UNIVERSITÀ DI PISA

Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea Specialistica in Tecnologie Informatiche

Tesi di Laurea Specialistica

OpenSPCoop: un'implementazione della Specifica di Cooperazione Applicativa per la Pubblica Amministrazione Italiana

Candidato: Andrea Poli

Relatori:

Prof. Andrea Corradini

Prof. Tito Flagella

Controrelatore:

Prof. Vincenzo Ambriola

Anno accademico 2004-2005

Sommario

Introduzione 4
1. La cooperazione applicativa nella Pubblica Amministrazione 10
1.1 Storia recente dell'informatica nella Pubblica Amministrazione 11
1.2 Contesto Tecnologico 15
1.2.1 Introduzione al mondo Web Services 15
1.2.2 SOAP 19
1.2.3 WSDL 23
1.2.4 UDDI e LDAP 24
1.3 Il Sistema Pubblico di Connettività e Cooperazione 28
1.3.1 La Porta di Dominio 30
1.3.2 La busta SPCoop 31
1.3.3 Il Registro dei servizi 37
1.3.4 Paradigma di cooperazione basata su eventi 41
1.3.5 Il Componente di Integrazione SIL-to-PdD 42
1.4 Possibili architetture del Componente di Integrazione 44
2. Architettura del sistema OpenSPCoop 50
2.1 Una visione d'insieme di OpenSPCoop 50
2.2 Componenti dell'architettura 51
2.2.1 Porta di Dominio 53
2.2.2 Componente di Integrazione SIL – to – PdD 58
2.2.3 Il Gestore degli Eventi 68
2.2.4 Il cuore SPCoop della porta di dominio: Modulo di Controllo 75
2.2.4.1 Sicurezza 76
2.2.4.2 Validazione busta SPCoop 77
2.2.4.3 Tracciamento 78
2.2.4.4 Gestione dei Riscontri 79
2.2.4.5 Gestione dei Duplicati 85
2.2.4.6 Consegna in Ordine 87
2.2.4.7 Profilo di Collaborazione Sincrono 94
2.2.4.8 Profilo di Collaborazione Asincrono Simmetrico 98
2.2.4.9 Profilo di Collaborazione Asincrono Asimmetrico 104
2.2.5 Il Registro dei Servizi 111
2.3 Casi d'uso esemplificativi 118
2.3.1 Invocazione di un Servizio 118
2.3.1 Pubblicazione / Sottoscrizione di un evento 124
3. Implementazione del prodotto OpenSPCoop 130
3.1 Architettura Software 130
3.2 Tecnologie utilizzate 132
3.2.1 JBoss Application Server 133
3.2.2 Java Message Service (JMS) 136
3.2.3 Message Driven Bean (MDB) 140
3.2.4 Axis Apache (Soap Engine) 141
3.2.5 JiBX (Binding XML to Java Code) 143
3.2.6 Log4j 145
3.2.7 Ant Apache Compile 148
3.3 Libreria e-Gov 151
3.3.1 Imbustamento e Sbustamento di una busta SPCoop 154
3.3.2 Uno scenario di utilizzo 160
3.4 Registro dei Servizi 166

3.5 Porta di Dominio 172
3.5.1 Il file di configurazione delle porte delegate e applicative 180
3.5.2 Configurazione del prodotto OpenSPCoop PdP 187
3.5.3 Scenario e vantaggi di utilizzo di OpenSPCoop PdD 191
4. Il progetto OpenSPCoop 196
4.1 Motivazione di una implementazione OpenSource 196
4.2 Infrastruttura a supporto 198
4.2.1 Sito Web 199
4.2.2 Mailing List 200
4.2.3 CVS 201
4.2.4 Bugzilla 202
4.3 Risultati Ottenuti 205
4.3.1 Comunità OpenSPCoop 205
4.3.2 Versioni pubblicate del progetto OpenSPCoop 206
4.3.3 Interoperabilità con altre porte di dominio esistenti 210
4.3.4 Quesiti sulla specifica rilasciata dal CNIPA 211
5. Conclusione e Sviluppi Futuri 213
Allegato A: Schema XSD del Registro dei Servizi 216
Allegato B: Schema XSD di Porte Delegate e Applicative 220
Bibliografia 224

Introduzione

Obiettivo di questa tesi è la progettazione e l'implementazione di un insieme di componenti Open Source che aderiscono alle specifiche per la *Cooperazione Applicativa nella Pubblica Amministrazione,* rilasciate dal Centro Nazionale per l'Informatica nella Pubblica Amministrazione (CNIPA) [A51, A52].

Con il termine Cooperazione Applicativa si descrive un contesto, finalizzato all'erogazione e alla fruizione di servizi, che necessita di uno scambio di dati tra due o più Pubbliche Amministrazioni (PA). L'esigenza di una regolamentazione dell'erogazione e dell'utilizzo di servizi è maturata in questi ultimi anni come conseguenza del forte processo d'informatizzazione che ha coinvolto le Pubbliche Amministrazioni (processo comunemente riferito come e-Government). Il processo di e-Government si è sviluppato attraverso le seguenti tappe fondamentali.

2001. Nasce il Piano Nazionale di e-Government, emanato dal Ministero per l'innovazione e le Tecnologie con l'obiettivo di definire una rete di servizi usufruibili dai cittadini e dalle imprese attraverso un mezzo di trasmissione telematico.

2002. Il CNIPA istituisce un gruppo di lavoro, a cui prendono parte circa 120 esperti, in rappresentanza delle Amministrazioni centrali e locali, delle Associazioni dei fornitori e del CNIPA con lo scopo di definire:

- un'infrastruttura di comunicazione in grado di collegare tutte le amministrazioni, denominato Sistema Pubblico di Connettività (SPC) [A53];
- un insieme di standard tecnologici e di servizi infrastrutturali che permettano alle amministrazioni di interoperare attraverso interfacce applicative standardizzate, denominato Sistema Pubblico di Cooperazione (SPCoop).

Aprile 2004. Il CNIPA rilasciata la versione 1.0 della specifica della busta e-Gov [CN3], che definisce il formato standard in cui debbano avvenire le richieste di servizio e lo scambio dei dati tra l'erogatore e il fruitore di un servizio nella Pubblica Amministrazione.

Novembre 2004. Il CNIPA rilascia la versione 1.0 dell'*Architettura SPCoop* [CN1], che descrive i servizi infrastrutturali comuni e le modalità d'interazione tra i vari componenti del Sistema Pubblico di Cooperazione.

Ottobre 2005. Il CNIPA completa la stesura di un insieme di documenti che costituiscono il riferimento tecnico per lo sviluppo dei servizi infrastrutturali che delinea il quadro tecnico-implementativo del Sistema Pubblico di Cooperazione**.**
Tra i vari documenti è interessante citare:

- *Porta di Dominio.* E' descritta l'entità che dovrà gestire i servizi offerti all'interno di un dominio pubblico [CN2].
- *Registro dei Servizi e Accordo di Servizio.* Il documento intitolato 'Accordo di Servizio' [CN5] contiene la descrizione e la specifica delle varie parti che compongono un Accordo di Servizio (erogato da una PA). Si tratta di un documento standard in XML che formalizza e regola l'erogazione/fruizione di un servizio applicativo in SPCoop. E' inoltre fornita la specifica di quanto concerne la registrazione e la pubblicazione di Accordi dei Servizi all'interno di un apposito registro [CN6], dove saranno registrati anche i Soggetti abilitati ad interagire nell'architettura SPCoop.
- *Busta di e*-Gov. Descrive la nuova versione 1.1 [CN4] della specifica della busta e-Gov.

Contributo della tesi.

Il lavoro svolto in questa tesi si introduce all'interno di un più ampio lavoro sul tema della Cooperazione Applicativa avviato circa due anni fa da una collaborazione del Dipartimento di Informatica dell'Università di Pisa con la società Link.it di Pisa. In particolare, questa tesi riparte dalle conclusioni presentate nella tesi "Progettazione di un framework Open Source per la cooperazione applicativa nella Pubblica Amministrazione", discussa a luglio del 2005 da Ruggero Barsacchi.
La tesi di Barsacchi ha analizzato in dettaglio il contesto della Cooperazione Applicativa e, attraverso lo studio di vari progetti pilota tra cui il progetto CART di Regione Toscana, è giunta a proporre un'architettura innovativa per la realizzazione di un'infrastruttura compatibile con le specifiche CNIPA, che riduceva significativamente l'impatto sui sistemi preesistenti, rispetto alle soluzioni ad oggi disponibili.

L'obiettivo principale di questa tesi consiste, sinteticamente, nella trasformazione di una valida proposta architetturale in un prodotto Open Source completo, denominato OpenSPCoop, effettivamente fruibile da una comunità di utenti e di sviluppatori. Il raggiungimento di un tale obiettivo ha richiesto di affrontare i seguenti aspetti principali:

- progettazione di dettaglio dell'architettura di sistema, affrontando aspetti specifici della norma CNIPA, non ancora affrontati nella progettazione di massima sinora svolta;
- realizzazione, in linguaggio Java usando l'architettura J2EE, dei componenti software progettati;
- realizzazione di strumenti per il build e l'installazione del software a partire dal codice sorgente;
- realizzazione di test suite ed esempi d'uso del sistema;
- stesura di adeguata documentazione utente e sviluppatore;
- realizzazione di un'infrastruttura atta ad ospitare il progetto, curando aspetti come mailing list per utenti e sviluppatori, sistema pubblico di versionamento del software (cvs), sistema pubblico di bug reporting (bugzilla).

I risultati del lavoro sono oggi fruibili su http://openspcoop.org, il sito che ospita il software e supporta la comunità che si sta sviluppando attorno al progetto.
La prima fase del lavoro, circa due mesi, è stata impiegata per scrivere una specifica dettagliata dei componenti dell'architettura del software OpenSPCoop, identificando i seguenti componenti principali:

- la libreria di base org.openspcoop.egov, che implementa le funzionalità di trattamento del formato *busta e-Gov*;
- la Porta di Dominio di OpenSPCoop, che si basa sulla libreria di base org.openspcoop.egov per implementare le funzionalità di Porta di Dominio dei Servizi Applicativi dell'architettura SPCoop, fungendo quindi da intermediario tra i sistemi informativi dell'Ente e i servizi esterni con cui tali sistemi interagiscono;

- il Registro dei Servizi OpenSPCoop, un'implementazione del Registro dei Servizi SPCoop, atto a mantenere l'elenco dei soggetti erogatori di servizi e, per ognuno di questi, l'elenco dei servizi erogati e i dettagli necessari al loro utilizzo da parte di terzi;
- il Servizio di Gestione Eventi, previsto nella specifica SPCoop per il supporto del modello di cooperazione per eventi (modello EDA), che prevede lo scambio di messaggi applicativi “uno a uno” o “uno a molti”, al fine di comunicare in maniera efficiente il verificarsi di uno specifico evento.

Si è quindi passati alla fase realizzativa, riuscendo a pubblicare una prima versione del software OpenSPCoop, la versione 0.1 che includeva una prima versione usabile della libreria e-Gov, il 27 Ottobre 2005. Poco dopo, il 4 Novembre 2005, è stata rilasciata la versione 0.2 contenente la prima versione della Porta di Dominio. In seguito sono state rilasciate ulteriori 7 versioni del software fino all’attuale versione 0.5.1, rilasciata il 16 gennaio 2006, che implementa un ampio sottoinsieme delle funzionalità progettate in OpenSPCoop, ed è già in uso in alcune importanti installazioni di Pubbliche Amministrazioni Centrali.

Particolarmente importante per la rapida evoluzione delle prime versioni del software è stato il contributo dei primi utenti del progetto, cosa che ha tra l’altro confermato la bontà dell’approccio Open Source. Alcuni utenti hanno svolto una serie di importanti test di interoperabilità con i prodotti commerciali SPCoop di Oracle e Microsoft, evidenziando alcuni problemi nel software e suggerendo soluzioni poi riportate nelle versioni successive, ed anche alcune possibili ambiguità nella specifica, puntualmente segnalate anche al CNIPA.

Ad oggi il software è stato scaricato da alcune centinaia di utenti molto qualificati, tra cui grandi aziende e pubbliche amministrazioni centrali e locali, e cominciano ad arrivare i primi contributi anche allo sviluppo del software. La prima partnership ufficializzata riguarda l’ingresso del gruppo Finsiel nello sviluppo del software con un proprio gruppo di sviluppo che sarà inizialmente allocato a rafforzare gli aspetti di sicurezza del progetto.

Contenuto della tesi.

Nel **primo capitolo** è presentato il paradigma di Cooperazione Applicativa per la Pubblica Amministrazione, iniziando con un breve accenno alla storia recente riguardante l'avvento dell'informatica nelle Pubbliche Amministrazioni (sezione 1.1). Nella successiva sezione 1.2 viene descritto il contesto tecnologico su cui si basa la piattaforma di Cooperazione Applicativa. La tecnologia utilizzata nei documenti del CNIPA è quella dei Web Services, che sarà presentata nel paragrafo 1.2.1 e sarà dettagliata nelle varie componenti nei successivi paragrafi della stessa sezione: protocollo Soap (1.2.2), il linguaggio WSDL (1.2.3) e il Registro dei Servizi (1.2.4).

Nella sezione 1.3 vengono presentati i documenti di specifica rilasciati dal CNIPA. In particolare viene analizzato il ruolo della Porta di Dominio (1.3.1), la struttura della busta di e-Gov (1.3.2), le relazioni di servizio ed il Registro che le contiene (1.3.3) ed infine il paradigma di cooperazione applicativa basato su eventi (1.3.4). All'interno della sezione viene anche evidenziato come la specifica non copra le modalità di interazione tra i Sistemi Informativi interni alla Pubblica Amministrazione e la Porta di Dominio (componente di integrazione). I diversi possibili approcci per realizzare il componente di integrazione, tra cui quello utilizzato in OpenSPCoop, saranno poi analizzati nella successiva sezione 1.4.

Il **secondo capitolo** della tesi presenta i risultati della progettazione di dettaglio di OpenSPCoop, svolta nella prima fase della tesi. Prima di entrare nel dettaglio di progettazione, viene presentata nella sezione 2.1 una breve introduzione al contesto dell'architettura globale di OpenSPCoop, descrivendo tutti gli attori in gioco. Si passa quindi ad analizzare il dettaglio dei vari componenti dell'architettura, la Porta di Dominio (2.2.1), il componente di integrazione (2.2.2), il modello di cooperazione applicativa basata su eventi (2.2.3), il gestore delle funzionalità inerenti al trattamento della busta di e-Gov (modulo di controllo, sezione 2.2.4) ed infine il registro dei servizi e dei relativi accordi di servizio (2.2.5). Vengono infine illustrati, nella sezione 2.3, dei casi d'uso che consentono di comprendere meglio e validare l'architettura OpenSPCoop. Gli scenari utilizzati nei casi d'uso riguardano una invocazione di un servizio (Interazione tra due porte di dominio) ed una pubblicazione/sottoscrizione di un evento (Interazione tra porta di dominio e gestore di eventi).

Nel **terzo capitolo** viene presentata l'implementazione dei componenti di OpenSPCoop effettivamente realizzati nella tesi. L'architettura software che riguarda questi componenti viene descritta nella sezione 3.1, mentre la tecnologia utilizzata per la loro implementazione viene presentata nella successiva sezione. Dei componenti progettati sono poi stati effettivamente realizzati:

- la libreria di base org.openspcoop.egov, descritta nella sezione 3.3; questa libreria è un progetto a se stante e può essere utilizzata sia direttamente, per realizzare una propria applicazione che dialoghi utilizzando il formato busta e-Gov, sia indirettamente, utilizzando la Porta di Dominio OpenSPCoop;
- una versione xml del registro dei servizi 'RegistryService', descritto nella sezione 3.4; un'altra implementazione del registro, realizzate su tecnologia UDDI sono incluse in OpenSPCoop, ma non sono parte di questo lavoro di tesi;
- la porta di dominio, la cui implementazione è descritta nella sezione 3.5; oltre ai dettagli implementativi e alle modalità di configurazione della porta, in questa sezione viene riportato anche un esempio significativo di utilizzo dalla porta di dominio OpenSPCoop; il paragrafo 3.5.3 mostra come la porta di dominio implementata eviti la realizzazione di un applicazione 'ad-hoc' per ogni nuovo servizio e-Gov da integrare nel contesto di cooperazione applicativa; ciò è possibile grazie alla realizzazione di un componente dinamico in grado di ricevere e di gestire le richieste applicative provenienti da qualsiasi sistema richiedente, in funzione della loro descrizione nel Registro dei Servizi OpenSPCoop.

Nel **quarto capitolo** viene infine descritta l'infrastruttura di progetto realizzata per OpenSPCoop, esaminando i motivi che rendono valida la scelta di un'implementazione Open Source (sezione 4.1) e descrivendo i vari componenti dell'infrastruttura predisposta per favorire la crescita e la diffusione del software (sezione 4.2). Infine, nella sezione 4.3, vengono riportati alcuni contributi concreti ottenuti grazie alla natura Open Source del progetto, evidenziando così ulteriormente i benefici derivanti al progetto dalla disponibilità di una comunità di utenti e sviluppatori esperti della specifica SPCoop.

1. La cooperazione applicativa nella Pubblica Amministrazione

Il presente capitolo raccoglie informazioni utili a comprendere le argomentazioni presenti nella restante parte della tesi.

Nella sezione 1.1 si evidenzia l'avvento dell'informatica nelle Pubbliche Amministrazioni, che ha permesso uno scambio diretto dei servizi telematici offerti. Verrà sottolineato come, nei giorni nostri, non sia più possibile uno scambio di informazioni diretto tra pubbliche amministrazioni, e quindi la necessità di uno standard che regoli lo scambio. La sezione 1.3 descrive lo standard prodotto dal Centro Nazionale per l'Informatica nella Pubblica Amministrazione (CNIPA) che fornisce dei documenti di specifica che descrivono l'architetttura necessaria per lo scambio dei servizi tra le pubbliche amministrazioni (PA). Come verrà evidenziato in questo paragrafo, i documenti del CNIPA parlano di un complesso insieme di componenti da porre tra una PA Cliente, ed una che eroga un servizio. La tecnologia di riferimento, su cui si basa l'implementazione della piattaforma di cooperazione applicativa specificata nei documenti, è quella dei Web Services. Una descrizione delle tecnologie che stanno alla base dei Web Services viene affrontata nella sezione 1.2.

Un aspetto solo accennato superficialmente dai documenti del CNIPA riguarda la descrizione di un eventuale protocollo da utilizzare tra i Sistemi Informativi Locali (SIL) di una PA e il componente Porta di Dominio dell'architettura SPCoop. Questo aspetto sarà illustrato nella sezione 1.4, dove verranno evidenziati alcuni possibili scenari. Si arriverà alla definizione di un proxy dinamico, che è frutto del lavoro di tesi svolta da un mio collega, Ruggiero Barsacchi, con cui sono stati evidenziati gli enormi vantaggi che comportano un suo utilizzo.

Laddove possibile, si è cercato di non entrare in dettagli tecnici, ma di privilegiare piuttosto una visione d'insieme.

1.1 Storia recente dell'informatica nella Pubblica Amministrazione

All'inizio degli anni '90 risale l'introduzione dell'informatica nelle pubbliche amministrazioni. Vi è quindi una trasformazione delle norme e dei processi applicativi, realizzati precedentemente attraverso un intervento umano, in logica applicativa da poter utilizzare con i computer. I vantaggi sono notevoli:

- Minore potere decisionale in mano all'uomo, fatto che comporta miglioramento dell'efficienza e eliminazione di favoritismi e corruzioni.
- Maggiore privacy per i cittadini
- Automatizzazione dei processi. La traduzione di un processo in logica applicativa comporta la riduzione del ciclo di vita di una pratica.
- Separazione delle funzioni di front-end da quelle di back-end. Le PA dispongono solitamente di duplici compiti, l'interazione con il cittadino (front-end) e la gestione dei processi amministrativi veri e propri (back-end) innescati dal cittadino stesso o da altre PA.
- Unificazione dei vari front-end offerti da una PA in un unico Sportello con cui il cittadino deve interagire. Attraverso questo sportello unico, le richieste effettuate dal cittadino possono essere reindirizzate al back-end corretto, ed inoltre è possibile realizzare servizi complessi componendo front-end diversi offerti magari da diverse PA.

La Figura 1-1 illustra un esempio di scenario di cooperazione applicativa. In questo contesto esemplificativo esistono tre pubbliche amministrazioni che contengono ognuna al loro interno un sistema informativo locale. La PA Centrale del dominio A ha bisogno sia di dati conservati dall'Amministrazione Locale, sia di dati conservati nella ASL e quindi effettuerà due richieste di servizio verso i sistemi informativi presenti nei rispettivi domini (frecce con etichetta 'a').
Dalla figura si può notare come il sistema informativo dell'ASL risolve la richiesta di servizio ricevuta senza dover interagire con altri domini, mentre il sistema del dominio B deve interagire anche con il dominio C (freccia con etichetta 'b').

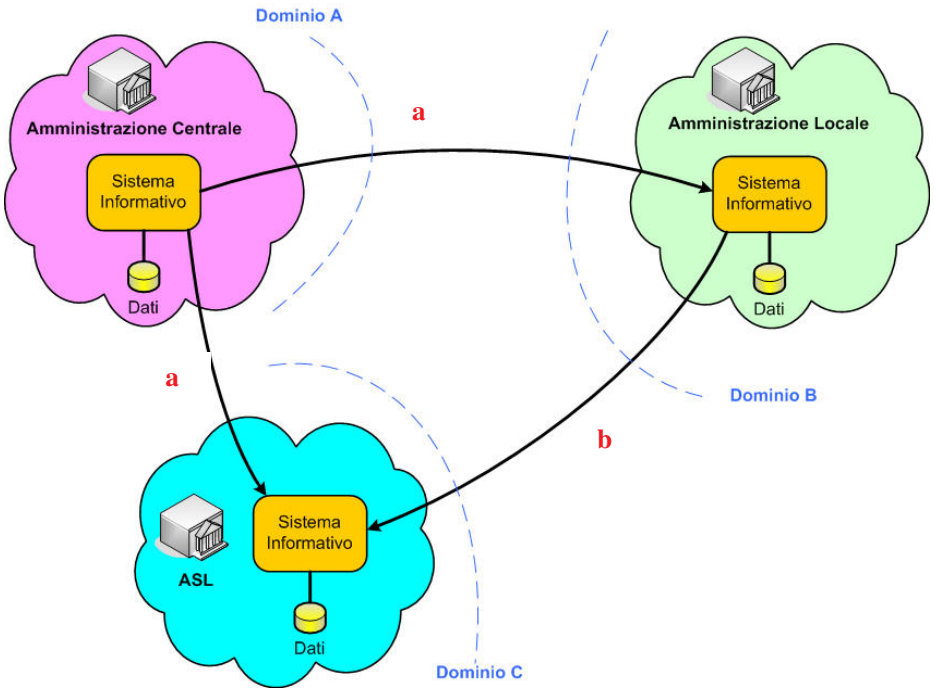


Figura 1-1, Scenario della cooperazione applicativa tra PA prima dell'avvento di SPCoop

Negli anni successivi all'avvento dell'informatica nelle PA, i compiti di front-end, effettuati dagli enti locali nei confronti dei cittadini e anche delle imprese, sono cresciuti sia in numero che in qualità. Questo poiché si è cercato di arrivare ad uno stato in cui un cittadino o una impresa che necessiti di un qualsiasi servizio possa rivolgersi allo sportello unico del proprio Comune per accedere a tutti i servizi offerti dalla Pubblica Amministrazione. In questo scenario, ad esempio, la PA centrale deve svolgere un ruolo di back-end nei confronti delle PA locali dove i cittadini si recano per effettuare le proprie richieste. In questa ottica sono sorti diversi problemi:

- Necessità di dare valore legale alle comunicazioni
- Esigenza di coordinazione di processi realizzati con il concorso di trattamenti distribuiti tra sistemi informatici di cui sono responsabili soggetti pubblici e privati. Il coordinamento e la collaborazione di tali sistemi devono essere corredati dalla capacità di avere un riscontro in ogni momento dello stato di avanzamento dei processi applicativi. Inoltre, oltre alle informazioni relative al processo in corso deve essere possibile ottenere varie informazioni, tra cui il soggetto che ne ha causato l'instanziazione, l'ente che ha raccolto la richiesta, ed altre informazioni di contesto.

