELECT corsia, ocr_result,SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=22 AND ID_STAT_AGG_PERIOD='H' AND ocr_result=0 GROUP BY corsia,ocr_result

23	SELECT corsia,SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=23 AND ID_STAT_AGG_PERIOD='H' GROUP BY corsia
24	SELECT corsia, ocr_result,SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=24 AND ID_STAT_AGG_PERIOD='H' AND ocr_result=0 GROUP BY corsia,ocr_result
26	SELECT corsia, ocr_result,SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=26 AND ID_STAT_AGG_PERIOD='H' AND ocr_result=1 GROUP BY corsia,ocr_result
28	SELECT corsia, ocr_result,SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=28 AND ID_STAT_AGG_PERIOD='H' AND ocr_result=2 GROUP BY corsia,ocr_result
30	SELECT corsia, classe,SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=30 AND ID_STAT_AGG_PERIOD='H' GROUP BY corsia,classe
31	SELECT SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=31 AND ID_STAT_AGG_PERIOD='H'
35	SELECT ocr_result,SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=35 AND ID_STAT_AGG_PERIOD='H' AND ocr_result=0 GROUP BY ocr_result
36	SELECT SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=36 AND ID_STAT_AGG_PERIOD='H'
37	SELECT ocr_result,SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=37 AND ID_STAT_AGG_PERIOD='H' AND ocr_result=1 GROUP BY ocr_result
38	SELECT ocr_result,SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=38 AND ID_STAT_AGG_PERIOD='H' AND ocr_result=2 GROUP BY ocr_result
39	SELECT SUM(VALUE) AS CNT FROM sicve_stat WHERE DATA<=? AND DATA>? AND ID_STAT_TYPE=39 AND ID_STAT_AGG_PERIOD='H'

10 APPENDICE B – DETTAGLIO TECNICO

I dettagli implementativi sono specificati nei seguenti documenti:

- UEL.ALL.01_Modulo Allarmi
- UEL.ALL.02_Modulo Capture

- UEL.ALL.03_Modulo ConfigurationManager
- UEL.ALL.04_Modulo Controllo Remoto
- UEL.ALL.05_Modulo OCR
- UEL.ALL.06_Modulo PActivationService
- UEL.ALL.07_Modulo ProceduraCopy
- UEL.ALL.08_Modulo PStatistics
- UEL.ALL.09_Modulo Sicurezza
- UEL.ALL.10_Modulo UELManager
- UEL.ALL.11_Modulo URVManager