- Esigenza di unificazione dei protocolli utilizzati per la comunicazione.
- Differenziazione di modalità di servizio. Ad esempio la produzione di una risposta in seguito all'arrivo di una richiesta di servizio può comportare l'attesa di ore o giorni. In questo contesto uno scenario applicativo richiesta/risposta dove il richiedente rimane in attesa non è praticabile.

In questo contesto è nata l'esigenza di approfondire e definire il paradigma di cooperazione applicativa, che si basa sullo scambio di dati, finalizzato all'erogazione e alla fruizione di servizi, tra sistemi informatici eterogenei. Attraverso uno studio effettuato dal CNIPA, è stata prevista la creazione di:

- Un'infrastruttura di comunicazione in grado di collegare tutte le amministrazioni, definita Sistema Pubblico di Connettività (SPC);
- Un insieme di standard tecnologici e di servizi infrastrutturali che consentano alle amministrazioni di interoperare attraverso delle interfacce standard. Tale prodotto è stato definito Sistema Pubblico di Cooperazione (SPCoop).

Nelle sezioni successive, verrà illustrato il Sistema Pubblico di Cooperazione. Questo comporta la definizione di tre componenti principali:

- un registro dei servizi, dove vengono registrati i servizi offerti dalle pubbliche amministrazioni;
- una porta di dominio, che viene utilizzata da una PA come porta a cui inviare le richieste di servizio. Sarà poi la porta di dominio a redirigere la richiesta verso la corretta destinazione (PA che eroga il servizio);
- un formato di scambio, denominato 'busta e-Gov' in cui viene inserita, da parte della porta di dominio, la richiesta ricevuta da una PA locale.

Inoltre vedremo come in questo sistema è possibile effettuare oltre ad un invocazione di servizio punto-punto anche una comunicazione basata su eventi, mediata da un gestore degli eventi. La Figura 1-2 illustra l'inserimento dell'architettura SPCoop all'interno dello scenario visto nella Figura 1-1 precedente.

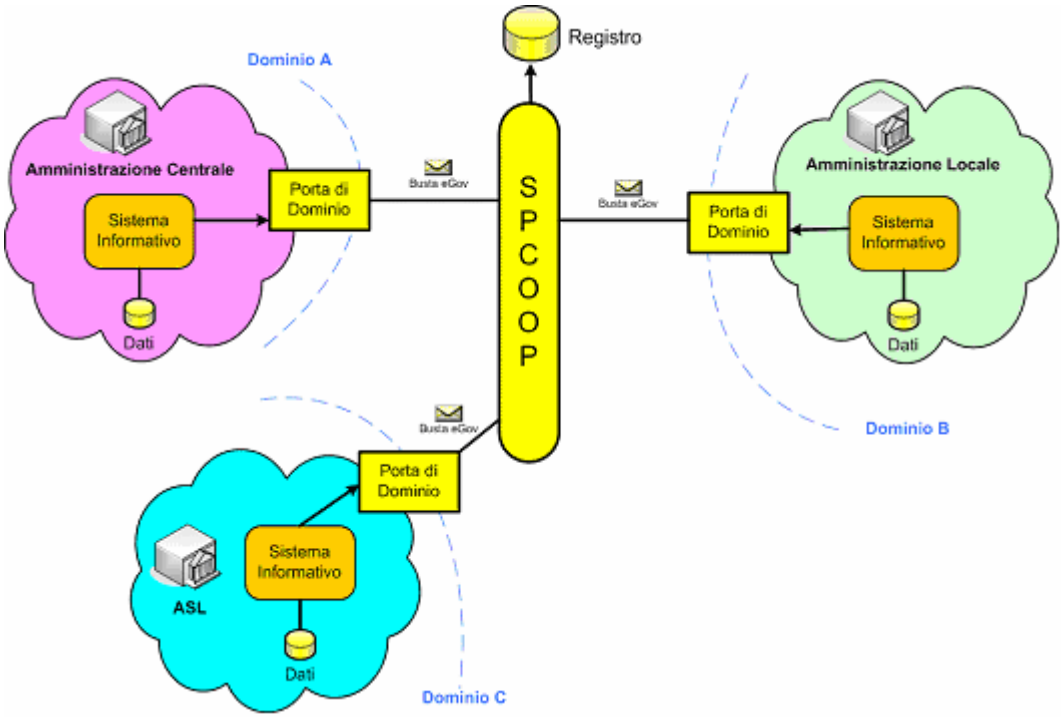


Figura 1-2, Scenario della cooperazione applicativa con l'avvento dell'architettura SPCoop

1.2 Contesto Tecnologico

Riuscire a comprendere il complesso funzionamento di una architettura di cooperazione applicativa e capirne le motivazioni che hanno condotto gli autori dei documenti di specifica CNIPA (che saranno illustrati nella sezione 1.3) alla loro stesura richiede un bagaglio di nozioni riguardanti la tecnologia dei Web Services [A38, A41, A42]. Un conciso riassunto dei concetti principali di questa tecnologia viene presentata in questa sezione.

1.2.1 Introduzione al mondo Web Services

Prima dell'avvento dei Web Services, le aziende producevano applicazioni web denominate B2C (Business to Consumer) che avevano come fruitore del servizio l'utente che, da casa o dall'ufficio, accede, attraverso una connessione telefonica o attraverso una rete aziendale, a servizi di vario genere. Successivamente è nata invece la necessità, da parte delle diverse aziende, di comunicare tra loro, di effettuare quello che si chiama B2B (Business to Business). Il rivenditore deve quindi comunicare con il fornitore per ordinare i prodotti del proprio catalogo. Nel caso di più fornitori, il rivenditore potrebbe avere la necessità di conoscere quale di questi sia per lui più conveniente. In questo contesto è necessaria una semplice comunicazione tra i due sistemi informativi, i quali devono però supportare un 'linguaggio di comunicazione' comune, che permetta di interoperare. Si presenta cioè un problema di interoperabilità fra sistemi informativi che possono utilizzare tecnologie molto differenti tra loro. Deve essere utilizzato un 'linguaggio di comunicazione' standard.

XML. Da qui all'utilizzo di documenti XML il passo è breve. Attraverso un documento XML è possibile descrivere un insieme di informazioni strutturate che possono essere trasmesse attraverso un protocollo di trasporto come può essere l'HTTP. Un documento XML ha infatti la fondamentale caratteristica di essere del "semplice testo" le cui informazioni possono essere estratte con gli strumenti ormai disponibili per tutti i linguaggi. Utilizzare l'HTTP e l'XML per la comunicazione è però ancora troppo generico.

Nel nostro scenario è necessario che il rivenditore ed il produttore parlino la stessa lingua e quindi che le informazioni contenute all'interno del documento XML inviato dal produttore siano comprese nel modo corretto da parte del rivenditore (e da qualunque rivenditore). Attorno quindi alla realizzazione di documenti XML serve un meccanismo che permetta alle diverse parti di concordare il formato da utilizzare per la rappresentazione delle informazioni e quindi permetta di descrivere alcune regole cui lo stesso documento dovrà sottostare per essere considerato valido. A tale scopo sono state realizzate diverse tecnologie. Quella denominata Data Type Definition (DTD) [A59] è stata probabilmente la prima tecnologia utilizzata a tale scopo. Un'altra tecnologia, molto diffusa attualmente, è l'XML Schema Definition Language (XSD) [A15, A17-A20]. Esempi di file XSD sono inseriti nell'appendice A e B della tesi. Lo schema XSD permette di definire elementi e attributi che possono o devono apparire in un documento, l'ordine ed il numero di elementi, se un elemento è vuoto o può contenere del testo, il tipo ed i valori di default o fissi di un elemento o di un attributo, ecc… Quando due sistemi devono comunicare tra loro devono quindi accordarsi attraverso uno schema XSD che definisce la struttura del documento XML che si scambieranno. Un documento XML che non soddisfa le regole dello schema dovrà essere scartato.

SOA. Supponiamo ora che il fornitore abbia la necessità di rendere disponibili le informazioni relative ai propri prodotti non solo al nostro rivenditore ma anche ad altri. Viceversa supponiamo che il rivenditore non voglia comprare tutto dallo stesso fornitore ma abbia la necessità di confrontare i prezzi con quelli di altri fornitori magari più convenienti. Possiamo quindi pensare al fornitore come colui che mette a disposizione un servizio per la consultazione dei propri prodotti. Possiamo pensare al rivenditore come un utente che utilizza i servizi esposti dai diversi fornitori per determinare quello più conveniente. Se pensiamo ad uno scenario di questo tipo si intuisce la necessità di alcuni servizi di supporto che permettano al rivenditore di:

- individuare i servizi messi a disposizione dai diversi fornitori per la consultazione delle informazioni di catalogo
- concordare con ciascun fornitore da consultare la modalità di accesso al servizio

- invocare il servizio passando dei parametri in input ottenendo delle informazioni in output.

Analogamente al fornitore serviranno degli strumenti che permettano di:

- descrivere il proprio servizio
- descrivere le modalità di accesso allo stesso
- informare i diversi rivenditori della disponibilità del proprio servizio

Per soddisfare i requisiti citati è possibile utilizzare una architettura di riferimento, la ***Service Oriented Architecture*** **(SOA)**, che si inserisce perfettamente all'interno delle considerazioni relative al mondo *business-to-business* (B2B) appena effettuate. Grazie a SOA è anche possibile capire facilmente gli scopi delle varie tecnologie che caratterizzano i Web Service.

Si può immaginare un servizio come una interazione tra due attori in gioco, quali ad esempio il fornitore e il richiedente. Questo è il paradigma che si trova nell'interazione di tipo *client-server*. Attraverso la SOA questa interazione viene arricchita con un ulteriore attore detto *Service Directory o Service Broker* che, come illustrato nella Figura 1-3, si inserisce all'interno della comunicazione tra fornitore e fruitore del servizio.

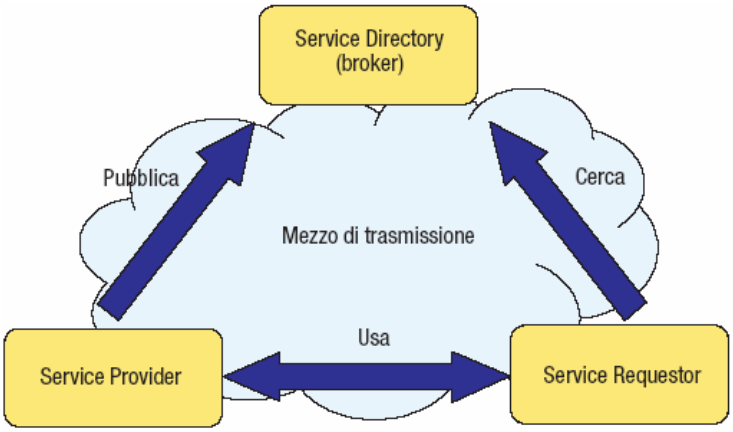


Figura 1-3, Service Oriented Architecture (SOA)

Il ruolo di ognuno dei tre attori, coinvolti nella SOA, è il seguente:

- ***Service Provider.*** E' l'attore che realizza e mette a disposizione un servizio. Tramite l'operazione di *publish* il servizio viene "pubblicato", in quanto le caratteristiche del servizio realizzato vengono memorizzate all'interno di un *registry* accessibile pubblicamente. Il Service Provider rimane, quindi, in attesa che un utente richieda tale servizio.
- ***Service Directory o Service Broker.*** Questo componente si occupa della gestione del registry, permettendo, a chi ha necessità, di ricercare un servizio sulla base delle caratteristiche con le quali è stato definito e memorizzato. Naturalmente, il Service Directory può seguire politiche di controllo degli accessi sulle interrogazioni in modo da limitare la visibilità sui servizi inseriti. Nel presente lavoro il registry, salvo diversa indicazione, viene considerato totalmente accessibile.
- ***Service Requestor.*** Rappresenta un potenziale utente che richiede un servizio. A tale scopo, tramite una primitiva di *find* l'utente interagisce con il Service Directory per ottenere il servizio più adatto ai propri obiettivi. Una volta individuato si collega al Service Provider corrispondente (*bind*) e inizia a fruire del particolare servizio (*use*).

L'approccio adottato da SOA ha il vantaggio di potersi applicare a qualunque mezzo di trasmissione come la telefonia mobile, il web, o paradossalmente anche la posta ordinaria, permettendo in tal modo di realizzare applicazioni multi-canale, fruibili cioè attraverso diversi dispositivi. Partendo da questa considerazione si può affermare che una architettura per Web Service è un'istanza di una SOA dove il mezzo di comunicazione considerato è il Web. Secondo la definizione data dal W3C un **Web service** (servizio web) è un sistema software progettato per supportare l'interoperabilità tra diversi elaboratori su di una medesima rete; caratteristica fondamentale di un Web Service è quella di offrire un'interfaccia software (descritta in un formato automaticamente elaborabile quale, ad esempio, il WSDL [W3, A11, A12]) che permette ad altri sistemi di interagire con il Web Service stesso attivando le operazioni descritte nell'interfaccia tramite appositi "messaggi" inclusi in una "busta" SOAP [W1, A1-A9]: tali messaggi sono, solitamente, trasportati tramite il protocollo HTTP e formattati secondo lo standard XML.

Nella Figura 1-4 sono evidenziate le tecnologie adottate nel mondo WebService, per quanto riguarda i tre attori coinvolti nel contesto SOA.

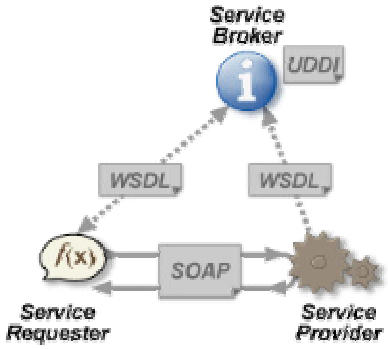


Figura 1-4, Istanza di SOA: Web Service

1.2.2 SOAP

SOAP (acronimo di **Simple Object Access Protocol**) [W1, A1-A9] è un protocollo leggero per lo scambio di messaggi tra componenti software. SOAP è una struttura operativa estensibile e decentralizzata che può operare sopra vari stack di protocolli per reti di computer, in teoria con qualunque protocollo di Internet esistente. Attualmente HTTP è il protocollo più comunemente utilizzato e l'unico ad essere stato standardizzato dal W3C. SOAP si basa sul metalinguaggio XML e la sua struttura segue la configurazione Head-Body, analogamente ad HTML.

SOAP gestisce due tipi di messaggi: Call e Response. Il messaggio di Call permette di invocare un servizio remoto (da parte del client), mentre un messaggio di Response include il risultato proveniente dal servizio invocato (fornito dal server). Come definito nelle sue specifiche, questo protocollo è costituito da tre parti:

- **SOAP envelope.** Definisce una struttura per descrivere cosa il messaggio contiene e come deve essere processato.
- **SOAP Encoding Rule.** Definisce un meccanismo di serializzazione che può essere usato per scambiare istanze di tipi di dati definiti dalle applicazioni. Il suo obiettivo è di codificare i dati presenti nella SOAP envelope attraverso schemi XML (es. http://schemas.xmlsoap.org/soap/encoding)

- **SOAP RPC representation.** Definisce una convenzione (insieme di regole) utilizzata per rappresentare le chiamate e le risposte alle procedure remote (RPC)

Una raffigurazione di quanto detto è rappresentato dalla Figura 1-5:

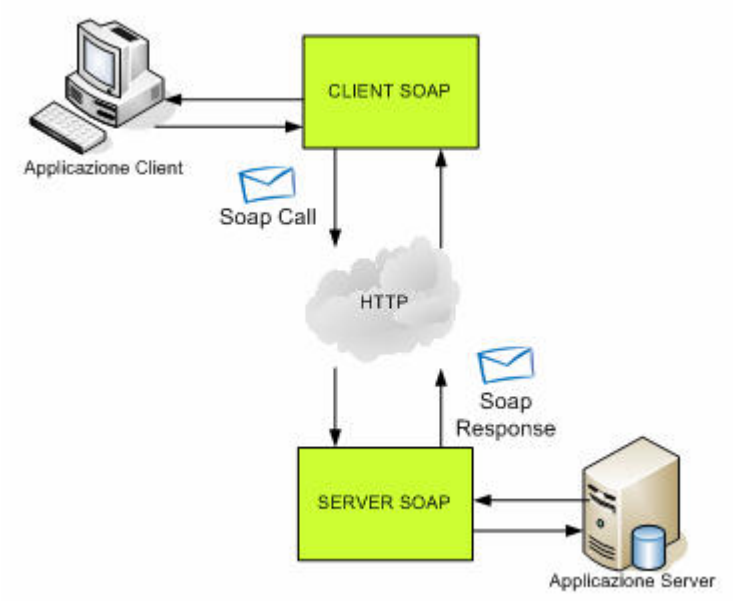


Figura 1-5, Scenario d'esempio di una SOAP Call e Response

La struttura di un messaggio SOAP, definito da un SOAP Envelope, è costituita da due parti:

- **Header.** Rappresenta la sezione dove è possibile specificare dei parametri di comunicazione ed informazione aggiuntiva sul messaggio trasmesso. Questa sezione non è obbligatoria, ma se presente deve sempre essere posta prima del Body. Deve rispettare le regole di encoding standard specificate in 'http://schemas.xmlsoap.org/soap/encoding', o deve essere esplicitamente definito un encoding diverso. Un header può essere composto da più elementi, i quali possono contenere degli attributi quali 'mustUnderstand', che forza un server soap all'analisi dell'elemento, e 'actor' che fornisce un identificatore all'elemento. L'header SOAP serve quindi per contenere informazioni sullo stato del sistema, sullo stato del client, sulle impostazioni relative alla sessione corrente dell'oggetto e tutto ciò che può essere utile alla personalizzazione della comunicazione.

- **Body.** Rappresenta la parte del messaggio SOAP obbligatoria che contiene i dati da fornire all'applicazione server. Deve rispettare anch'essa regole di encoding standard.

Un messaggio SOAP ha quindi il seguente scheletro:

```
 <soapenv:Envelope
      soapenv:encodingStyle=http://schemas.xmlsoap.org/soap/encoding
      xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
<soapenv:Header>
........
</soapenv:Header>
<soapenv:Body>
........
</soapenv:Body>
</soapenv:Envelope>
```

Durante la comunicazione o l'elaborazione di messaggi SOAP, esiste la possibilità che ci siano dei problemi, con la conseguente generazione di errori classificabili in errori di trasporto o errori SOAP. Nel primo caso la gestione degli errori è a carico del protocollo di trasporto (ad es. un messaggio http con codice d'errore 404 corrisponde a 'file non trovato'). Nel secondo caso la gestione degli errori è a carico dei componenti SOAP, i quali segnalano le eccezioni attraverso l'utilizzo dell'elemento speciale Fault, inserito all'interno del SoapBody.
Le regole che definiscono un elemento Fault sono:

- Se presente, l'elemento Fault deve comparire all'interno dell'elemento Body;
- L'elemento Fault può comparire al massimo una volta all'interno dell'elemento Body;
- L'elemento Fault può contenere i seguenti sottoelementi:
    - Fault code. Specifica il codice (standard) d'errore, è obbligatorio.
    - Fault string. Descrizione testuale dell'errore, è obbligatorio.
    - Fault actor. Specifica l'attore che ha fatto generare l'errore, è opzionale.
    - Details. Specificano ulteriori dettagli. Sono opzionali.

Un altro speciale messaggio, è il SOAP Message con Attachments [W2, A6, A10]. Si tratta di un messaggio SOAP trasmesso insieme con degli attachments di varia natura, come ad es. immagini, documenti ecc…

In questo caso il messaggio è formato usando il tipo HTTP standard 'Multipart/Related' utilizzato anche per altri contesti dove sono necessari attachments, quali ad es. l'e-Mail. Le regole per la definizione di un messaggio SOAP con attachments sono le seguenti:

- Il messaggio SOAP, deve essere inserito nella parte radice della struttura Mulipart/Related ed il tipo associato alla parte contenente il SOAPEnvelope è 'text/xml'.
- Le altre parti della struttura potranno contenere attachments, con l'unico vincolo che esse devono contenere un'introduzione che ne specifica un identificatore (Content-ID), un tipo (Content-Type) ed una locazione (Content-Location), in accorda alla specifica dell'header MIME definito nel RFC 2557.

Un esempio di messaggio SOAP con Attachments è il seguente (è incluso anche l'header HTTP):

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary;
type=text/xml;
        start="<claim061400a.xml@claiming-it.com>"

--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: <claim061400a.xml@claiming-it.com>

<?xml version='1.0' ?>
<SOAP-ENV:Envelope
........
</SOAP-ENV:Envelope>

--MIME_boundary
Content-Type: image/tiff
Content-Transfer-Encoding: binary
Content-ID: <claim061400a.tiff@claiming-it.com>

...binary TIFF image...
--MIME_boundary—
```

1.2.3 WSDL

WSDL (acronimo di **Web Services Description Language**) [W3, A11, A12] è un linguaggio formale in formato XML utilizzato per la creazione di interfacce che descrivono la modalità di utilizzo di un servizio fornito come Web Service. Un "documento" WSDL contiene infatti, relativamente al Web Service descritto, informazioni su:

- *cosa* può essere utilizzato (le "operazioni" messe a disposizione dal servizio);
- *come* utilizzarlo (il protocollo di comunicazione da utilizzare per accedere al servizio, il formato dei messaggi accettati in input e restituiti in output dal servizio ed i dati correlati);
- *dove* risiede il servente che eroga un servizio (*endpoint* del servizio che solitamente corrisponde all'indirizzo - in formato URI - che rende disponibile il Web Service)

Le operazioni supportate dal Web Service ed i messaggi che è possibile scambiare con lo stesso sono descritti prima in maniera astratta e quindi collegati ad uno specifico protocollo di rete e ad uno specifico formato. Uno scheletro d'esempio di un documento WSDL è il seguente:

```
<wsdl:definitions name="NomeDelServizio"
                  targetNamespace="URL"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
      <!--  Definizioni astratte  -->
      <wsdl:types>....</wsdl:types>
      <wsdl:message>....</wsdl:message>
      <wsdl:portType>....</wsdl:portType>
      <!-- Definizioni concrete  -->
      <wsdl:binding>....</wsdl:binding>
      <wsdl:service>....</wsdl:service>
</wsdl:definitions>
```

Di seguito è riportato l'elenco di tutte le sezioni assieme alla loro funzione:

- *tipi (types)* definisce i tipi utilizzati all'interno dei messaggi del servizio, attraverso la definizione di un tipo di encoding da utilizzare, specificato da un XML Schema;
- le sezioni *messaggio (message)* contengono le definizioni dei messaggi di scambio del servizio e fanno uso di parti definite come tipi nelle sezioni *tipi*;

- la sezione *tipi di porta (portType)* definisce le operazioni esposte dall'erogatore del servizio; concettualmente si tratta dei metodi dell'oggetto remoto che costituiscono i servizi web; per ogni metodo viene definito il messaggio di input ed il messaggio di output;
- la sezione *associazioni (binding)* contiene il collegamento tra i tipi di porta (cioè la definizione astratta del servizio) e l'endpoint fisico; qui sono contenute le informazioni che indicano il protocollo da utilizzare e come ricondurre i messaggi di input ed output al protocollo utilizzato;
- la sezione *servizi (service)* contiene la definizione del servizio, includendo la sua descrizione e l'end-point da dove è possibile usufruire del servizio (es. una URL).

1.2.4 UDDI e LDAP

UDDI (Universal Description Discovery and Integration) [A13, A14] affronta le problematiche di pubblicazione e reperimento di servizi, proponendo una architettura che permette di affrontare le tre problematiche riassunte nell'acronimo UDDI:

- Accesso alla descrizione di servizi, di tipologie di servizi e di fornitori di servizi secondo una struttura dati ben definita;
- Astrazione dalla tecnologia utilizzata nella realizzazione del servizio. Questo al fine di permettere l'integrazione tra servizi realizzati in modo tecnologicamente differente;
- Ricerca di un servizio secondo differenti chiavi di ricerca.

UDDI, quindi, non è un contenitore di servizi o di descrizioni di servizi, bensì uno strumento che, utilizzando opportune strutture dati, tiene traccia della dislocazione dei servizi e delle loro descrizioni.

L'architettura UDDI prevede la creazione di un ambiente distribuito *peer-to-peer* in cui i vari *nodi*, che contengono una parte dei servizi disponibili, possano interoperare tra di loro allo scopo di soddisfare le richieste di pubblicazione e ricerca di servizi. Per garantire l'interoperabilità, ogni nodo dell'infrastruttura, gestito da una figura che assume il ruolo di *operatore*, deve essere realizzato secondo le specifiche rilasciate dal gruppo di lavoro UDDI.

Tali specifiche definiscono sia la struttura delle informazioni che un registro UDDI deve memorizzare, sia il set minimo di API da implementare per l'accesso alle informazioni stesse.

All'interno di un registro UDDI abbiamo le seguenti tipologie di informazioni:

- **businessEntity** – fornisce informazioni relative a un'azienda che espone servizi;
- **businessService** – fornisce informazioni non tecniche relative a un servizio offerto da un'azienda;
- **bindingTemplate** – fornisce informazioni tecniche relative a un servizio;
- **tModel** - fornisce informazioni tecniche aggiuntive relative all'interfaccia del servizio;
- **tassonomie** – rappresentano schemi di classificazione delle informazioni attraverso cui le entità vengono identificate.

In un registro UDDI, una businessEntity contiene riferimenti a una o più businessService. Un businessService referenzia una o più bindingTemplate che puntano a loro volta ad un indirizzo con cui il servizio può essere invocato. Le varie BusinessEntity sono memorizzate nel **Business Registry**. Una tModel invece contiene l'URL da cui e' possibile scaricare il file WSDL, interfaccia standard dei web Services. Le varie tModel sono memorizzate nel **Service Type Registry**.

Uno schema che descrive le relazioni tra le tipologie di informazioni esistenti in un registro UDDI è presentato nella Figura 1-6.

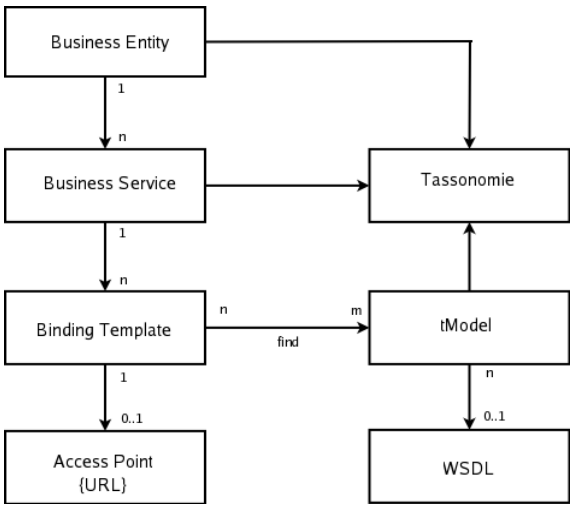


Figura 1-6, Relazioni tra tipologie di informazioni presenti in un registro UDDI

La Figura 1-7 illustra i concetti appena descritti, evidenziando la struttura di un nodo UDDI.

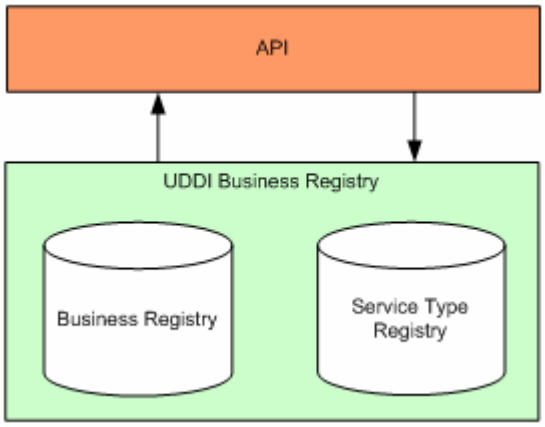


Figura 1-7, Struttura di un nodo UDDI

UDDI, si appoggia internamente ad un Database su cui memorizzare i dati. Una possibile combinazione è quindi UDDI + Oracle, UDDI + MySQL ecc… Una interessante combinazione, che sarà utilizzata per la progettazione del registro dei servizi di OpenSPCoop (vedi paragrafo 2.2.5) è l'utilizzo di un registro UDDI che si appoggia ad un server LDAP.

LDAP.

Il protocollo Lightweight Directory Access Protocol (LDAP) [A54, A55] è uno standard aperto per l'erogazione di servizi di directory tramite una rete Intranet o Internet. La gestione delle informazioni in LDAP è basata sul concetto di entry. Un'entry è una raccolta di attributi che fanno riferimento ad un Distinguished Name (DN). Il DN è unico ed è usato per riferirsi inequivocabilmente all'entry. Ogni attributo ha un tipo ed uno o più valori. I tipi sono tipicamente sequenze mnemoniche, come "cn" per nome comune o "mail" per un indirizzo e-mail. La sintassi dei valori dipende dal tipo di attributo. Per esempio, un attributo tipo "cn" dovrebbe contenere il nome di un utente. Il funzionamento di LDAP si basa sul paradigma client-server e su una gestione gerarchica degli oggetti presenti nella directory. Un grosso vantaggio di LDAP consiste nel fatto che sia possibile ottenere una distribuzione uniforme dei dati, anche se posti su distinti database, mantenendo tempi di risposta ridotti nella lettura del contenuto degli oggetti e quindi avendo la possibilità di replicare le informazioni su server distinti per un bilanciamento del carico di lavoro nella rete. L'unico inconveniente può essere dato dal verificarsi di inconsistenze temporali nei database replicati dei server LDAP dovute ad aggiornamenti non sincronizzati dei dati.

In LDAP vi è il concetto di **Directory**. Una directory è un database ottimizzato per la ricerca e la lettura di informazioni riguardanti oggetti presenti all'interno di una rete. Le informazioni sono basate su attributi. Un servizio di directory deve permettere sofisticate funzioni di ricerca su tali informazioni e fornire tempi di risposta molto brevi. I servizi di directory possono essere di 2 tipi: locali e distribuiti. I servizi di directory locali permettono di accedere ad informazioni in un contesto molto ristretto (es. servizio finger su un singolo host). I servizi di directory distribuiti permettono la partizione e la replica dei dati su differenti macchine per bilanciare il carico di lavoro (es. DNS).

1.3 Il Sistema Pubblico di Connettività e Cooperazione

Il Sistema Pubblico di Connettività (SPC) e di Cooperazione (SPCoop) è una architettura progettata dal CNIPA con lo scopo di regolare, attraverso delle specifiche, l'erogazione e la fruizione di servizi delle pubbliche amministrazioni italiane. Costituisce l'infrastruttura che permette la cooperazione applicativa, che consente uno scambio di servizi tra applicazioni informatiche appartenenti sia a soggetti privati che a enti amministrativi. La definizione di questo standard di cooperazione include la specifica di una rete sicura di trasporto tra le amministrazioni (Connettività) e di un livello di comunicazione a messaggi (application layer) che permetta protocolli di comunicazione a livello applicativo (Cooperazione).

Il modello di riferimento dell'architettura del Sistema pubblico di cooperazione si basa sulla tecnologia dei Web Services. Ogni PA partecipante alla cooperazione espone le proprie funzionalità svolte sotto forma di una lista di servizi erogati, dotandoli di modalità di interfacciamento standardizzate. Quindi una qualunque PA può utilizzare servizi erogati da un'altra PA. La fruizione di un servizio è regolata da un accordo di servizio effettuato tra l'ente fruitore e quello erogatore ed accessibile attraverso un registro dei servizi. In questo accordo sono incluse diverse informazioni quali interfacce di scambio dei messaggi per l'interrogazione (interfaccia standard dei web services: WSDL), requisiti di sicurezza, proprietà sul servizio richiesto, ecc...

I vantaggi principali che derivano dall'adozione di un tale modello sono:

- L'architettura dei servizi basati su tecnologie Web Services è pienamente distribuito e permette l'interoperabilità e la cooperazione dei sistemi informatici sulla base di accordi che prevedono scambio di funzionalità regolate da interfacce di interrogazione.
- L'indipendenza delle scelte implementative, delle funzionalità applicative e delle funzionalità di infrastruttura dei sistemi cooperanti rende tale modello perfettamente adatto alle esigenze di autonomia di gestione delle amministrazioni centrali e locali e all'interoperabilità con sistemi applicativi di imprese private.

- L'utilizzo della tecnologia Web Services permette di minimizzare l'impatto sull'implementazione delle funzionalità applicative già realizzate nei sistemi informativi esistenti. Quindi l'adozione di questa tecnologia permette di avere un costo economico contenuto.
- La componente di interfacciamento del servizio deve essere specificata nel formato WSDL come richiesto dall'architettura SOA. Questo comporta il vantaggio che un'applicazione client possa adattarsi dinamicamente a variazioni dell'interfaccia del servizio.

Una caratteristica fondamentale del modello dell'architettura di servizi è che l'interazione tra i sistemi partecipanti avviene esclusivamente per mezzo dello scambio di messaggi in rete. Per soddisfare questo requisito è necessario che due sistemi che devono interagire siano connessi per mezzo di uno o più canali che trasportano i messaggi scambiati. Il Sistema Pubblico di Cooperazione utilizza per il trasporto dei messaggi esclusivamente i canali e i meccanismi di trasporto forniti dal Sistema Pubblico di Connettività. Quindi il SPC fornisce l'infrastruttura di base per il trasporto dei messaggi ed un insieme di funzionalità infrastrutturali di sicurezza e qualità di servizio applicabili al trasporto dei messaggi.
Nei paragrafi di questa sezione saranno illustrati i vari componenti che compongono il Sistema Pubblico di cooperazione, senza descrivere ulteriormente il Sistema Pubblico di Connettività. Per quanto riguarda il SPC è sufficiente sapere che è semplicemente ed esclusivamente il fornitore dell'infrastruttura per il trasporto dei messaggi, e che si baserà sul protocollo di trasporto HTTP e HTTPS, il protocollo utilizzato più comunemente nei Web Services, per lo scambio di messaggi SOAP. Questo poiché la specifica di SOAP cita un suo possibile utilizzo su qualsiasi protocollo di trasporto esistente, ma nella pratica descrive dettagliatamente l'associazione SOAP con HTTP. Per questo motivo attualmente la maggior parte dei Web Services sono usufruibili con un protocollo di trasporto HTTP.

1.3.1 La Porta di Dominio

Nell'ambito delle Pubbliche Amministrazioni, un dominio comprende le responsabilità di un ente pubblico o di un soggetto privato e raccoglie al suo interno tutti i servizi da esso gestiti. Una porta di dominio può appartenere anche a più di un soggetto, ognuno dei quali è comunemente chiamato Sistema Informativo Locale (SIL). Le comunicazioni da e verso un dominio devono attraversare la Porta di Dominio (PdD) che costituisce quindi l'unico punto di accesso tra i domini. Sulla propria PdD ogni SIL veicola tutte le richieste di servizio verso gli altri domini e riceve tutte le richieste provenienti dagli altri domini. La PdD si configura in pratica come un proxy applicativo. La Figura 1-8 illustra lo scenario appena descritto:

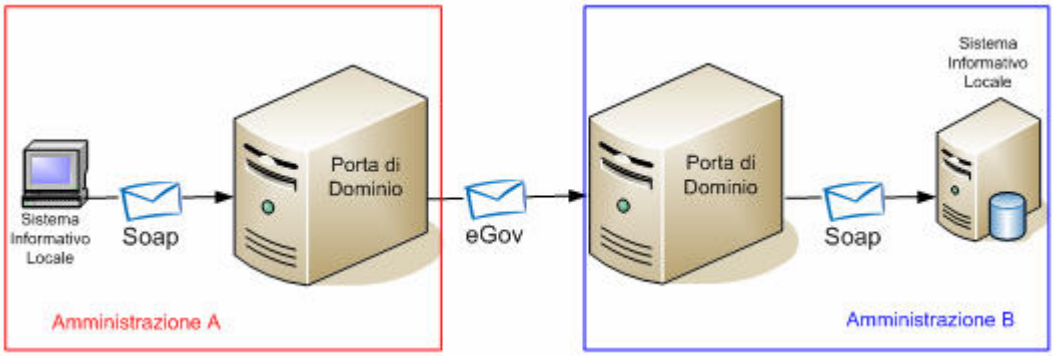


Figura 1-8, Scenario di utilizzo delle Porte di Dominio

Una porta di dominio è l'insieme dei componenti distribuiti che implementano le funzionalità infrastrutturali che permettono la messa in opera dello scambio di messaggi e dei requisiti di sicurezza a diversi livelli di comunicazione, in modo indipendente dalla logica applicativa. I compiti di una PdD sono i seguenti:

- Scambio a livello connessione (HTTP). La PdD dovrà inglobare:
  - un client HTTP, che si incarica di aprire una connessione, di formare la richiesta HTTP e di trasmetterla sulla connessione, e di mettersi in attesa di una risposta con un eventuale carico applicativo da smistare al Sistema Informativo Locale.
  - un server HTTP, che si occuperà di ricevere una richiesta, e mantenendo la connessione HTTP aperta, inoltrerà la richiesta all'appropriato SIL, aspetterà una risposta applicativa ed inserirà essa nel carico applicativo della risposta HTTP.

- Sicurezza a livello connessione (SSL, TLS). Verrà effettuata una autenticazione, un controllo di integrità del carico (firma digitale e sua validazione) ecc…
- Gestione dei messaggi. Vedremo nel paragrafo 1.3.2 il tipo di messaggio scambiato tra porte di dominio. Il messaggio è un SoapEnvelope che include nell'Header informazioni per la gestione e per il routing della richiesta/risposta di un servizio verso una porta di dominio. Questo tipo di messaggio viene chiamata 'busta SPCoop' o 'busta e-Gov'.
- Tracciatura dei messaggi. La PdD deve registrare le buste scambiate e le operazioni effettuate in logs.
- Smistamento (Routing) dei messaggi. La PdD deve capire, attraverso informazioni incluse nella busta e-Gov a quale PdD remota è destinata la busta.

Inoltre, se esistono applicazioni presenti all'interno del dominio, che non erano originariamente state progettate per essere utilizzate con tecnologia Web Service, la PdD deve fornire funzionalità di integrazione attraverso opportuni Wrappers.
Riassumendo quanto detto, le porte di dominio possono essere viste come degli adattatori, che consentono a sistemi informatici esistenti, o comunque progettati e realizzati in base alle esigenze del dominio specifico, di affacciarsi a SPCoop e partecipare alla cooperazione applicativa.

1.3.2 La busta SPCoop

La busta di e-Government specifica il formato dei messaggi scambiati tra le Porte di Dominio nelle interazioni di cooperazione applicativa e ne costituisce l'elemento informativo di base. Una busta e-Gov è una specializzazione di un messaggio Soap con l'aggiunta opzionale delle estensioni riguardanti la presenza di attachments nel messaggio.

La Figura 1-9 illustra la struttura del messaggio SOAP che forma una Busta SPCoop:

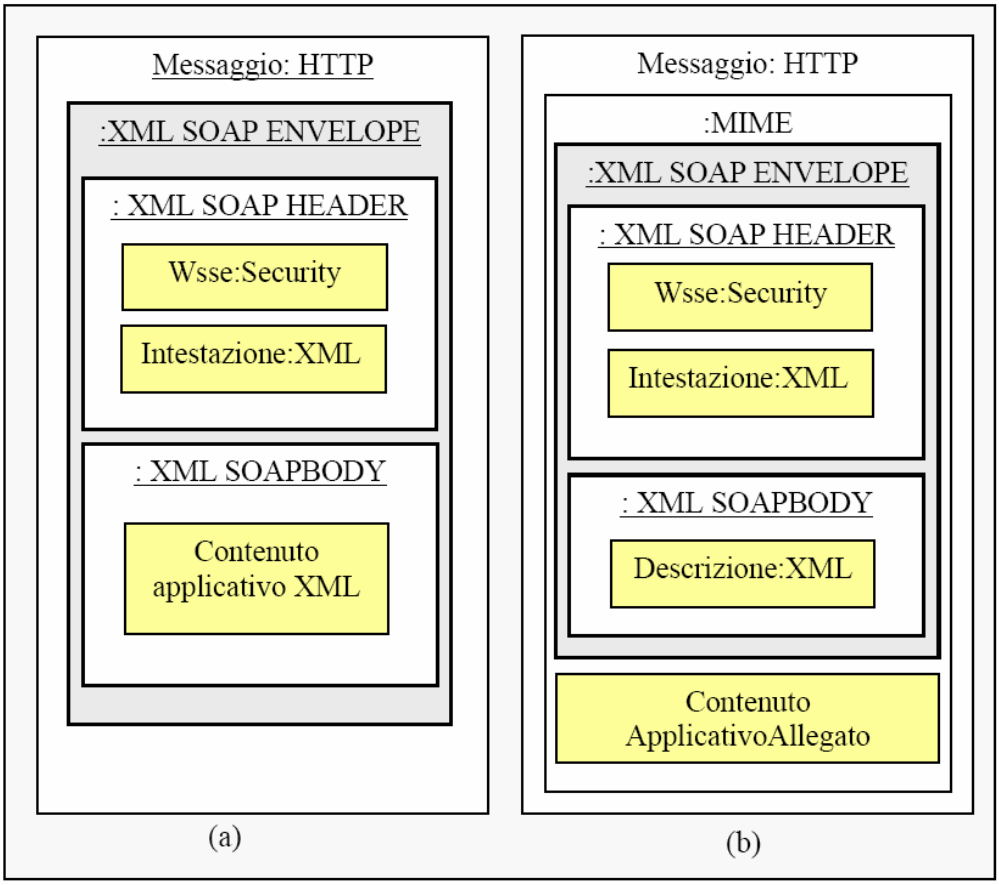


Figura 1-9, Struttura di una Busta SPCoop senza (a) e con (b) attachments

Per la completa definizione del contenuto e del formato di codifica, quindi, sono stati scelti dal CNIPA due strumenti standard come il linguaggio XML ed il protocollo SOAP. La struttura della parte più esterna del messaggio non è prefissata e dipende dal protocollo di trasporto utilizzato (HTTP, SMTP, ecc.), anche se le specifiche SOAP, pur prevedendo l'uso di diversi protocolli, definiscono esattamente il "binding" con il protocollo HTTP.

Il messaggio si compone di una parte di intestazione (Header) e di una parte specifica dell'applicazione. Nell'Header è definito un elemento "Intestazione" che contiene tutte le informazioni legate alla logica di gestione del messaggio. E' prevista la possibilità di inserire anche una o più firme digitali (una per ogni elemento firmato). Nel *Corpo (Body)*, è presente il dato applicativo da spedire.

La struttura SOAP del messaggio, inoltre, può essere inclusa in una struttura MIME allo scopo di allegare al messaggio uno o più documenti applicativi, in base alle specifiche SOAP with Attachments [W2, A6, A10], che prevedono l'aggregazione *multipart* in base allo standard MIME. In questo modo è possibile aggregare in un unico messaggio più blocchi distinti di informazioni anche eterogenee. Per la verifica dell'autenticità e della provenienza dell'attachment, e per gestire la segretezza dell'informazione, è previsto l'utilizzo dei formati PKCS#7, dove trovano posto l'eventuale firma di provenienza ed il certificato X.509 v3.
Lo standard CNIPA non specifica naturalmente quale debba essere il contenuto del body o degli attachments ma solo il formato di un elemento del Soap Header: la busta SPCoop. Di questo elemento, lo standard ne specifica l'insieme di elementi XML che lo compongono attraverso la stesura di un appropriato schema XSD [A16-A20]. La busta SPCoop forma il messaging layer di SPC, e cioè definisce il protocollo applicativo di SPCoop. Il CNIPA raccomanda, insieme all'header che definisce la busta SPCoop, di inserire nel Soap Header anche uno o più elementi Web Service Security (WSS) per gestire la sicurezza dei messaggi. Infatti la specifica WSS descrive una estensione dei messaggi Soap introducendo dei meccanismi di protezione, rispetto all'integrità, alla confidenzialità e all'autenticazione dei messaggi stessi.

Elementi di una Busta SPCoop. Come detto precedentemente, il CNIPA ha rilasciato uno schema XSD che definisce gli elementi che formano un header e-Gov. Vediamo un esempio di busta e-Gov, creata dalla porta di dominio con identificativo 'PDexampleSPCoopIT', in seguito ad una richiesta pervenuta da un SIL Mittente del dominio gestito (SIL1) che desidera usufruire dell'azione 'Ping' di un servizio 'Quote' erogato da un SIL destinatario di un altro dominio (SIL2):

```
<soapenv:Envelope
      soapenv:encodingStyle=http://schemas.xmlsoap.org/soap/encoding
      xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
<soapenv:Header>
<eGov_IT:Intestazione
  soapenv:actor="http://www.cnipa.it/eGov_it/portadominio"
  soapenv:mustUnderstand="1"
  xmlns:eGov_IT="http://www.cnipa.it/schemas/2003/eGovIT/Busta1_0/">
```

```
<eGov_IT:IntestazioneMessaggio>
      <eGov_IT:Mittente>
            <eGov_IT:IdentificativoParte tipo="SPC">
                  SIL1
            </eGov_IT:IdentificativoParte>
      </eGov_IT:Mittente>
      <eGov_IT:Destinatario>
            <eGov_IT:IdentificativoParte tipo="SPC">
                  SIL2
            </eGov_IT:IdentificativoParte>
      </eGov_IT:Destinatario>
      <eGov_IT:ProfiloCollaborazione >
            EGOV_IT_MessaggioSingoloOneWay
      </eGov_IT:ProfiloCollaborazione>
      <eGov_IT:Servizio tipo="test">Quote</eGov_IT:Servizio>
      <eGov_IT:Azione>Ping</eGov_IT:Azione>
      <eGov_IT:Messaggio>
            <eGov_IT:Identificatore>
                  SIL1_PDExampleSPCoopIT_0000004_2005-11-16_16:49
            </eGov_IT:Identificatore>
            <eGov_IT:OraRegistrazione tempo="EGOV_IT_SPC">
                  2005-11-16T16:49:07.982Z
            </eGov_IT:OraRegistrazione>
      </eGov_IT:Messaggio>
      <eGov_IT:ProfiloTrasmissione inoltro="EGOV_IT_PIUDIUNAVOLTA"
                        confermaRicezione="false" />
</eGov_IT:IntestazioneMessaggio>
<eGov_IT:ListaTrasmissioni>
<eGov_IT:Trasmissione>
      <eGov_IT:Origine>
            <eGov_IT:IdentificativoParte tipo="SPC" >
                  SIL1
            </eGov_IT:IdentificativoParte>
      </eGov_IT:Origine>
      <eGov_IT:Destinazione>
            <eGov_IT:IdentificativoParte tipo="SPC">
                  SIL2
            </eGov_IT:IdentificativoParte>
      </eGov_IT:Destinazione>
      <eGov_IT:OraRegistrazione tempo="EGOV_IT_SPC" >
            2005-11-16T16:49:07.982
      </eGov_IT:OraRegistrazione>
</eGov_IT:Trasmissione>
</eGov_IT:ListaTrasmissioni>
</eGov_IT:Intestazione>
</soapenv:Header>
<soapenv:Body>
 XXXXXXXXX  DATI APPLICAZIONE   XXXXXXXXXXXXXXX
</soapenv:Body>
</soapenv:Envelope>
```

La struttura di un header e-Gov possiede un componente obbligatorio (IntestazioneMessaggio), contenente le informazioni principali per la gestione del messaggio, e altri tre elementi opzionali, contenenti informazioni di supporto, come viene evidenziato nella Figura 1-10.

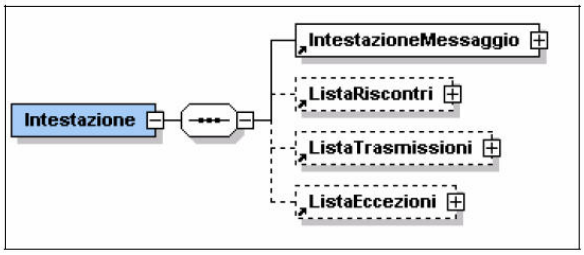


Figura 1-10, Struttura di un header e-Gov

L'elemento principale ed obbligatorio della busta è il tag 'intestazioneMessaggio' che ha la seguente struttura:

- **Mittente e Destinatario.** Rappresenta l'identificativo rispettivamente della parte mittente e della parte destinataria di una cooperazione applicativa.
- **Profilo Di Collaborazione** (opzionale). Indica il tipo di interscambio di messaggi utilizzato tra le porte di dominio tra i possibili seguenti valori:
  - *Messaggio Singolo OneWay*. La porta di dominio mittente invia un messaggio alla PdD destinataria senza attendere alcuna risposta.
  - *Servizio Sincrono*. La porta di dominio mittente invia un messaggio SPCoop contenente una richiesta applicativa alla porta di dominio destinataria. Dopodichè rimane in attesa attiva di un messaggio SPCoop contenente una risposta applicativa, sulla connessione utilizzata per la richiesta.
  - *Servizio Asincrono Simmetrico.* La porta di dominio mittente invia un messaggio contente una richiesta applicativa senza attendere una immediata risposta; la risposta potrà essere inviata, dalla porta di dominio destinataria, in un tempo successivo.
  - *Servizio Asincrono Asimmetrico.* La porta di dominio mittente invia un messaggio contenente una richiesta applicativa senza restare in attesa di una risposta; in un tempo successivo, la porta di dominio mittente richiede lo stato di esecuzione della propria richiesta alla porta di dominio destinataria rimanendo in attesa della risposta. Se il servizio è stato eseguito nel dominio destinatario, la relativa porta inoltrerà alla Porta di dominio mittente la risposta applicativa, altrimenti segnala che il processamento non è stato ancora completato e l'interrogazione sullo stato si ripeterà dopo un intervallo di tempo.

- **Collaborazione** (opzionale)**.** Possiede l'identificatore di una collaborazione (gruppo di messaggi) a cui appartiene il messaggio. Rappresenta l'identificatore del messaggio capostipite che ha originato i successivi messaggi.
- **Servizio** (opzionale)**.** Rappresenta l'identificativo logico del servizio richiesto / erogato. E' opzionale poiché non è presente nel caso di messaggio di servizio (es. riscontro di un precedente messaggio).
- **Azione** (opzionale). Specifica il tipo di azione richiesta. Deve essere univoca nell'ambito del servizio in cui è definita.
- **Messaggio.** Contiene i dati identificativi del messaggio. Possiede la seguente struttura:
    - **Identificatore.** Rappresenta l'identificatore unico del messaggio, ed è costruito tramite un numero sequenziale, una marca temporale, e l'identificativo del mittente e della porta di dominio.
    - **Ora Registrazione.** Rappresenta una marca temporale che indica il momento di creazione della busta e-Gov.
    - **Riferimento Messaggio** (opzionale)**.** Contiene un riferimento ad un messaggio precedentemente inviato/ricevuto. Il riferimento non è altro che l'identificatore del messaggio da riferire.
    - **Scadenza** (opzionale)**.** Rappresenta una marca temporale che indica una data di scadenza della busta e-Gov. Se la porta di dominio non è riuscita a consegnare la busta entro la scadenza associata, non deve riprovarci.
- **Profilo di Trasmissione** (opzionale). Descrive la modalità con cui il messaggio può essere processato dalla porta di dominio ricevente, al fine di garantire predefinite proprietà sulla consegna del messaggio all'applicazione destinataria. Contiene due attributi chiave:
    - *Conferma Ricezione*. Indica se l'avvenuta ricezione del messaggio deve essere riscontrata tramite l'invio di un messaggio di ritorno (acknowledgement).
    - *Inoltro*. Indica se uno stesso messaggio, in seguito ad eventuali re-invii di una porta di dominio mittente, può essere ricevuto in singola copia o in duplicata copia dall'applicazione destinataria.

- **Sequenza** (opzionale). In caso di collaborazione presente, indica il numero di sequenza all'interno di essa. Serve per realizzare una consegna in ordine di eventuali messaggi verso una applicazione destinataria.

Inoltre l'header e-Gov possiede anche altri tre elementi opzionali:

- **Lista Riscontri.** Contiene l'acknowledgment, da parte della porta di dominio destinataria, di uno o più messaggi precedentemente ricevuti. Nel riscontro viene indicato l'identificatore del messaggio precedentemente ricevuto e la data di riscontro.
- **Lista Trasmissioni.** Contiene informazioni relative al tracciamento del messaggio. Ogni volta che una porta di dominio prende in carico un messaggio deve inserire nella lista i dati relativi al passaggio, insieme ad una marca temporale.
- **Lista Eccezioni.** Contiene gli eventuali errori riscontrati nel processamento dell'header e-Gov. Tale tag deve essere accompagnato dal relativo SoapFault che, nel caso di errore, come da specifiche Soap, è obbligatorio.

1.3.3 Il Registro dei servizi

Tra due sistemi informatici si attiva una relazione di servizio, se uno dei due sistemi, che riveste il ruolo di fruitore, utilizza i risultati di trattamenti informatici effettuati dall'altro sistema, che riveste il ruolo di erogatore. Un servizio è l'insieme dei risultati prodotti dai trattamenti effettuati dal sistema erogatore ed utilizzati dal fruitore. Il risultato prodotto da un trattamento effettuato dall'erogatore prende il nome di "prestazione di servizio".
Le prestazioni di servizio appartengono a quattro categorie:

- prestazioni di calcolo, cioè la fornitura da parte dell'erogatore al fruitore di dati risultato di puri trattamenti di calcolo;
- prestazioni di informazione di stato, che prevedono la gestione da parte dell'erogatore degli stati e delle transizioni di stato di risorse su richiesta diretta o indiretta del fruitore;

- prestazioni di cambiamento di stato, determinate da caratteristiche specifiche di qualità di servizio degli stati e delle transizioni di stato delle risorse gestite dall'erogatore;
- prestazioni di interazione, che consistono in interazioni da parte del sistema erogatore con l'ambiente esterno (altri sistemi, utenti, dispositivi diversi) su richiesta diretta o indiretta del fruitore.

Un sistema può partecipare a più relazioni di servizio e ricoprire in queste relazioni i ruoli di erogatore e fruitore. Quindi un sistema può sia erogare un insieme di funzionalità implementandole al suo interno, sia fruire delle prestazioni fornite da altri sistemi dell'architettura. Nella Figura 1-11 viene raffigurata una generica relazione di servizio

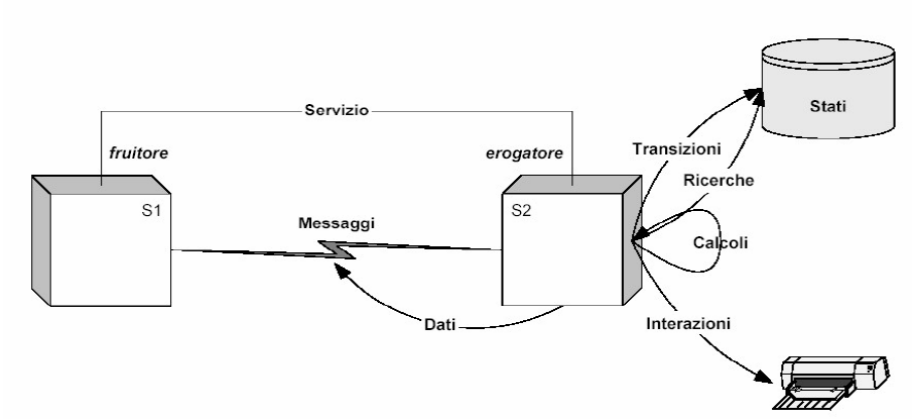


Figura 1-11, Raffigurazione di una generica relazione di servizio.

Prima di poter realizzare una relazione di servizio tra sistemi è necessario definire un accordo esplicito sull'erogazione e la fruizione delle prestazioni: **l'accordo di servizio** [CN5]. Tale accordo riguarda le prestazioni del servizio e le modalità di erogazione e di fruizione dello stesso.

In dettaglio, l'accordo di servizio si occupa di definire:

- **Identificazione delle parti**. L'accordo di servizio impone l'identificazione delle parti coinvolte nella erogazione/fruizione del servizio che possono coprire il ruolo di Erogatore, Fruitore o Intermediario. I titolari dei sistemi erogatori del servizio e gli intermediari devono essere identificati nell'accordo di servizio;
- **Descrizione delle funzionalità**. Fornisce una descrizione funzionale delle classi di prestazioni del servizio;

- **Descrizione delle interfacce di scambio dei messaggi**. Vengono definiti:
  - Protocolli di conversazione
  - Formato del corpo e della testata dei messaggi. In questo ambito sono definite le funzionalità SPCoop da utilizzare, come il profilo di collaborazione, l'identificativo logico del servizio e dell'azione ecc…
  - Descrizione del collegamento con il protocollo di connessione
  - Indirizzi (URI) dei punti di accesso.
- **Descrizione dei requisiti di sicurezza.** Sono evidenziati i requisiti di sicurezza dell'erogazione e della fruizione, elencando la necessità di requisiti di autenticazione, autorizzazione, controlli d'integrità e riservatezza, non ripudiabilità, ispezione ecc…
- **Descrizione dei requisiti di qualità del servizio.** I requisiti di qualità di servizio mirano a caratteristiche operative (non funzionali) dell'erogazione / fruizione del servizio e dello scambio dei messaggi. I requisiti di qualità possono riguardare vari aspetti come scadenze temporali, tracciamento, affidabilità dello scambio, ecc.. E' interessante notare come, ad esempio, a seconda del tipo di affidabilità che sarà scelto nell'accordo di servizio, la busta e-Gov scambiata tra le porte di dominio del sistema fruitore e del sistema erogatore conterrà elementi opzionali impostati a valori differenti. Nella Tabella 1-1 viene evidenziato il valore degli attributi 'inoltro' e 'confermaRicezione' dell'elemento 'Profilo di Trasmissione' e la presenza o meno degli elementi 'Collaborazione' e 'Sequenza' al variare dell'affidabilità di scambio impostata nell'accordo di servizio.

Affidabilità Scelta	**Inoltro senza duplicati**	**confermaRicezione**	**Collaborazione / Sequenza**
Nessuna	Non presente	Non presente	Non presente
Al più una volta	true	false	Non presente
Esattamente una volta	true	true	Non presente
Almeno una volta	false	true	Non presente
Esatt. una volta + consegna in ordine	true	true	presente

Tabella 1-1, Valori associati ad alcuni elementi della busta SPcoop, in relazione con il tipo di affidabilità associato ad un servizio.

La Figura 1-12 raffigura il contesto di un generico accordo di servizio.

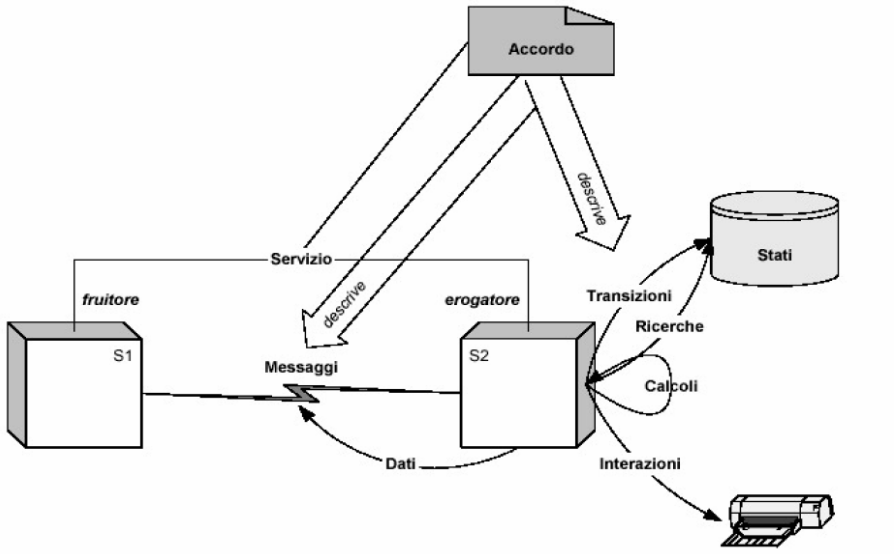


Figura 1-12, Contesto esemplificativo di un generico accordo di servizio.

La definizione di un accordo di servizio sarà conservata all'interno di un'altra infrastruttura di SPCoop: il **Registro dei servizi** [CN6]. Il registro permetterà la pubblicazione ed il recupero degli accordi di servizio, da parte rispettivamente dei pubblicatori e dei fruitori.

Nel Registro dei servizi, oltre agli accordi di servizio, sono contenuti altri aspetti quali:

- Indice dei domini che possono agire come fruitori od erogatori all'interno della infrastruttura SPCoop.
- Elenco dei domini erogatori.
- Rubrica dei punti di accesso alle porte di dominio che gestiscono sistemi eroganti.

Il registro dei servizi viene comunemente inteso come un registro realizzato tramite lo standard UDDI. Si prevede, attraverso l'utilizzo di una semplice interfaccia grafica (es. Web), la gestione e la consultazione del registro da parte degli amministratori dei domini, per eseguire sia il processo di pubblicazione da parte dell'amministrazione erogatrice che quello di adesione da parte delle amministrazioni riceventi. Le amministrazioni che pubblicano i servizi possono esporre un nuovo servizio inserendo, attraverso l'interfaccia, i dati e le informazioni previsti dal registro per ogni nuova pubblicazione.

La Figura 1-13 illustra lo scenario in cui è presente un accordo di servizio tra un unico fruitore ed un unico erogatore:

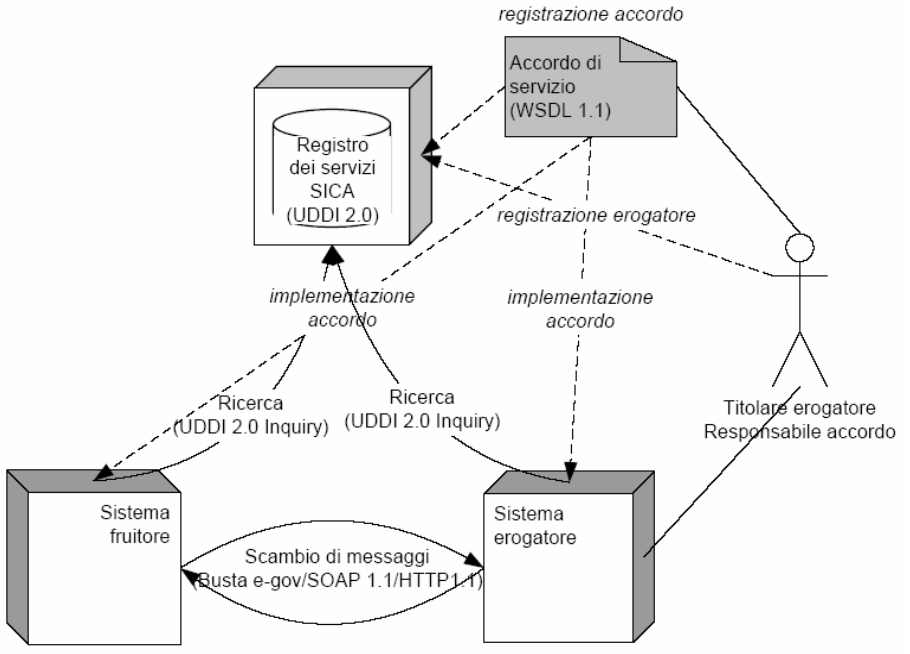


Figura 1-13, Scenario esemplificativo di un accordo di servizio tra un fruitore ed un erogatore

1.3.4 Paradigma di cooperazione basata su eventi

La tecnologia utilizzata da SPCoop, i WebServices, presuppone come paradigma di cooperazione quello dell'architettura di servizi (SOA) che si basa su un modello di comunicazione punto a punto. Questo limita notevolmente gli scenari di cooperazione possibili, ed inoltre proprio alcuni scenari amministrativi necessitano di modelli di cooperazione più complessi che coinvolgano più di due attori. A tale scopo il CNIPA ha suggerito un altro attore nell'architettura SPCoop: il **Gestore degli eventi**.

Con questo ulteriore componente, è possibile effettuare cooperazione basata su eventi. Esistono due categorie di soggetti in gioco, i pubblicatori che generano gli eventi ed i sottoscrittori che li consumano.

Per implementare tale modello, il Gestore degli eventi deve disporre di due tipi di funzionalità:

- La pubblicazione di un determinato tipo di evento da parte di un dominio
- La sottoscrizione di un dominio per ricevere notifica di eventi. La sottoscrizione può essere ulteriormente classificata in base alla modalità di ricezione dell'evento:
    - Il consumatore può restare in attesa attiva, con il gestore, per la ricezione degli eventi (push);
    - Il consumatore può interrogare il gestore circa l'arrivo di nuovi eventi a lui destinati (polling);
    - Il consumatore può interrogare il gestore circa l'arrivo di determinati tipo di eventi (polling selettivo);
    - Il consumatore può segnalare al gestore la propria disponibilità a ricevere gli eventi a lui destinati (push sollecitato);
    - Il consumatore può segnalare al gestore la propria disponibilità a ricevere solo certi tipo di eventi (push sollecitato e selettivo).

1.3.5 Il Componente di Integrazione SIL-to-PdD

I documenti di specifica forniti dal CNIPA illustrano come deve essere definita l'architettura SPCoop, evidenziandone i componenti necessari per lo scambio di messaggi SPCoop tra Porte di Dominio, chiamando l'insieme dei componenti adibiti a tale scopo: 'componente di cooperazione'. Mettono inoltre in risalto l'accordo di servizio, che descrive come un servizio viene erogato e da quali soggetti, fornendo un componente il cui compito è la gestione dei vari accordi di servizio esistenti, il Registro dei Servizi. L'unico aspetto dove la specifica CNIPA non entra nel dettaglio, è con quale protocollo i SIL di un dominio debbano interagire con la Porta di Dominio. I documenti di specifica parlano costantemente di una 'componente di integrazione' che deve essere realizzata per permettere al SIL di poter parlare con la porta di dominio.

La visione "classica" del CNIPA della porta di dominio utilizzata per la cooperazione applicativa è rappresentata nella Figura 1-14:

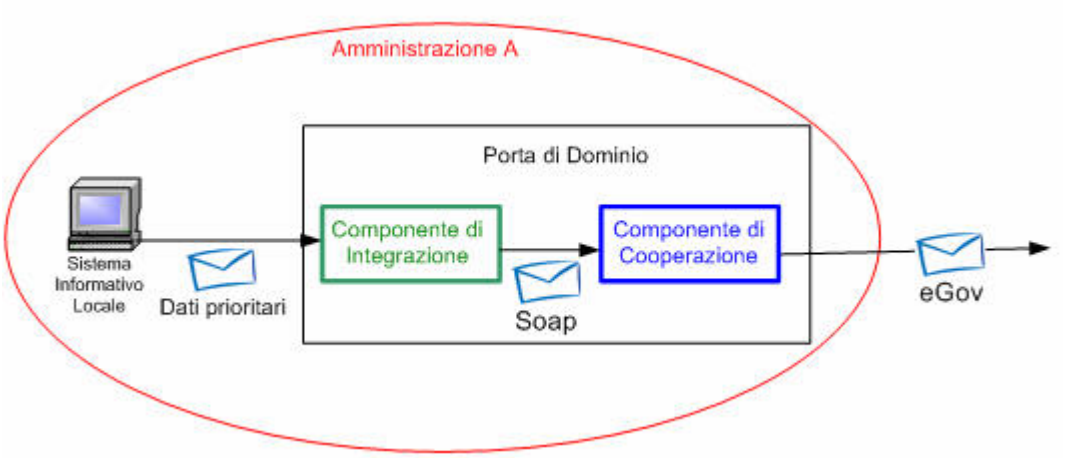


Figura 1-14, Visione 'classica' del CNIPA per la Porta di Dominio (Componente di Integrazione e Componente di Cooperazione)

Si individua, all'interno di ogni Porta di Dominio, una componente di integrazione, responsabile di risolvere i problemi di interoperabilità con i Sistemi Informativi Locali, e una di cooperazione, responsabile di trattare le comunicazioni secondo lo standard della busta e-Gov. Nella sezione 1.4 saranno illustrati alcuni possibili approcci alla realizzazione del componente di integrazione, per giungere infine a presentare la soluzione adottata in OpenSPCoop PdD, la porta di dominio che è oggetto di discussione di questa tesi.

1.4 Possibili architetture del Componente di Integrazione

I documenti di specifica CNIPA individuano, all'interno di ogni Porta di Dominio, una componente di integrazione, responsabile di risolvere i problemi di interoperabilità con i Sistemi Informativi Locali, e una di cooperazione, responsabile della gestione delle comunicazioni secondo lo standard della busta e-Gov.

Le specifiche SPCoop, tuttavia, trattano solamente l'aspetto di come le porte debbano comunicare tra di loro (componente di cooperazione), senza affrontare il tema di come vada gestita la parte di integrazione. Sono presenti nei documenti di specifica solo accenni o esempi di funzionalità che il componente di integrazione dovrebbe effettuare. Vediamo nel seguito di questa sezione alcuni possibili approcci alla realizzazione del componente di integrazione, per giungere infine a presentare la soluzione adottata in OpenSPCoop.

Cooperazione tra Sistemi Legacy.

Nella Figura 1-15 viene riportato uno schema completo di interazione nella cooperazione applicativa, dove si mostra un primo possibile approccio per realizzare la componente di integrazione della porta.

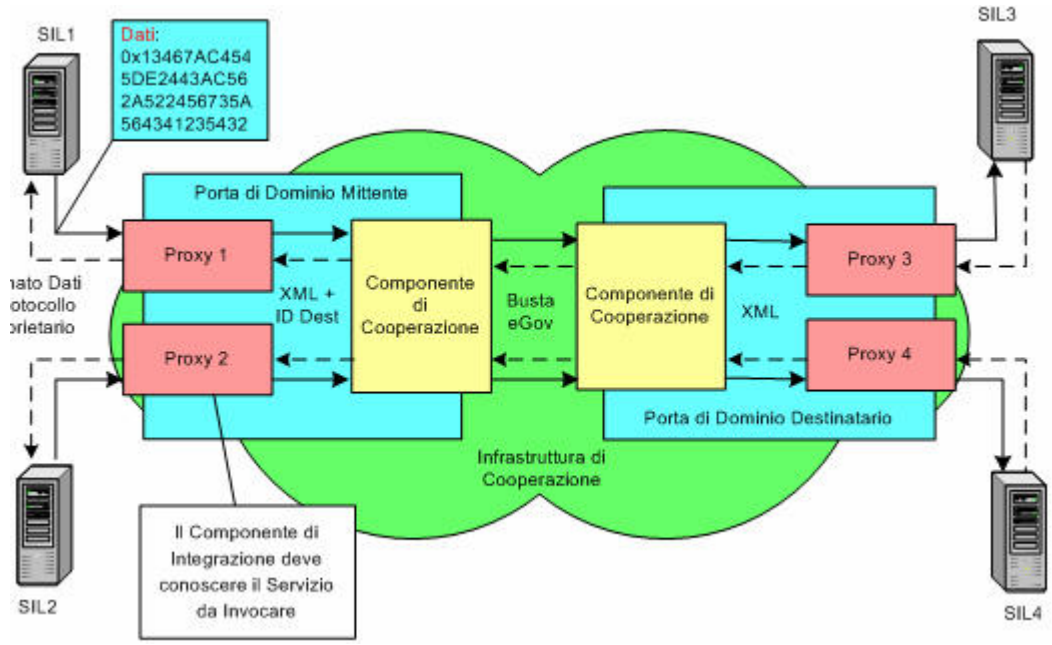


Figura 1-15, Scenario di cooperazione tra Sistemi Legacy

La prima cosa da notare nella Figura 1-15 è che i due sistemi applicativi locali (richiedente ed erogante del servizio) non possono comunicare in maniera diretta per almeno due motivi:

- i due server appartengono a due reti private diverse, e non possono quindi interoperare direttamente tra di loro;
- se anche fosse possibile stabilire una comunicazione diretta, esiste un quadro normativo che impone che l'interazione telematica tra applicativi di enti appartenenti a domini diversi, debba essere, per avere validità legale, realizzata tramite SPCoop.

La necessità di far comunicare i due Server tramite SPCoop pone il problema di come trasportare, all'interno delle buste e-Gov, i dati proprietari delle richieste e delle risposte dei due sistemi. Una prima soluzione è quella mostrata nella Figura 1-15, dove vengono realizzati dei proxy, **sviluppati ad-hoc per ogni SIL,** che funzionano come componente di integrazione per interfacciare il SIL fruitore od erogatore del servizio alla propria porta di dominio. I proxy si occuperanno, rispettivamente, di incapsulare e deincapsulare i dati proprietari in un formato XML, poi utilizzato dai componenti di cooperazione come body della busta e-Gov. Il proxy di un SIL fruitore di un servizio ha anche un secondo compito che è quello di conoscere l'indirizzo del destinatario reale del SIL che ne eroga le prestazioni, a cui la richiesta dovrà essere diretta. In qualche modo, quindi, la destinazione è *cablata* nel proxy del SIL fruitore.

Cooperazione tra Sistemi Web Services

Di recente, la grande diffusione dell'XML e dei Web Services, che usano l'XML non solo per il trasporto dei dati (buste SOAP), ma anche per la definizione del loro formato (schemi XSD), permette di semplificare la realizzazione del Componente di Integrazione. Si può infatti considerare un sistema interno già capace o facilmente adattabile al dialogo tramite Web Services o anche più semplici POST HTTP che ne simulino il comportamento. In tal caso la componente di integrazione non dovrà usare tecnologie diverse per ogni possibile sistema legacy da interfacciare, ma potrà essere invece un generico container (ad esempio J2EE) capace di ospitare web services che fungano da proxy per le richieste di servizio formulate dai server interni. Una soluzione di questo tipo, viene rappresentata nella Figura 1-16.

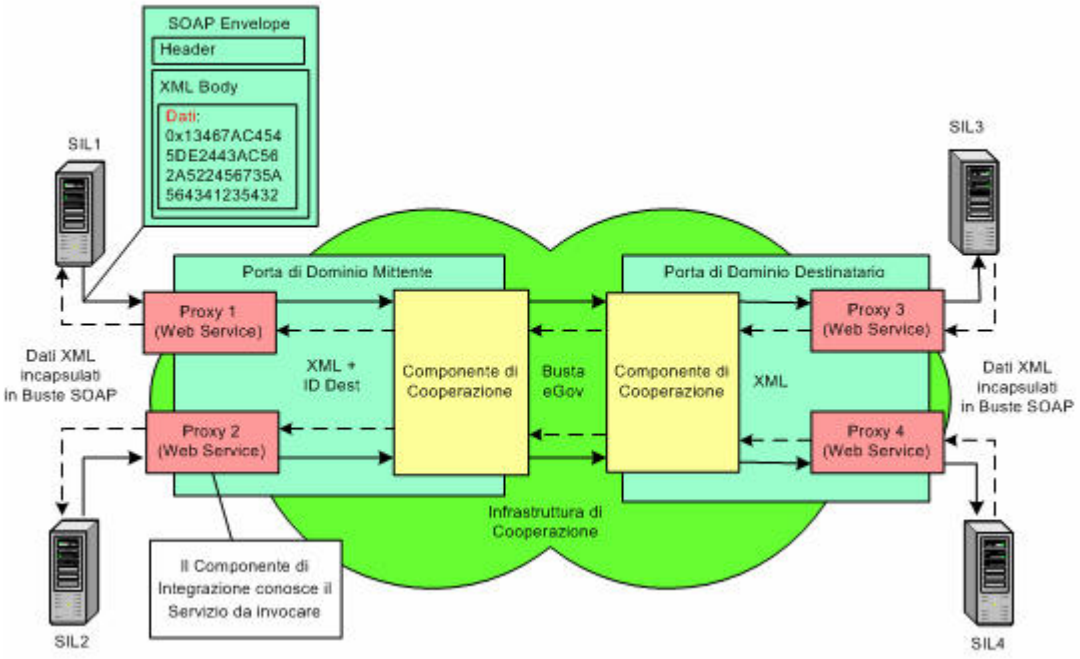


Figura 1-16, Scenario di cooperazione con Web Services

Lo scenario di cooperazione con Web Services ha il vantaggio di semplificare la realizzazione dei proxy, ma richiede ancora che il proxy del SIL fruitore conosca l'indirizzo di destinazione dove il servizio viene erogato, in modo da potergli girare le richieste in arrivo dal sistema informativo locale. Sarà ancora necessario, quindi, realizzare ed installare un proxy associato ad ogni SIL, come nello scenario precedente.

Cooperazione tra sistemi attraverso un Servizio Generico di Imbustamento

Una possibile soluzione al problema dell'identificazione del servizio di destinazione è mostrata in Figura 1-17. Si tratta di realizzare un web service generico di imbustamento, che prenda in input il generico XML da inviare più le informazioni necessarie a individuare il Servizio destinazione. In questo modo non è più necessario realizzare un proxy per ogni SIL fruitore di un servizio, ma tutti i client possono utilizzare questo unico servizio di spedizione.

Per quanto questa soluzione possa apparentemente sembrare una soluzione valida, in effetti si tratta di una soluzione di fatto impraticabile. Come evidenziato in Figura 1-17, con questa soluzione il SIL fruitore genererà una richiesta SOAP contenente oltre ai dati normalmente prestabiliti anche informazioni per l'identificazione del destinatario.

La soluzione seguente costringe quindi il SIL ad avere la visibilità dell'infrastruttura di trasporto, infrastruttura che la porta di dominio dovrebbe per sua stessa definizione nascondere. E' facile immaginare il forte impatto negativo di una soluzione così intrusiva nell'uso da parte dei Sistemi Informativi Locali.

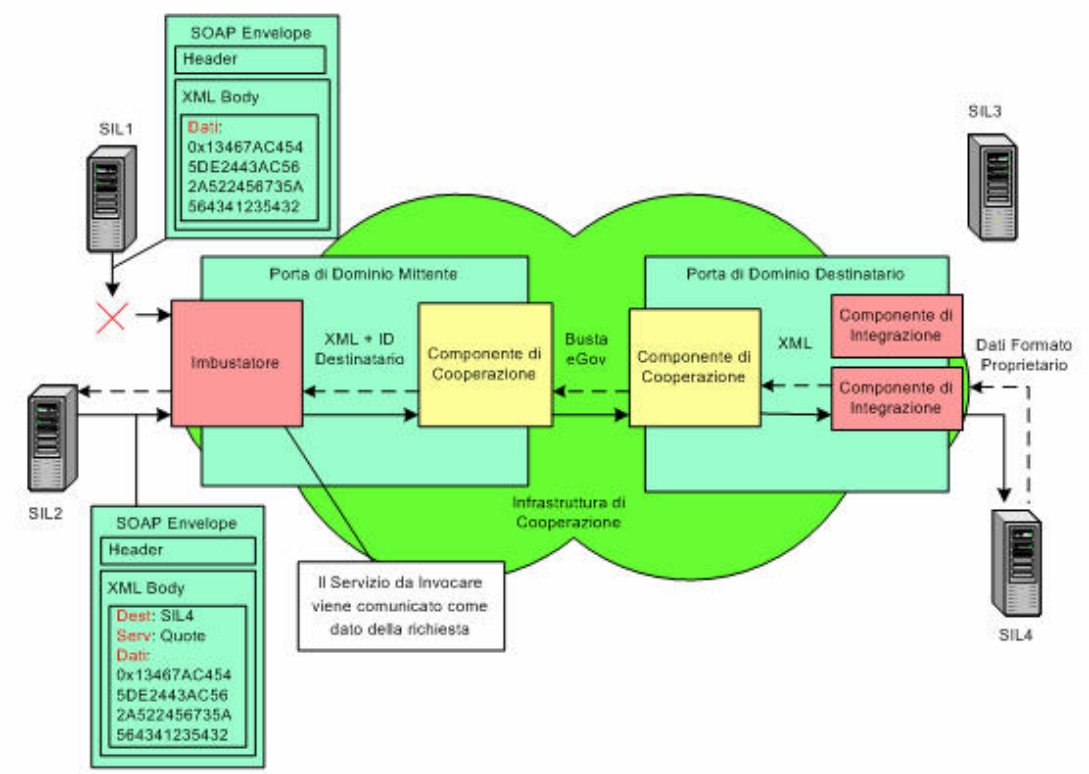


Figura 1-17, Scenario di cooperazione attraverso un servizio generico di imbustamento

Cooperazione tra sistemi attraverso un Proxy Dinamico

Lo studio di possibili interfacciamenti SIL-to-PdD è stata oggetto di tesi di un mio collega, Ruggero Barsacchi, che evidenziando i pregi/difetti di varie possibili interfacce, è giunto ad un soluzione del problema implementando un Proxy dinamico, capace di identificare dinamicamente il Servizio Destinatario appoggiandosi a delle informazioni di supporto: **tabella di porte delegate/applicative** registrate sulla Porta di Dominio. **Una porta delegata permette ad un SIL interno ad un dominio di usufruire di un servizio erogato da un SIL risiedente in un altro dominio**.

Una porta delegata è indirizzabile, da un SIL, attraverso una semplice invocazione di servizio esistente su di una determinata 'url'. Contiene le informazioni di trasporto necessarie alla porta di dominio per spedire una busta e-Gov, che conterrà i dati XML pervenuti al momento dell'invocazione della porta delegata stessa. Le informazioni di trasporto saranno utilizzate, oltre che per la costruzione della busta SPCoop, anche per ottenere l'indirizzo della porta di dominio destinataria a cui spedire la busta, attraverso una interrogazione del registro dei servizi. Così facendo i sistema legacy (SIL), non devono essere modificati per dialogare con la porta di dominio, e soprattutto non hanno visibilità dell'infrastruttura di trasporto. La porta delegata sarà quindi la chiave per l'interfaccia SIL-to-PdD, mentre sarà necessaria anche la definizione di una porta applicativa per risolvere l'interfacciamento PdD-to-SIL. **Una porta applicativa permette ad una porta di dominio, che ha ricevuto una busta SPCoop, di identificare il SIL, interno al dominio, erogatore del servizio richiesto nella busta.** La porta applicativa sarà indirizzabile attraverso informazioni contenute nella busta e-Gov, e conterrà informazioni sulla consegna del XML pervenuto (sbustato da informazioni di trasporto SPCoop), quali ad es. una url, una modalità di consegna (es. HTTP, SOAP….) ecc…

Questa soluzione, illustrata in Figura 1-18, resta una soluzione generica, come nel caso del servizio di imbustamento, e non richiede quindi l'installazione di un proxy (componente di integrazione) per ogni servizio da utilizzare (porta delegata) o da fornire (porta applicativa), ma non ha il problema della soluzione precedente, lasciando l'interfaccia d'uso del servizio esattamente la stessa del servizio originale. Il proxy dinamico interviene quindi come un proxy SOAP trasparente, in grado di assicurare tutti gli stessi livelli di servizio di un proxy applicativo, senza bisogno di realizzare applicazioni ad hoc. Questa soluzione sarà adottata nel progetto OpenSPCoop discusso in questa tesi, per risolvere l'interfacciamento SIL-to-PdD.

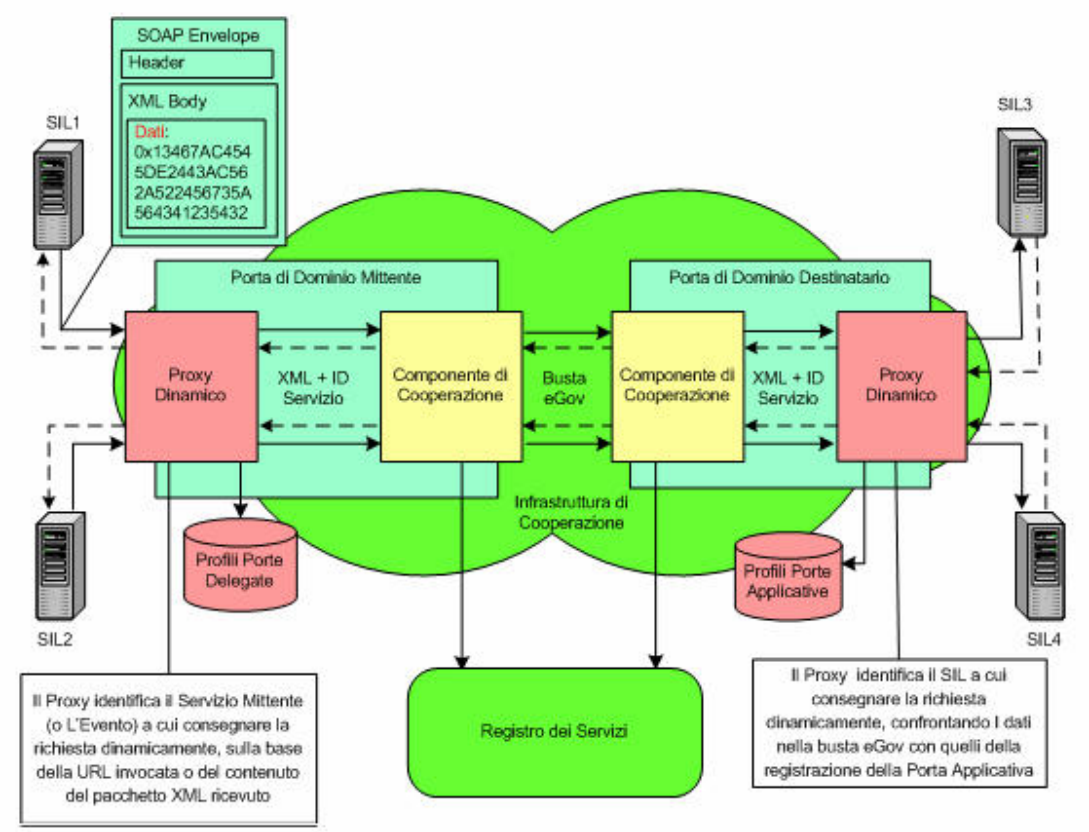


Figura 1-18, Scenario di cooperazione attraverso un proxy dinamico

2. Architettura del sistema OpenSPCoop

Il presente capitolo descrive la progettazione del sistema OpenSPCoop, adottando come stile di presentazione un approccio top-down.
Nella sezione 2.1 viene introdotta una visione d'insieme del sistema evidenziando tutti gli attori coinvolti in OpenSPCoop. Non si entra nel dettaglio di progettazione, ma si cerca di comprendere tutti i possibili scenari esistenti. La sezione 2.2 descrive l'architettura di OpenSPCoop, esaminandone i componenti nel dettaglio progettuale. Nella sezione 2.3 vengono presentati dei semplici casi d'uso che cercano di illustrare il funzionamento dell'architettura nel suo insieme. I casi esemplificati riguarderanno una semplice invocazione di servizio ed un contesto ad eventi.

2.1 Una visione d'insieme di OpenSPCoop

La progettazione di OpenSPCoop è stata effettuata cercando di fare cooperare tutti i soggetti descritti dalle specifiche CNIPA, quali le porte di dominio, il registro dei servizi ed il gestore degli eventi. Nella sezione 2.2 verranno esaminati in dettaglio ogni singolo elemento in gioco. La Figura 2-1 ilustra una visione d'insieme di tutte le componenti distribuite di OpenSPCoop. Come si può notare, l'unico mezzo di accesso per un Sistema Informativo Locale, per arrivare a consultare un altro SIL è la busta e-Gov, attraverso la mediazione della propria porta di dominio. Nella Figura 2-1 viene illustrato, tra i vari componenti, pure il gestore degli eventi, che sarà utilizzabile, da un SIL, per pubblicazioni o ricezioni di eventi attraverso la solita mediazione della porta di dominio ed attraverso l'utilizzo della busta e-Gov.

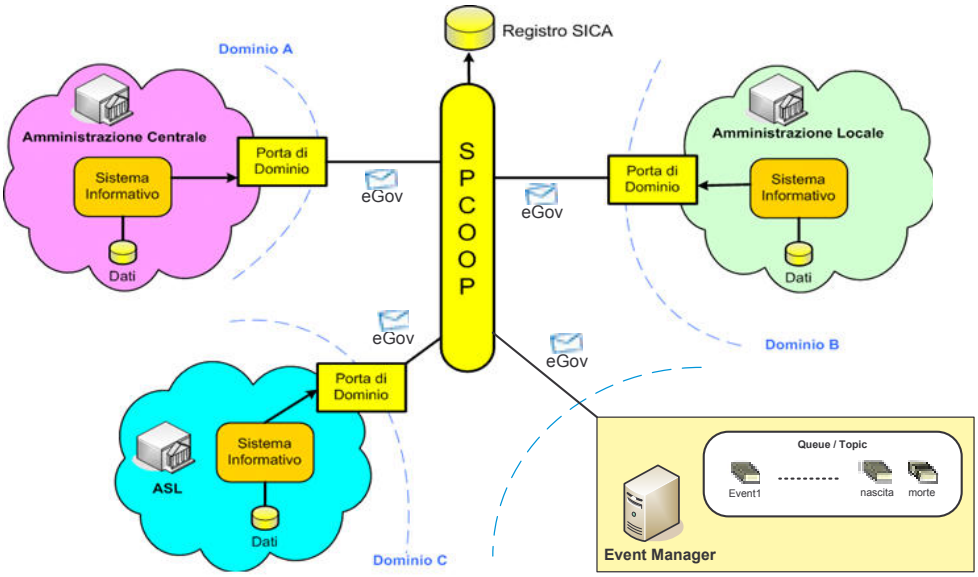


Figura 2-1, Architettura del sistema OpenSPCoop

2.2 Componenti dell'architettura

Per avere una visione progettuale di ogni singolo componente, viene elencata di seguito una descrizione funzionale generica di ognuno.

La **Porta di Dominio** (paragrafo 2.2.1) è costituita dal componente periferico OpenSPCoop PdD, ospitato da macchine appositamente create ed impostate nel dominio di gestione. La caratteristica principale di questo componente è il disaccoppiamento totale della componenti di ricezione e consegna delle buste da quella di controllo e gestione delle buste SOAP/e-Gov in transito, tramite l'uso di code persistenti per il mantenimento dei messaggi, realizzate tramite JMS [A27-A30] (sezione 3.2.2). Sarà OpenSPCoop PdD ad interagire direttamente sia con i SIL che con i componenti centrali dell'infrastruttura (Gestore degli eventi e Registro dei servizi). L'adozione di una porta di dominio, strumento di mediazione della comunicazione tra SIL, permette di assicurare un maggior controllo dal punto di vista della sicurezza e del tracciamento dei messaggi.

Come descritto nella sezione 1.4, verrà adottato un **componente di integrazione SIL-to-PdD composto da un proxy dinamico** (paragrafo 2.2.2), che richiede la presenza di una tabella per la registrazione di porte delegate e applicative necessarie a OpenSPCoop PdD per sapere in che modo gestire le richieste, sia in uscita (porte delegate), che in ingresso (porte applicative).

Le porte di dominio, oltre che dialogare con altre porte di dominio, potranno accedere ad un **gestore di eventi** (paragrafo 2.2.3) direttamente tramite listener in ascolto su code (o topic) [A27] interne al gestore, ottenendo una maggiore efficienza di prestazioni, o anche interagendoci tramite scambio di buste e-Gov, vedendo il gestore come un servizio che si occupa di trasmettere le comunicazioni di evento tra i soggetti interessati (sistemi informativi locali). Il Gestore degli eventi è strutturato quindi come una porta applicativa, il cui compito è ricevere e consegnare messaggi contenenti eventi, utilizzando la busta SPCoop.

Un componente molto importante della PdD, è quello che si occupa di gestire la logica SPCoop. La sua progettazione ha richiesto un tempo di studio ed analisi notevole, poiché cerca di trattare la specifica CNIPA nel dettaglio, sviscerandone tutti i vari aspetti. Il componente in questione, nominato **Modulo di Controllo** (paragrafo 2.2.4), viene chiamato in causa all'arrivo di una richiesta di servizio da parte di un SIL, prima di redirigere il messaggio verso un'altra porta di dominio, e all'arrivo di una busta SPCoop che includa una richiesta di servizio destinata ad un SIL. Tra i vari compiti del modulo di controllo vi sono tutti i vari aspetti di gestione della specifica CNIPA riguardante la busta SPCoop. Verrà quindi effettuata una validazione della busta (controllo semantico e sintattico), un controllo di autorizzazione, una eventuale gestione dei duplicati (affidabilità), una gestione dei riscontri (alla ricezione dei messaggi e alla spedizione della conferma ricezione), una gestione del tracciamento del messaggio e una gestione delle eccezioni. Il modulo di controllo si occupa anche di interagire con il registro dei servizi per conoscere le varie informazioni del servizio, sia nel momento dell'imbustamento, per effettuare la creazione della busta, sia al momento dello sbustamento, per effettuare una validazione della busta. Inoltre, durante l'accesso al registro per ottenere informazioni di imbustamento, verrà effettuata una ulteriore ricerca per conoscere l'indirizzo fisico della porta di dominio destinataria a cui spedire la busta SPCoop prodotta.

Il **Registro dei Servizi** (paragrafo 2.2.5) è costituito da un server LDAP [A54, A55], che mantiene le informazioni relative ai servizi e ai diritti di accesso dei Sistemi Informativi Locali ai servizi. E' stato inserito nell'architettura un registro UDDI [A13, A14] compatibile con le specifiche SPCoop ed utilizzato da OpenSPCoop PdD per identificare le caratteristiche del Servizio, come il profilo (sincrono, asincrono, simmetrico, asimmetrico) e gli altri parametri di quality of service. Un punto di forza dell'architettura proposta è l'uso da parte dell'UDDI del backend LDAP per immagazzinare i dati, in modo da evitare duplicazioni di informazioni.

2.2.1 Porta di Dominio

Il componente OpenSPCoop PdD implementa le funzionalità di Porta di Dominio dell'architettura SPCoop, fungendo quindi da intermediario tra i Sistemi Informativi Locali (SIL) e i servizi esterni con cui i SIL interagiscono. Per riuscire a supportare diverse possibili tipologie di interazione tra SIL e PdD (descritte in seguito), nonché tutte le complesse interazioni tra le porte di dominio, OpenSPCoop PdD utilizza una sofisticata architettura costituita da numerosi componenti, ognuno ottimizzato per uno specifico task, che interagiscono tra loro per mezzo di code di tipo persistente. I messaggi possono giungere alla porta di dominio, dai SIL interni, come **richieste destinate a servizi erogati in altri domini di cooperazione (Porte Delegate)** o, da altre porte di dominio, come **buste e-Gov in arrivo per servizi ospitati nel dominio di cooperazione servito (Porte Applicative)**. Dopo l'arrivo, le buste attraversano un pipeline di moduli, arricchendosi di informazioni utili al loro trattamento, mantenute negli header dei messaggi scambiati nelle code. L'architettura generale del componente OpenSPCoop PdD è schematizzato nella Figura 2-2. In seguito verranno esaminati i vari componenti.

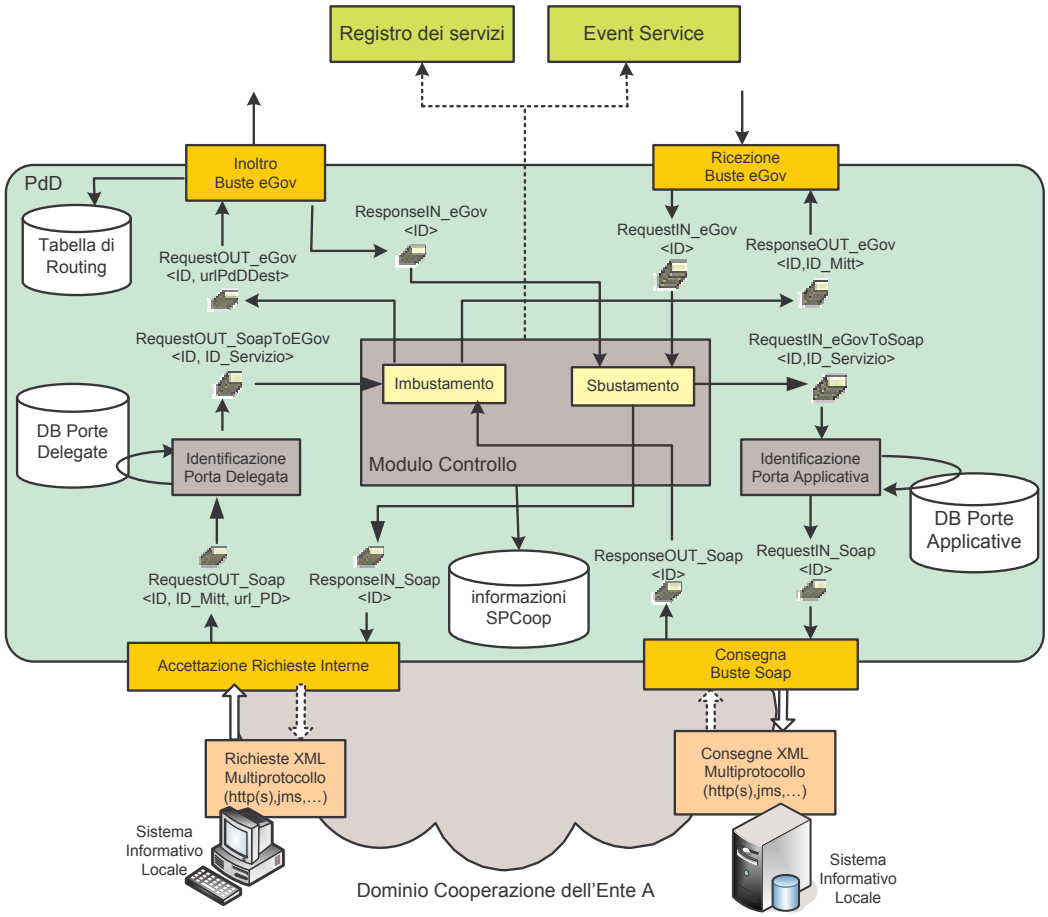


Figura 2-2, Architettura della porta di dominio OpenSPCoop

Modulo di Accettazione Richieste Interne.

Questo componente si occupa di prendere in carico le richieste dei SIL per le porte delegate registrate sulla porta di dominio. Per ogni richiesta ricevuta (i possibili protocolli tra SIL e PdD possono essere HTTPS, inserimento in code persistenti, web services, ecc…) viene effettuata l'autenticazione del mittente a livello trasporto, tramite il certificato X.509 usato per la connessione SSL. Il modulo di Accettazione dovrà essere in grado di gestire richieste sia di tipo SOAP che di tipo XML, in modo da essere compatibile sia con client di tipo Web Services che con client che usano semplici POST HTTPS. Se il contenuto non è già una busta SOAP si occuperà quindi di creare la busta SOAP e memorizzarla nella coda 'RequestOUT_Soap', associandoci, come proprietà del messaggio, l'identificatore del mittente, la URI utilizzata per l'invocazione della porta delegata ed un identificatore univoco creato appositamente.

Se siamo in presenza di una richiesta di tipo sincrono, il thread che sta gestendo la richiesta del SIL resta in ascolto sulla coda 'ResponseIN_Soap', filtrando la ricezione al valore della proprietà dell'identificatore unico del messaggio che attende di ricevere dalla coda. Non appena gli sarà notificata la disponibilità di una risposta potrà quindi rispondere al SIL in attesa, eventualmente estraendo dalla busta SOAP la sola parte di *Body*, nel caso abbia a che fare con una normale richiesta POST HTTPS e non Web Services.

Modulo di Identificazione della Porta Delegata.

La principale funzione di questo componente è quella di identificare a quale porta delegata sia relativa la richiesta in arrivo. Il componente di integrazione di OpenSPCoop PdD è infatti un proxy dinamico, in grado di gestire dinamicamente la richiesta senza alcun bisogno di proxy dedicati all'integrazione. Quindi tutte le richieste giungono ad OpenSPCoop, che deve essere poi in grado di capire a che richiesta di servizio (o tipologia di evento) corrispondono, interagendo con il DB delle porte delegate registrate. La chiave di accesso alla tabella delle porte delegate è costituita dalla parte finale dell'URI di accesso a OpenSPCoop PdD, utilizzata dal SIL che richiede un servizio. Ad esempio, supponendo l'indirizzo di accesso alla porta di dominio, configurato come 'http://www.openspcoop.org/pdd', il SIL per invocare una porta delegata registrata con l'URI 'servizioDiEsempio' dovrà utilizzare l'indirizzo 'http://www.openspcoop.org/pdd/**servizioDiEsempio**'.

Il risultato dell'identificazione sarà una tripla (Destinatario, Servizio e tipo, Azione) che identifica univocamente una richiesta di servizio nel registro dei servizi. Il valore viene memorizzato come un'ulteriore proprietà del messaggio, che viene poi inserito nell'apposita coda di ingresso del modulo di controllo ('RequestOUT_SoapToEGov').

Modulo di Controllo.

E' il nucleo centrale del sistema, occupandosi dell'autorizzazione, del tracciamento, e soprattutto del trattamento delle buste in transito, che possono richiedere un semplice imbustamento/sbustamento, nei casi più semplici, fino all'implementazione di sofisticate politiche di gestione delle funzionalità SPCoop, nei casi più complessi.

Il Modulo di Controllo è implementato come un pool di thread attivati dalla presenza di nuove buste in una delle quattro code previste di ingresso. Dopo il trattamento della busta pervenuta, il risultato delle operazioni prodotte dal modulo sarà depositato, in funzione della logica della busta gestita, in una delle quattro code previste in uscita. La logica operativa del modulo è suddivisibile in due grandi aree, una delle quali si occupa di gestire buste SPCoop ricevute, mentre l'altra si occupa di creare buste SPCoop in seguito all'arrivo di richieste di servizio dai SIL interni al dominio di gestione.

Per ogni messaggio e-Gov ricevuto il Modulo di Controllo effettua innanzitutto la validazione del messaggio, come definito nella standard CNIPA, che impone che il destinatario del messaggio non è autorizzato a intraprendere alcuna operazione di trattamento di esso, prima di aver completato la verifica di formato, sintattica e semantica del messaggio stesso. Viene inoltre verificato che il mittente del messaggio sia autorizzato a interagire con il servizio richiesto, interagendo con il registro dei servizi, depositario delle informazioni sui ruoli e sulle politiche di accesso. Infine viene effettuato il trattamento del messaggio, e cioè vengono intraprese l'insieme delle operazioni che il destinatario deve compiere alla ricezione del messaggio, nell'ambito del coordinamento della erogazione/fruizione del servizio a cui partecipa. Questa fase si occupa quindi della gestione di:

- stato del profilo di collaborazione (sincrono, asincrono, simmetrico, asimmetrico);
- eliminazione di buste ricevute in duplice copia;
- gestione delle sequenze nell'ambito di una Collaborazione Applicativa;
- gestione dei riscontri (acknowledgement)
- Tracciamento del messaggio;
- Eccezioni.

Il modulo di controllo, oltre alle code persistenti dove sono depositate le buste da gestire, utilizza un DB di appoggio, indicato in Figura 2-2 come '*Informazioni SPCoop'*, necessario per tenere traccia delle interrelazioni tra le varie buste nella gestione degli scenari più complessi. Ognuna di queste opzioni sarà analizzata in dettaglio nel paragrafo 2.2.4.

Modulo di Ricezione delle Buste e-Gov.

Questo componente si occupa di prendere in carico le buste e-Gov in arrivo da altre Porte di Dominio. Per ogni richiesta ricevuta viene effettuata l'autenticazione del mittente a livello trasporto, tramite il certificato X.509 usato per la connessione SSL. Le buste ricevute vengono immagazzinate nella coda 'RequestIN_eGov', marcate con una proprietà che identifica il mittente (oltre ad un identificativo che marca la gestione della richiesta all'interno della porta di dominio), per la successiva gestione da parte del Modulo di Controllo.

Modulo di Inoltro delle Buste e-Gov.

Questo componente è implementato tramite un pool di thread che ha il compito di inviare le buste e-Gov preparate dal modulo di Controllo, alle Porte di Dominio competenti, prelevandole nella coda 'RequestOUT_eGov'. Gli indirizzi fisici delle porte di dominio destinatarie sono ricavati o attraverso la lettura di una proprietà del messaggio estratto dalla coda (in questo caso l'indirizzo è stato prelevato dal registro dei servizi dal modulo di controllo) o dalla Tabella di Routing. La tabella di Routing verrà ad esempio utilizzata sulle PdD per forzare la spedizione di buste verso una porta di Dominio Proxy che funziona come relay per tutte le comunicazioni esterne.

Modulo di Identificazione della Porta Applicativa.

La principale funzione di questo componente è quella dell'identificazione di un servizio interno al dominio di gestione (Porta Applicativa) a cui è destinata la busta SPCoop pervenuta. Per identificare il servizio applicativo da invocare, si utilizza la tripla [Destinatario,Servizio,Azione], precedentemente estratta dal Modulo di Controllo e memorizzata all'interno dell'apposita proprietà del messaggio estratto dalla coda 'RequestIN_eGovToSoap'. Questa informazione viene quindi confrontata con i valori della tripla [ServiceProvider,Servizio,Azione] presenti nella tabella delle porte applicative registrate in OpenSPCoop PdD. Non appena individuata la entry corretta, se ne ricavano le informazioni necessarie per la consegna del contenuto della busta pervenuto al Servizio Informativo Locale destinatario.

Modulo di Consegna delle Buste SOAP.
Questo componente è implementato tramite un pool di thread che ha il compito di consegnare il contenuto applicativo delle buste SPCoop pervenute alla porta di dominio, ormai già sbustate dal modulo di Controllo, ai SIL destinatari interni al Dominio di Cooperazione. Il componente preleva le buste Soap dalla coda 'RequestIN_Soap', assieme alle ulteriori informazioni necessarie alla consegna (come ad esempio il tipo di protocollo da usare e la URL del SIL Applicativo su cui è attivo un server per la consegna). Utilizza le informazioni estratte per effettuare la consegna.

2.2.2 Componente di Integrazione SIL – to – PdD

Come è stato evidenziato nella sezione 1.4 di questa tesi, è stato scelto un approccio dinamico per il componente di integrazione SIL-to-PdD di OpenSPCoop PdD, che include un Proxy dinamico, capace di identificare il Servizio Destinatario utilizzando una **tabella di porte delegate/applicative** registrate sulla Porta di Dominio. Una porta delegata, che permetta ad un SIL interno al dominio l'invocazione di un servizio erogato in un altro dominio, è indirizzabile attraverso una semplice 'url' e contiene le informazioni di trasporto necessarie alla porta di dominio per spedire una busta e-Gov, che conterrà i dati XML pervenuti al momento dell'invocazione della porta delegata stessa. La porta delegata sarà quindi la chiave per l'interfaccia SIL-to-PdD, mentre sarà necessaria anche la definizione di una porta applicativa per risolvere l'interfacciamento PdD-to-SIL. La porta applicativa permette ad una porta di dominio che riceve una busta SPCoop l'individuazione del servizio richiesto dalla busta, che è erogato all'interno del dominio. La porta applicativa sarà indirizzabile attraverso informazioni contenute nella busta e-Gov, e conterrà informazioni sulla consegna del XML pervenuto (sbustato da informazioni di trasporto SPCoop), quali ad es. una url, una modalità di consegna (es. HTTP, SOAP….) ecc…
La Figura 2-3 evidenzia il contesto di porte delegate e applicative descritto.

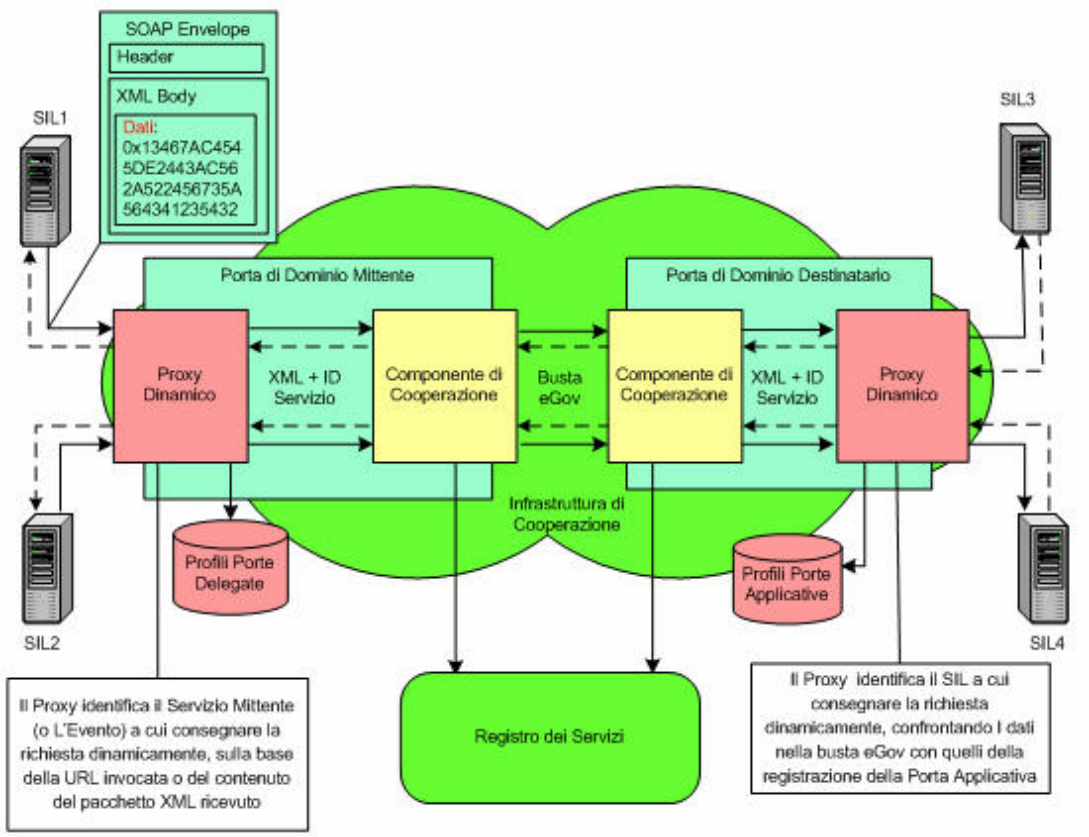


Figura 2-3, Scenario di cooperazione attraverso un proxy dinamico

Verranno adesso esaminati in dettaglio le porte delegate e le porte applicative necessarie nell'architettura OpenSPCoop PdD.

Porte Delegate.

Una porta delegata permette ad un SIL, facente parte del dominio di cooperazione gestito da una porta di dominio, di usufruire di un servizio erogato da un SIL risiedente in un altro dominio. Al momento della registrazione di una porta delegata in OpenSPCoop PdD, dovranno essere specificati i seguenti dati:

- ***URL.*** Rappresenta la url di invocazione della porta delegata utilizzata dai SIL. Ad esempio, supponendo che l'indirizzo di accesso alla porta di dominio sia configurato come 'http://www.openspcoop.org/pdd', il SIL per invocare una porta delegata registrata con l'url 'servizioDiEsempio' deve utilizzare l'indirizzo 'http://www.openspcoop.org/pdd/**servizioDiEsempio**'.
- ***Tipo*.** Indica il tipo di porta delegata tra tre tipi possibili (Static, Url-Based e Content-Based). Per ulteriori chiarimenti vedi il resto di questo paragrafo.

- ***Service Provider*.** Indica un Provider, appartenente ad un dominio di cooperazione di un'altra porta di dominio, che fornisce l'erogazione del servizio associato a questa porta delegata. Può essere anche il gestore degli eventi se il servizio richiesto è una pubblicazione o sottoscrizione evento.
- ***Servizio (e tipo).*** Indica l'identificativo del servizio fornito dal Service Provider.
- ***Azione***. Specifica il tipo di azione richiesta al servizio (può identificare anche un evento o una lista di eventi).
- ***SIL*-Mittente.** Indica i SIL che sono autorizzati ad invocare la porta Delegata.
- ***URL e Modalità Consegna.*** L'interazione per la consegna di una risposta applicativa tra SIL e Porta di Dominio, potrà essere attiva o passiva.

  Nel caso di **consegna della risposta passiva** dovrà essere specificata la URL del servizio esposto dal SIL a cui effettuare la consegna della risposta. L'url sarà utilizzato per la consegna di:
  - Risposte di tipo asincrone;
  - Sottoscrizione di eventi;
  - Risposte di tipo sincrono, in cui il SIL non sia riuscito a recuperare la risposta durante la stessa connessione http della richiesta (a causa di timeout o altri errori);

  In questo contesto, la ***ModalitàConsegna*** indica il tipo del servizio di consegna esposto dal SIL e può assumere i seguenti valori:
  - WSDL. In questo caso il servizio di consegna è un WebService, e quindi la URL permette di ottenere l'associato WSDL;
  - HTTP_POST. In questo caso il servizio di consegna è un HTTP Server, e la URL ne identifica l'indirizzo di accesso.
  - JMS. La consegna verrà effettuata su di una coda JMS.

  Nel caso, invece, di **consegna della risposta al SIL attiva**, al momento della registrazione non bisognerà fornire né una URLConsegna né una ModalitàConsegna. Sarà il SIL che dovrà recuperare le risposte applicative, usando uno speciale servizio offerto da OpenSPCoop PdD: 'servizio GetMessage'.

Il servizio 'getMessage', illustrato più avanti, dovrà essere utilizzato anche da un SIL che ha registrato una porta delegata con interazione passiva, se il servizio di consegna registrato diventa non più disponibile alla porta di dominio nel momento della consegna di una risposta applicativa (es. caduta momentaneamente del server HTTP).

Nella Figura 2-4 viene illustrato un esempio di invocazione di un servizio, da parte di un SIL. Il SIL1 vuole effettuare una registrazione di una nascita. In questo esempio, SIL1 ha precedentemente configurato la porta delegata nella porta di dominio, fornendo i dati del ServiceProvider, del servizio e dell'azione richiesta al momento della registrazione. L'accesso alla porta delegata, al momento dell'invocazione del servizio da parte del SIL, fornisce esattamente le tre informazioni (ServiceProvider,Servizio,Azione) necessarie per la creazione della busta e-Gov.

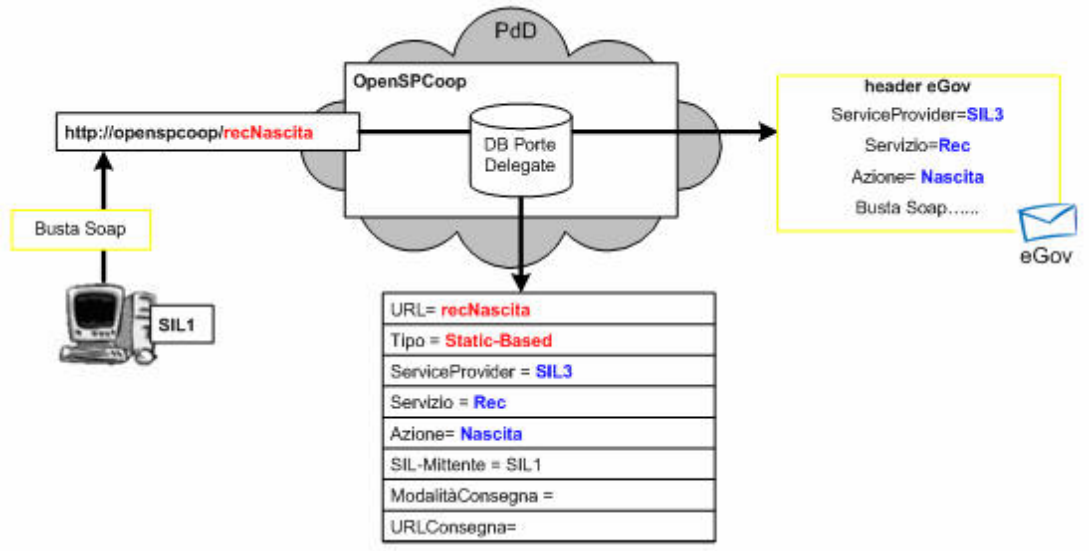


Figura 2-4, Scenario di utilizzo di una Porta Delegata di tipo 'Static-Based'

Un Servizio erogato da un sistema informativo locale di un dominio viene identificato, attraverso la tripla (ServiceProvider, Servizio e tipo, Azione). Quindi, come si è ampiamente descritto precedentemente, ad una porta delegata, al momento della sua creazione, deve essere fornita questa 'tripla' per poter fungere da collegamento al servizio erogato in un altro dominio. Tuttavia se la definizione di queste informazioni fosse possibile solo al momento della registrazione di una porta delegata, dovrebbero esistere una porta delegata per ogni servizio invocabile, e quindi questo approccio presenterebbe una forte limitazione di flessibilità.

Per ovviare a questo svantaggio, le informazioni possono anche non essere definite staticamente al momento della creazione della porta delegata, ma invece captate dinamicamente nel momento in cui un SIL effettua l'invocazione di una porta delegata attraverso una URL. L'identificazione della tripla, nella porta di dominio OpenSPCoop, potrà quindi avvenire in una delle tre modalità elencate (il tipo di porta delegata specifica la modalità da utilizzare):

- ***Static-Based***. Tutte le informazioni sono specificate al momento della registrazione della porta delegata.
- ***URL-based*.** In questo caso l'identificazione di alcune informazioni avvengono attraverso una interpretazione della url originaria invocata dal client. Ad esempio, supponendo che un SIL invochi l'indirizzo 'http://www.openspcoop.org/pdd/protocollo/milano', la porta di dominio può ricavarne la url di accesso alla tabella delle porte delegate attraverso la parola chiave 'protocollo' (esisterà una porta delegata registrata con url 'protocollo') e potrà utilizzare come azione il valore specificato dopo 'protocollo', in questo esempio 'milano'. Quindi il valore dell'azione della busta SPCoop che sarà prodotta sarà identificata come il primo componente successivo del path di invocazione della porta delegata.
- ***Content-based*.** In questo caso l'identificazione del servizio richiesto potrà essere effettuata anche tramite il contenuto dell'XML, individuato tramite un pattern standard di Xpath.

Di seguito vediamo un esempio dei metodi di identificazione URL Based e Content Based.

Esempio 1, identificazione URL-Based. Il SIL1 vuole effettuare una registrazione di una nascita. In questo esempio, a differenza del precedente, il SIL aveva effettuato la registrazione della porta delegata, fornendo solo i dati del ServiceProvider e del servizio. Il tipo di azione viene identificata grazie all'url invocata.

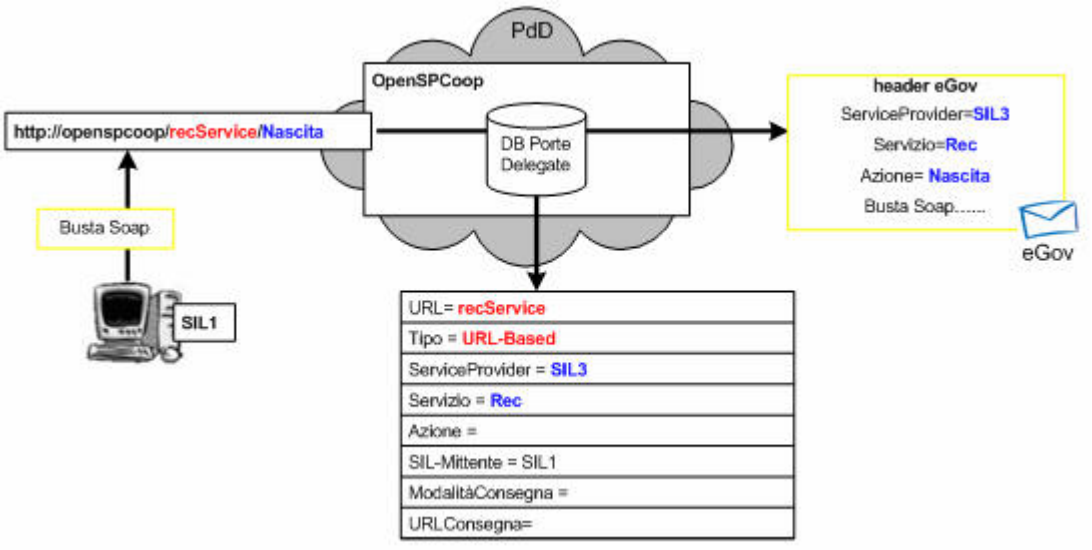


Figura 2-5, Scenario di utilizzo di una Porta Delegata di tipo 'URL-Based'

Esempio 2, identificazione Content Based. Il SIL1 vuole effettuare una invocazione di servizio per la registrazione di una nascita. In questo esempio, il SIL aveva effettuato la registrazione della porta delegata esattamente come nell'esempio precedente, fornendo solo i dati del ServiceProvider e del servizio. Il valore dell'azione viene, in questo contesto, identificato esaminando il contenuto XML della richiesta pervenuta alla porta di dominio.

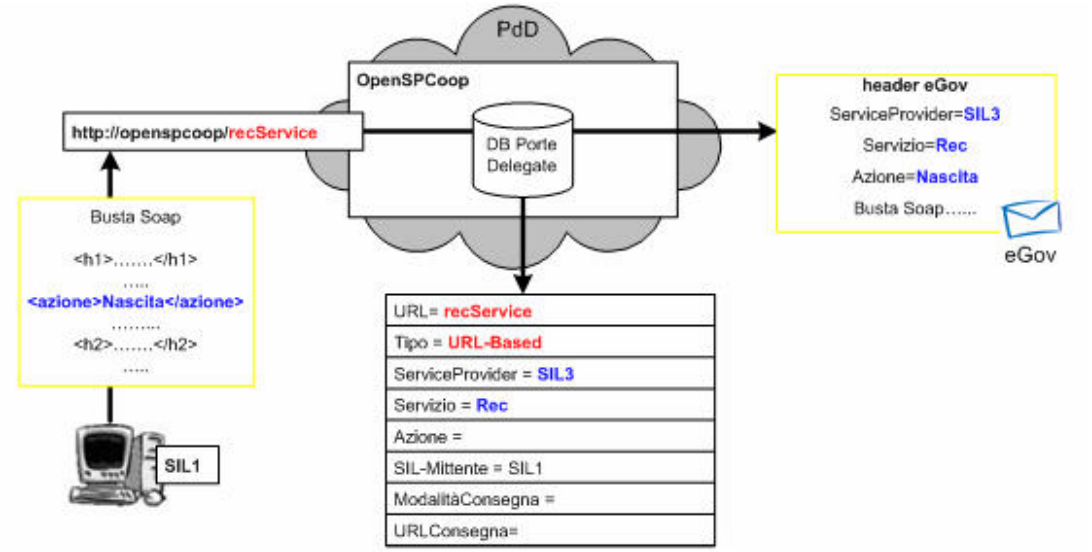


Figura 2-6, Scenario di utilizzo di una Porta Delegata di tipo 'Content-Based'

Il tipo di porta delegata registrata è quindi un parametro importante al momento della registrazione. Un altro parametro importante è la consegna di un eventuale risposta applicativa, pervenuta alla porta di dominio, in seguito alla richiesta effettuata.

E' stata precedentemente specificata la possibilità di impostare una url ed una modalità di consegna della risposta al momento della registrazione della porta delegata. In questo contesto, la gestione della risposta è **passiva**, poiché è la porta di dominio che deve effettuare una connessione verso un server di consegna, i cui parametri sono stati forniti nella registrazione della porta delegata. Se invece non viene fornita alcuna url di consegna, siamo in un contesto di gestione della consegna **attiva** dove la risposta applicativa viene ritornata direttamente sulla connessione precedentemente effettuata dal SIL per intraprendere una richiesta. Nel caso la connessione cada prima della ricezione della risposta applicativa (ad es. il protocollo HTTP possiede un timeout sulla connessione), sarà compito del SIL di verificare ad intervalli personali, l'arrivo della risposta relativa alla richiesta effettuata, utilizzando uno speciale servizio offerto da OpenSPCoop PdD.

La Figura 2-7 illustra un esempio di **interazione SIL-to-PdD passiva.** Nella definizione della porta delegata, viene specificato dove consegnare un risultato applicativo. Nell'esempio il SIL1 che invoca un servizio dispone di un WebService risiedente all'indirizzo **http://sil1/ricezione**, invocabile dalla porta di dominio per restituire il risultato applicativo.

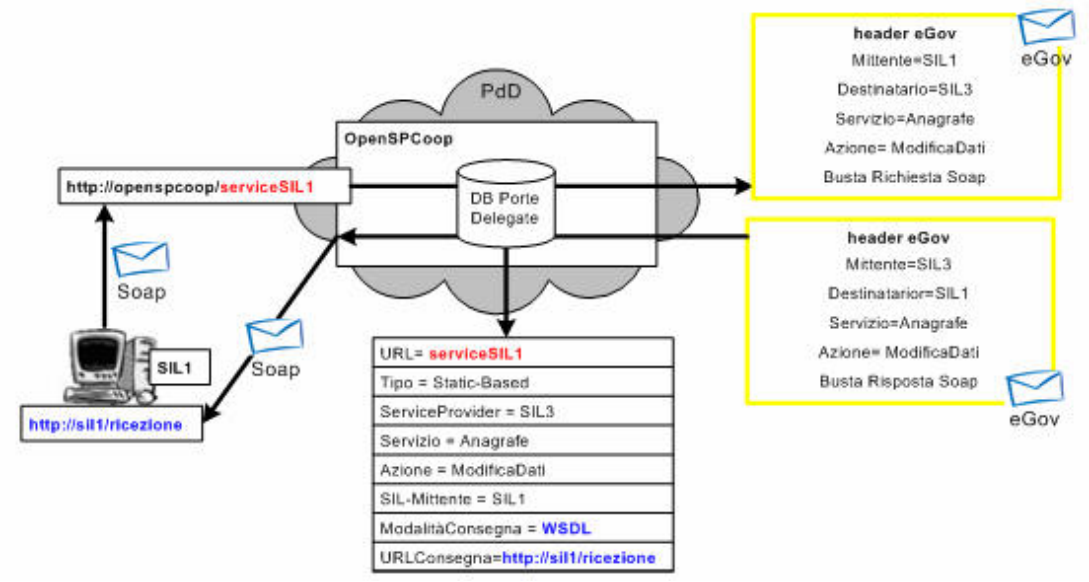


Figura 2-7, Scenario di utilizzo di una Porta Delegata con consegna passiva della risposta

Nel caso, invece, in cui **l'interazione con il SIL sia attiva** (o in caso di caduta momentanea del server di consegna definito nell'interazione passiva), come è stato precedentemente specificato, al momento della registrazione non viene fornita né un URLConsegna né una ModalitàConsegna. Nel caso in cui non è stato possibile inviare sulla connessione, effettuata dal SIL, la risposta applicativa, sarà il SIL che dovrà recuperare le buste arrivate, usando uno speciale servizio '**getMessage**' offerto da OpenSPCoop PdD.

Quindi nel caso di interazione attiva, o nel caso di caduta momentanea del server di consegna, l'architettura di OpenSPCoop PdD, per non perdere le risposte applicative ricevute, le memorizza in una coda di Risposte associandogli l'identificativo del SIL a cui è destinata la risposta. Per scoprire il SIL destinatario della risposta, viene acceduta la porta delegata (utilizzata dal SIL precedentemente per effettuare la richiesta) e viene prelevata l'informazione SIL-Mittente presente. Insieme all'identificativo del SIL Mittente, alla risposta Soap viene associato anche il servizio e l'azione corrispondente. A questo punto, il SIL può ottenere la risposta applicativa utilizzando lo speciale servizio 'getMessage' che viene implementato sia come un Web Service, che come un server HTTP, per garantire l'interoperabilità con un ampio raggio di tecnologie presenti sul SIL. Il Servizio è richiamabile con le seguenti modalità:

getMessage( )
getMessage(servizio,azione)

Il primo metodo permette di effettuare un Polling normale, mentre il secondo permette di effettuare un polling selettivo, filtrando le risposte ottenibili attraverso il servizio e l'azione. E' interessante notare che ad ogni risposta applicativa inserita nella coda risposte, viene associato l'identificativo del SIL che aveva richiesto il servizio. Questa informazione viene utilizzata da OpenSPCoop per restituire ai SIL che usufruiscono dei speciali servizi getMessage, solo le risposte che sono autorizzati a ricevere.

La Figura 2-8 illustra un esempio di interazione attiva, dove il SIL 1, ha associata una porta delegata, in cui non è stata definita una url di consegna.

Quando il SIL1 effettua una richiesta di servizio, supponendo che la connessione cada prima dell'arrivo della risposta applicativa, la risposta sarà memorizzata nella coda risposte, con le informazioni associate sul mittente (SIL1), il servizio e l'azione richiesta. La Figura 2-9 descrive invece l'utilizzo del servizio getMessage da parte del SIL1 per richiedere la risposta applicativa.

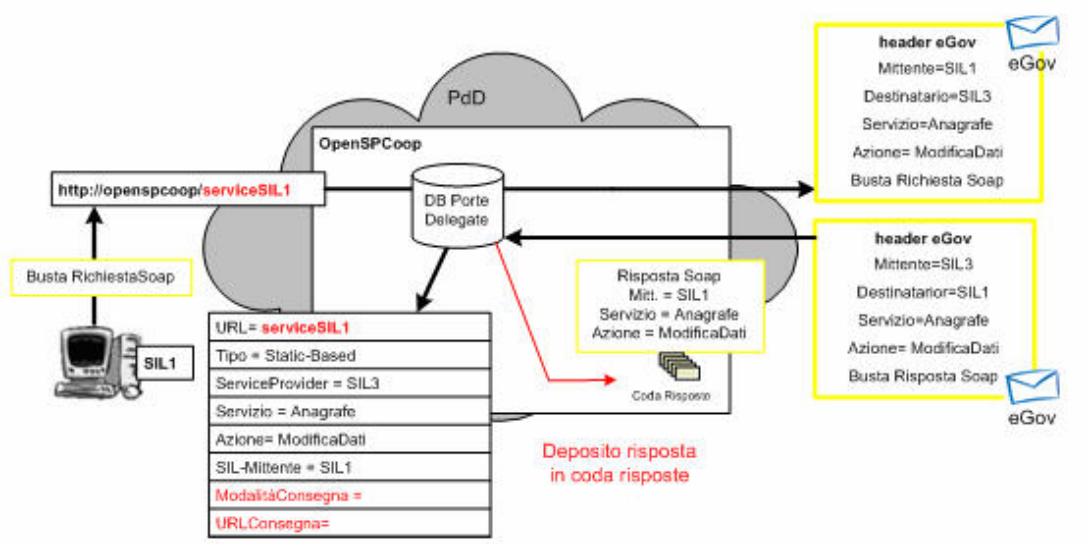


Figura 2-8, Porta Delegata con consegna attiva della risposta: la risposta pervenuta viene depositata in una coda apposita e gli vengono associate delle informazioni di contesto.

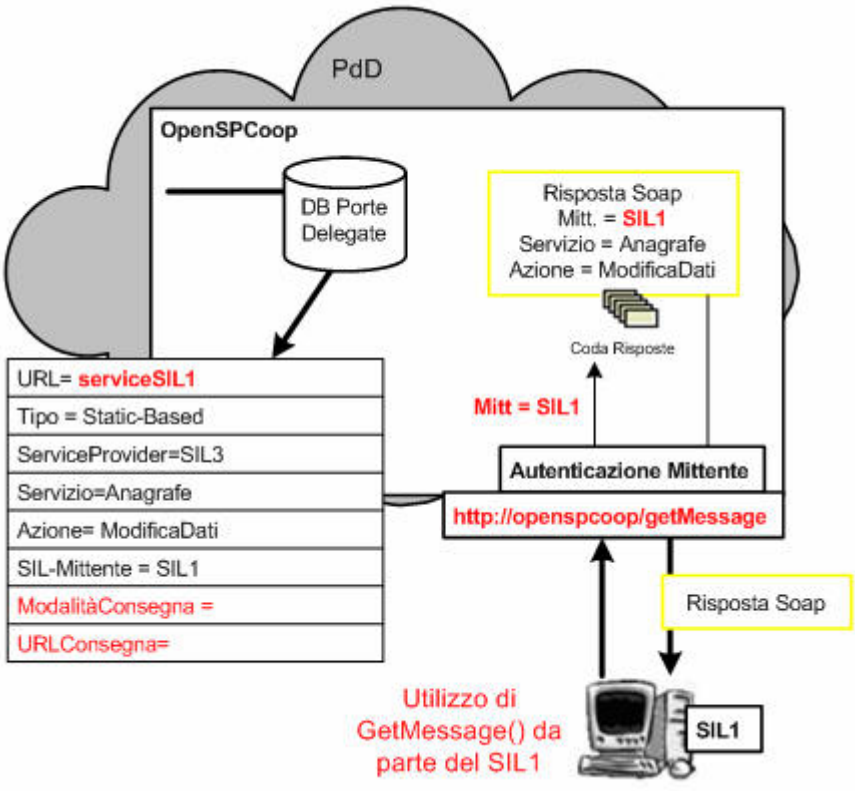


Figura 2-9, Recupero di una risposta applicativa, attraverso il servizio 'GetMessage', da parte di un Sistema Informativo Locale

Porte Applicative

Una porta applicativa permette ad una porta di dominio, che ha ricevuto una busta SPCoop, di identificare il SIL, interno al dominio, erogatore del servizio richiesto nella busta. Qualunque servizio applicativo, fornito da un SIL interno di un dominio, deve quindi essere registrato nella tabella dei **Profili delle Porte Applicative**, all'interno della porta di dominio OpenSPCoop. Le informazioni necessarie al momento della creazione di una porta applicativa, sono le seguenti:

- ***Service Provider.*** Indica un Provider, appartenente al dominio di cooperazione della porta di dominio, che fornisce l'erogazione del servizio associato a questa porta applicativa.
- ***Servizio (e tipo).*** Indica l'identificativo del servizio fornito dal Service Provider.
- ***Azione***. Specifica il tipo di azione richiesta al servizio.
- ***ModalitàConsegna.*** Specifica il tipo di protocollo utilizzato per la consegna del contenuto applicativo interno alla busta SPCoop pervenuta alla porta di dominio. Possibili protocolli sono WebService, http POST, JMS ecc…
- ***URLConsegna.*** Specifica la url del servizio applicativo a cui consegnare il contenuto della busta SPCoop pervenuta alla porta di dominio.

La Figura 2-10 illustra un esempio di porta applicativa.

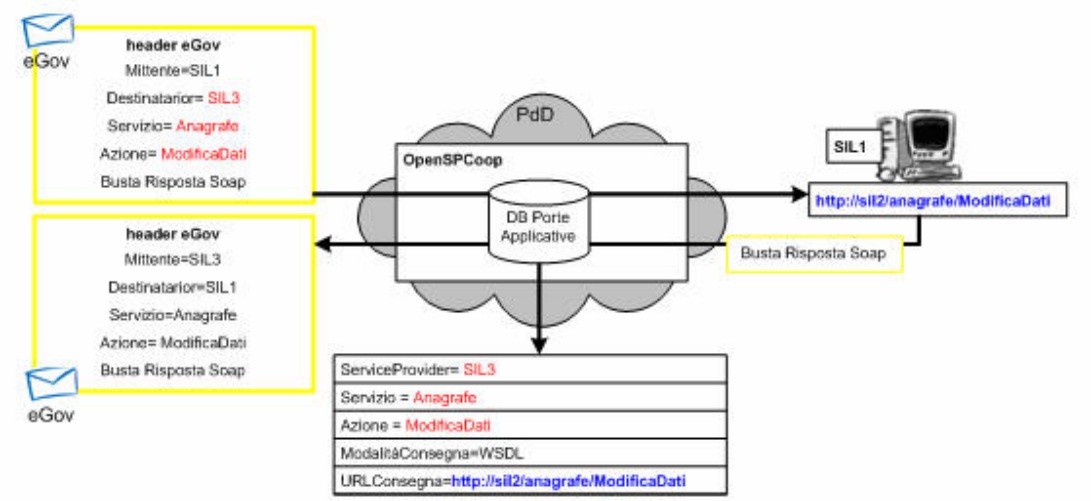


Figura 2-10, Scenario di utilizzo di una Porta Applicativa

2.2.3 Il Gestore degli Eventi

OpenSPCoop dovrà disporre di un servizio applicativo in grado di implementare il modello di cooperazione per eventi, che prevede lo scambio di messaggi al fine di comunicare il verificarsi di uno specifico evento. La Figura 2-11 illustra lo scenario, che utilizza il modello P&S (Publish & Subscribe) all'interno dell'architettura OpenSPCoop. In questo modello cooperano domini che pubblicano eventi e domini che sottoscrivono eventi. Gli attori di questo sistema sono: i domini pubblicanti, quelli sottoscrittori ed il gestore del servizio degli eventi (Event Manager), residente in un dominio apposito.

L'interazione di una porta di dominio, mediatrice di un pubblicatore o sottoscrittore, avviene con il gestore degli eventi tramite lo scambio di buste e-Gov con la porta di dominio posta davanti al gestore e quindi usufruendo di tutti i servizi offerti dalla specifica e-Gov. In pratica il gestore è possibile vederlo come un servizio che si occupa di gestire le comunicazioni di evento tra i soggetti interessati (sistemi informativi locali). Il Gestore Eventi è strutturato quindi come una porta applicativa appartenente ad una porta di dominio speciale, con il compito di ricevere e consegnare messaggi contenenti eventi, incorporati in buste SPCoop (**pubblicazione/sottoscrizione indiretta**).

Le porta di dominio di un pubblicatore/sottoscrittore, oltre che dialogare con la porta di dominio di un gestore degli eventi, potrà accedere direttamente alle code (o topic) interne al gestore, come illustrato dalle frecce rosse della Figura 2-11, ottenendo una maggiore efficienza di prestazioni (**pubblicazione/sottoscrizione diretta**). Il guadagno di prestazioni, rispetto alla gestione indiretta, lo si ha poiché non avviene la gestione del protocollo e-Gov tra le due porte di dominio e quindi non vengono gestite le varie funzionalità di consegna affidabile, eliminazione dei duplicati, ecc…

Vediamo nel dettaglio dell'architettura OpenSPCoop PdD, descritta nei paragrafi 2.2.1 e 2.2.2, il funzionamento in OpenSPCoop della gestione diretta di pubblicazione / sottoscrizione. La gestione indiretta può essere vista come un invocazione di un particolare servizio, e quindi il funzionamento è re-indirizzabile alla gestione di una porta applicativa illustrato nel paragrafo 2.2.2.

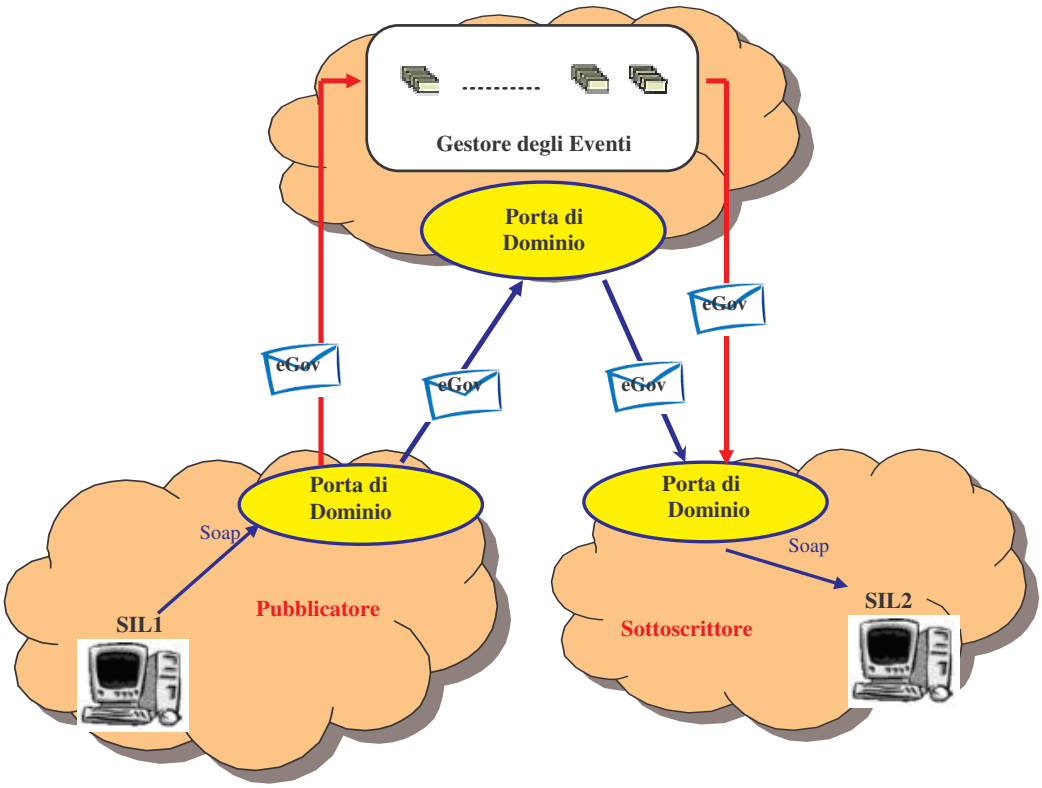


Figura 2-11, Scenario di una cooperazione applicativa ad eventi in OpenSPCoop

Pubblicatore

Un SIL produttore di eventi, per effettuare la pubblicazione di un evento deve avere precedentemente creato una porta delegata apposita nella sua porta di dominio. L'evento che dovrà essere pubblicato può essere specificato direttamente al momento della creazione della porta delegata (Static-Based), oppure può essere ottenuto al momento dell'invocazione della porta delegata da parte del SIL, attraverso l'analisi dell'url invocato (URL-Based) o attraverso l'analisi della richiesta XML (Content-Based), come illustrato nel paragrafo 2.2.2.

La Figura 2-12 illustra le azioni che sono intraprese all'interno dell'architettura OpenSPCoop, dopo che il produttore invoca il suo servizio di pubblicazione

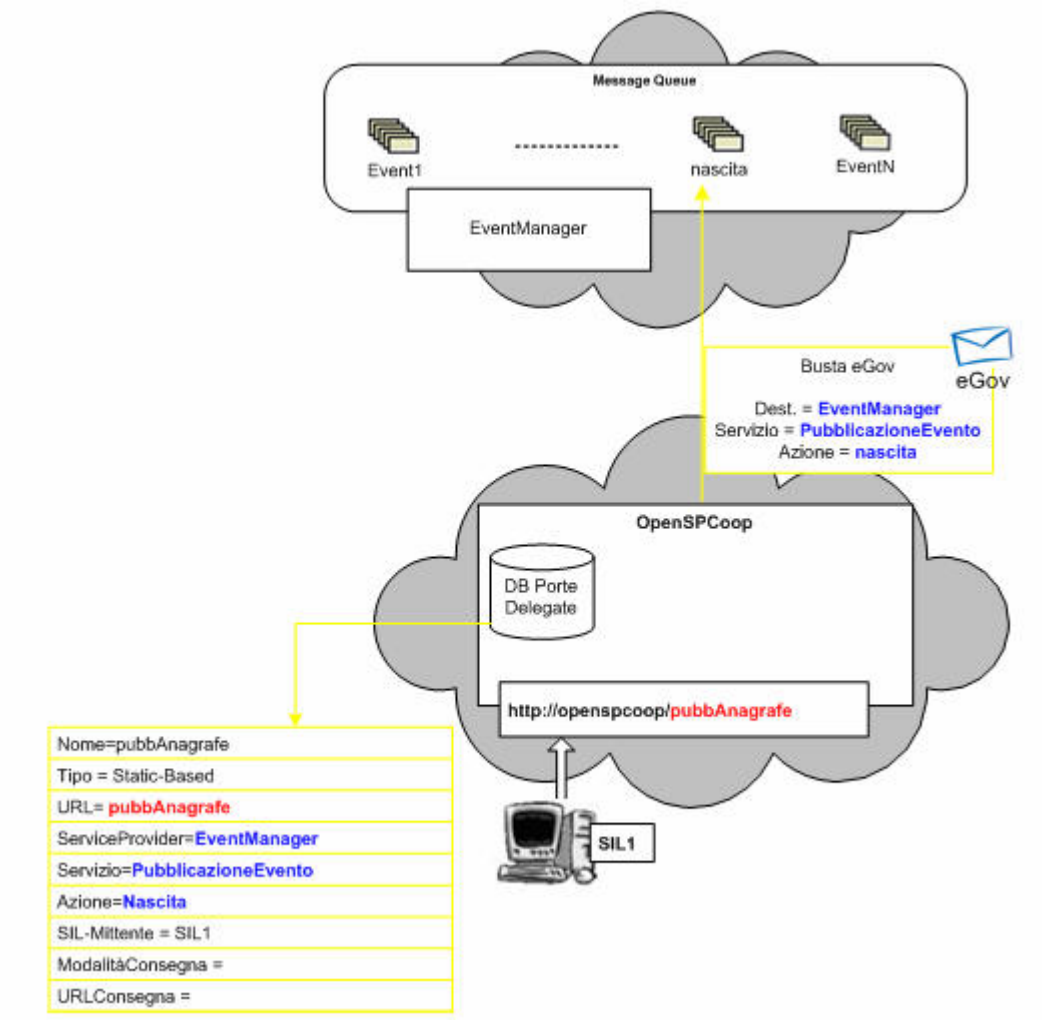


Figura 2-12, Scenario di utilizzo di una porta delegata registrata per la pubblicazione di un evento

Sottoscrittore

Un consumatore che è interessato alla ricezione di un evento, deve effettuare la sottoscrizione all'evento. Il sottoscrittore ha a sua disposizione diverse soluzioni:

- Interrogare la sua porta di dominio circa l'arrivo di nuovi eventi a lui destinati (polling);
- Interrogare la sua porta di dominio circa l'arrivo di determinati tipi di eventi (polling selettivo);
- Segnalare alla sua porta di dominio la propria disponibilità a ricevere gli eventi a lui destinati, fornendo una url dove consegnare gli eventi (ricezione passiva).

La sottoscrizione include quindi i seguenti requisiti:

- ***Evento (Lista di eventi).*** Rappresenta uno specifico evento (o più di uno) presente nell'Event Manager.
- ***Mittente (anche più di uno).*** Rappresenta l'identificativo del consumatore (o più di uno) interessato a ricevere l'evento.
- ***Modalità e URLConsegna.*** Indica una url di un server di consegna ed il protocollo da utilizzare per recapitare un evento.

Un SIL, all'interno dell'architettura di OpenSPCoop PdD, per ottenere una sottoscrizione ad un evento deve registrare una porta delegata apposita, nella sua porta di dominio, che conterrà le informazioni sulla ricezione dell'evento (ModalitàConsegna e URLConsegna), il serviceProvider (l'EventManager), il servizio (SottoscrizioneEvento), l'azione (evento o lista di eventi) e il SIL-Mittente (identificativo/i del consumatore interessato all'evento).

Vediamo un esempio di creazione di porte delegate. La Figura 2-13 illustra un contesto in cui un SIL1 crea una porta Delegata apposita per la sottoscrizione agli eventi 'Nascita' e 'Morte'. L'esempio mostra anche la creazione di un porta delegata, che specifica la ricezione di un evento (nascita) destinato a più SIL, (l'evento sarà ricevuto dal SIL2 e dal SIL3), i quali non forniscono un server di consegna.

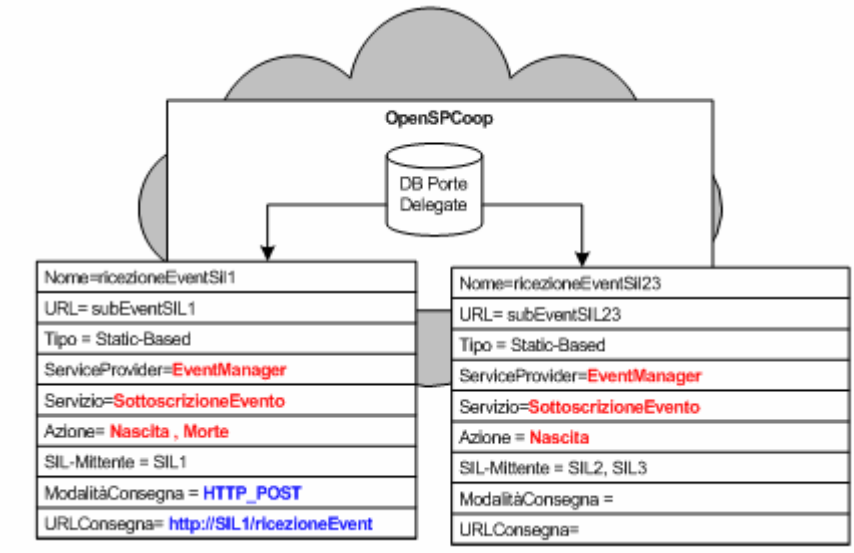


Figura 2-13, Scenario di utilizzo di porte delegate che realizzano una sottoscrizione ad eventi

La ricezione di eventi, attraverso OpenSPCoop PdD, comporta la creazione di un Listener che si occupa di ricevere tutti i possibili eventi che interessano ai sottoscrittori che si sono registrati all'interno della porta di dominio.

Una volta ricevuto un certo tipo di evento, vengono utilizzate le informazioni e-Gov presenti nell'evento per accedere alla tabella delle porte delegate e per ottenere l'url di consegna dell'evento e la modalità di consegna, se forniti al momento della creazione della porta delegata da parte del SIL. A questo punto, se è presente una url, viene effettuata la consegna dell'evento utilizzandola. La consegna può andare a buon fine, se sull'url è attivo un servizio di ricezione. Questo caso viene evidenziato nella Figura 2-14.

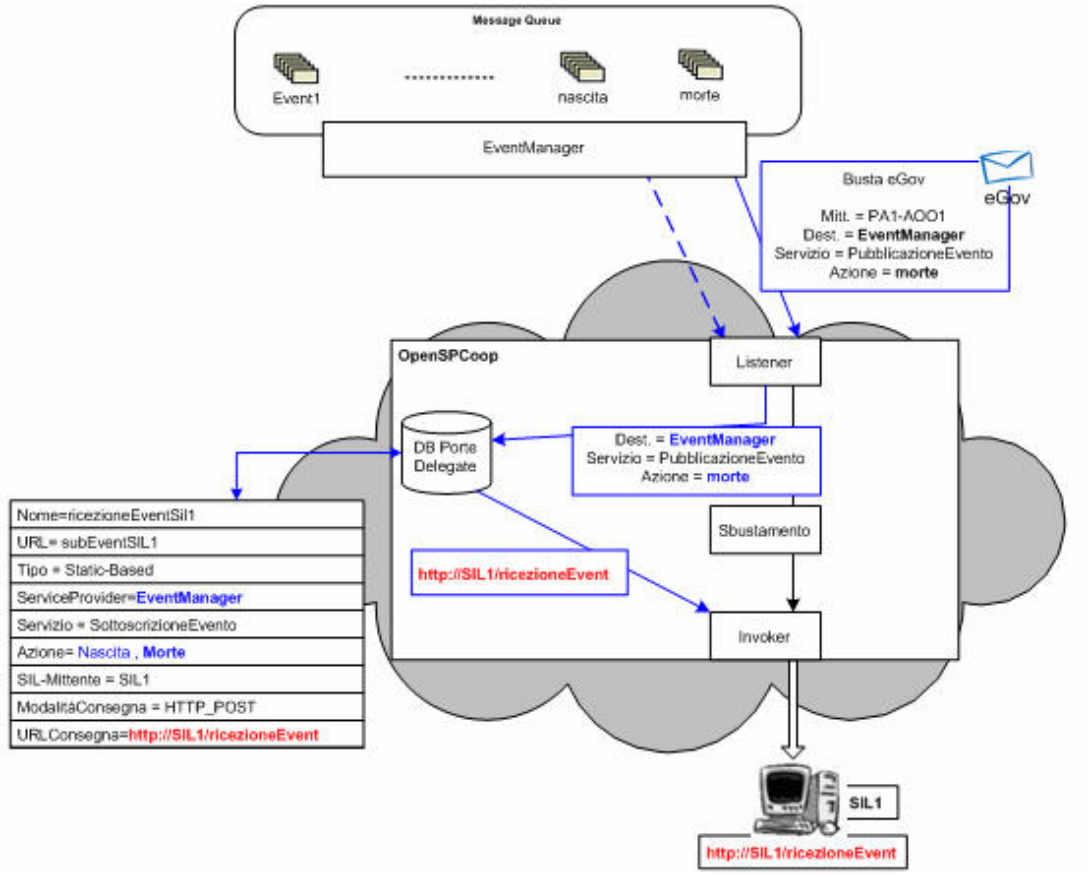


Figura 2-14, Scenario di utilizzo di una porta delegata che permette la sottoscrizione ad un evento con modalità di consegna passiva

La consegna direttamente attraverso il server di consegna, può non andare a buon fine, poiché è possibile che sul SIL1 sia non attivo momentaneamente il servizio di ricezione. Inoltre, il SIL1, durante la creazione della porta delegata, può non avere fornito un url di consegna, visto che intendeva ricevere gli eventi solo dietro un polling personalizzato.

In entrambi i casi, l'architettura di OpenSPCoop, per non perdere gli eventi ricevuti li memorizza in una coda di Risposte. Vengono utilizzate le informazioni e-Gov presenti nell'evento arrivato per accedere alla tabella delle porte delegate e ottenere i SIL che si sono sottoscritti alla ricezione dell'evento (avendo creato una porta delegata apposita). A questo punto, viene depositato nella coda di Risposte una replica dell'evento per ogni iscritto, associandogli le informazioni Mittente (SIL iscritto), servizio (SottoscrizioneEvento) e azione (evento). Sarà poi il consumatore che invocherà lo speciale servizio 'getMessage ' offerto da OpenSPCoop e descritto nel paragrafo 2.2.2.

La Figura 2-15, illustra il caso in cui la consegna diretta dell'evento arrivato, tramite server di consegna, non può essere effettuata, poiché il servizio di ricezione sul SIL non è attivo. Quindi l'evento viene inserito nella coda Risposte (freccia rossa).

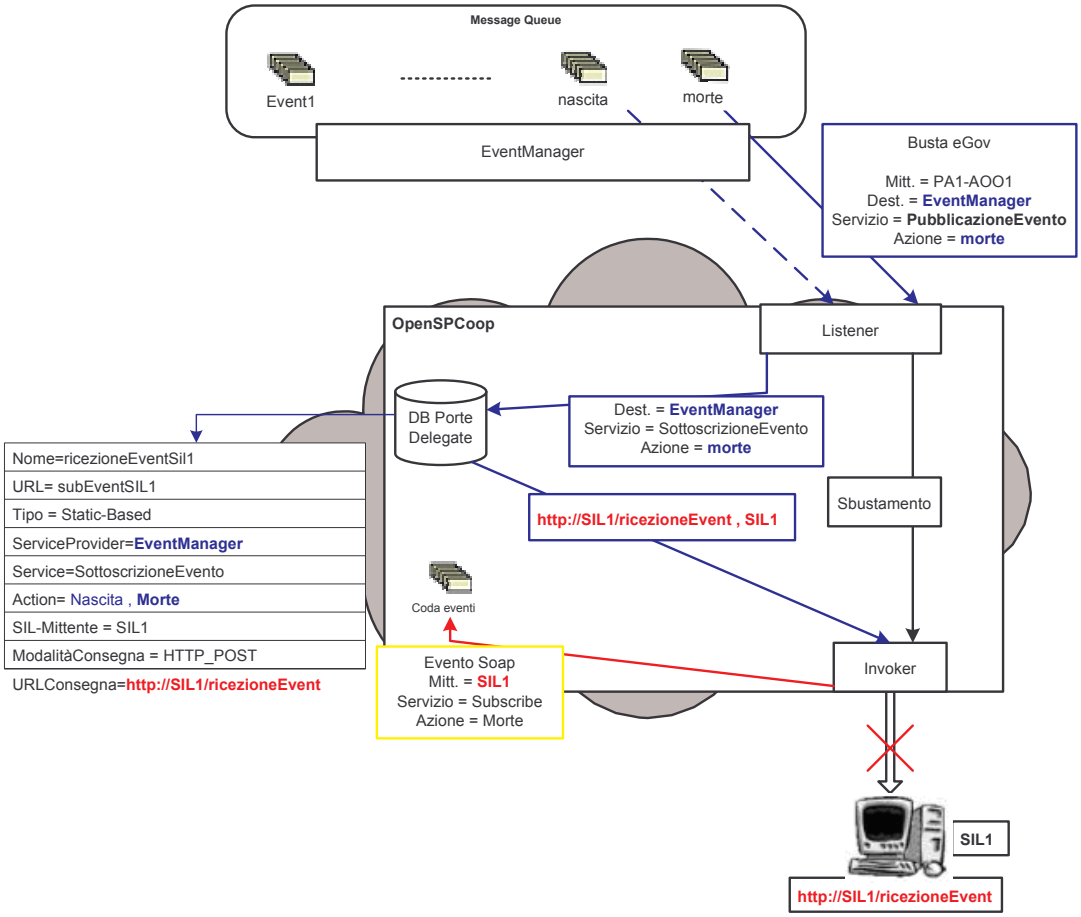


Figura 2-15, Contesto esemplificativo dove sussiste la ricezione di un evento associabile ad una porta delegata configurata con modalità di consegna passiva, in cui il server di consegna è momentaneamente non disponibile.

Lo speciale servizio getMessage fornito da OpenSPCoop, come è già stato illustrato nel paragrafo 2.2.2, è richiamabile in una tra queste due seguenti firme:

getMessage( )
getMessage(Servizio, Azione)

Il primo metodo permette di effettuare un Polling normale, mentre il secondo permette di effettuare un polling selettivo. In pratica, utilizzando la seconda modalità di utilizzo, il servizio, deve essere 'Subscribe', mentre l'azione indica il tipo di evento che il consumatore è interessato a ricevere (es. nascita). E' interessante notare che ad ogni evento inserito in coda risposte, viene associato l'identificativo del consumatore che può prelevare l'evento (SIL-Mittente). Questa informazione viene utilizzata da OpenSPCoop per restituire a chi usufruisce dei speciali servizi getMessage, solo gli eventi che sono autorizzati a ricevere. Saranno presenti degli eventi, solo se si il sottoscrittore si era precedentemente iscritto alla ricezione dell'evento.
Nella Figura 2-16 viene illustrato il caso del Polling effettuato dal consumatore dell'evento (es. SIL1, frecce blu), dopo che un evento 'morte' per il SIL 1 è stato depositato nella codaEventi.

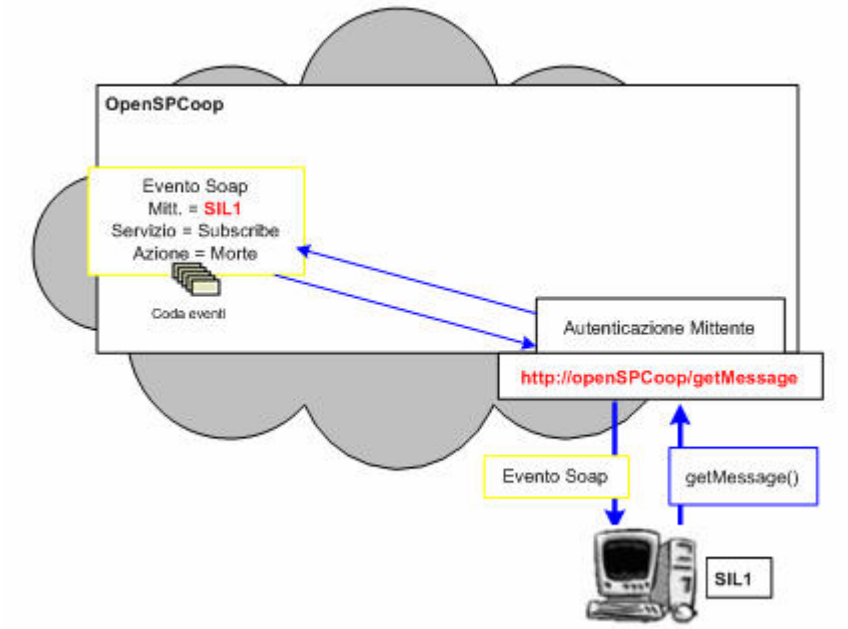


Figura 2-16, Scenario di utilizzo del servizio 'GetMessage' per la ricezione di eventi

2.2.4 Il cuore SPCoop della porta di dominio: Modulo di Controllo

Nell'architettura di OpenSPCoop, il modulo che si occupa di gestire lo scambio di una busta SPCoop tra due porte di dominio SPCoop è il Modulo di Controllo. Si occupa di implementare tutte le funzionalità richieste dalla specifica CNIPA, riguardante la busta e-Gov. Il modulo di controllo è il cuore dell'architettura di OpenSPCoop PdD, e risiede al centro di essa. Ogni singola interazione attivabile in un qualsiasi punto di ingresso dell'architettura, provocherà un pipeline di processamento, tra i vari moduli di OpenSPCoop, in cui uno stadio sarà sempre e comunque il modulo di controllo. Le funzionalità del modulo sono suddivisibili in due sottocomponenti:

- Il **modulo 'Imbustamento'**. Viene utilizzato, quando un sistema informativo locale del dominio di cooperazione richiede un servizio attraverso l'invocazione di una porta delegata. Il modulo dovrà creare una apposita busta SPCoop da spedire alla porta di dominio destinataria che gestisce il SIL che eroga il servizio richiesto.
- Il **modulo 'Sbustamento'.** Viene utilizzato, quando OpenSPCoop PdD riceve una busta SPCoop da un'altra porta di dominio. La busta sarà innanzitutto validata semanticamente e sintatticamente. Dopodichè verrà utilizzata per identificare una porta applicativa da utilizzare per la consegna del contenuto applicativo risiedente nella busta e-Gov pervenuta.

La Figura 2-17 illustra la suddivisione in 'modulo Imbustamento' e 'modulo Sbustamento' del modulo di controllo evidenziandone le funzionalità e-Gov gestite.
I paragrafi sottostanti entrano nel dettaglio di ogni singolo componente, il cui compito è gestire una funzionalità SPCoop associabile ad un determinato servizio. Le informazioni delle funzionalità e-Gov associate ad un servizio applicativo, come è stato descritto brevemente nel paragrafo 1.3.3, sono memorizzate nell'accordo di servizio all'interno del registro dei servizi.

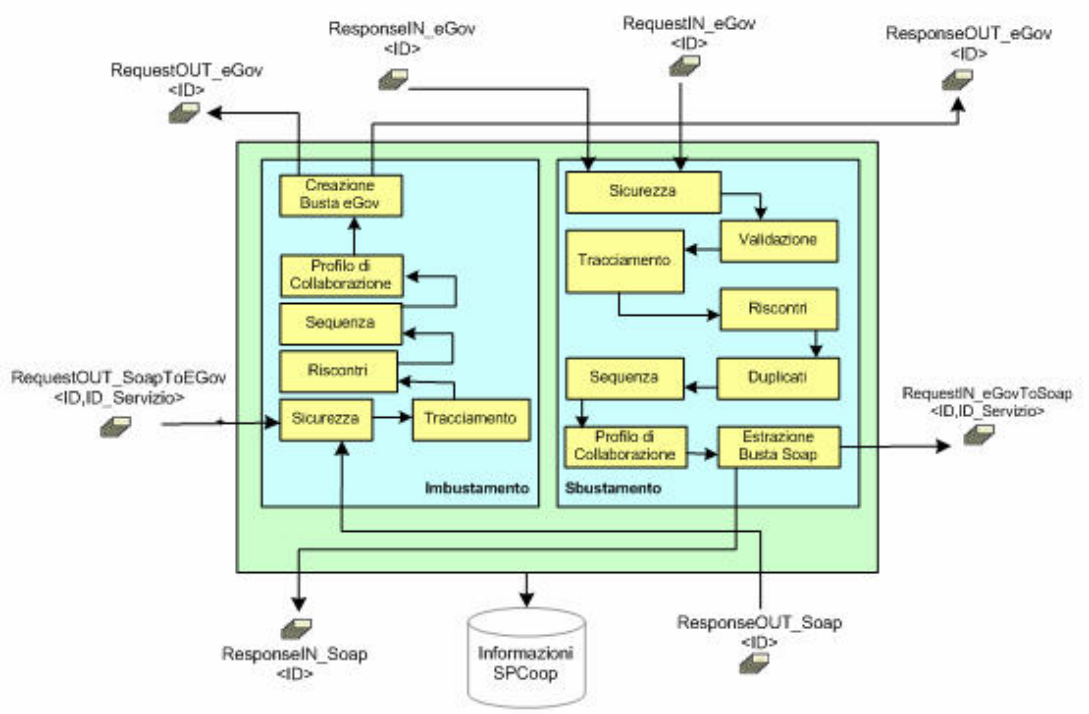


Figura 2-17, Architettura del modulo di Controllo

2.2.4.1 Sicurezza

Questa funzionalità è presente sia nella parte 'Imbustamento' che nella parte 'Sbustamento' del modulo di controllo. I compiti di questo modulo includono un controllo sul mittente del messaggio pervenuto al modulo, esaminando se il mittente abbia i diritti e l'autorizzazione per invocare un servizio. Si occupa, quindi, di gestire tutti gli aspetti riguardanti la sicurezza, come autorizzazione all'invocazione di un servizio, autenticazione di un sistema informativo locale, ecc. Le informazioni sui mittenti autorizzati ad accedere ad un determinato servizio, ed altre informazioni di contesto, sono prelevate dal registro dei servizi LDAP (vedi paragrafo 2.2.5).

Gestire il complesso mondo della sicurezza non fa parte del lavoro di tesi e quindi non verranno proposte nel restante capitolo, e nei capitoli successivi, soluzioni progettuali o implementative che risolvano questo determinato compito.

2.2.4.2 Validazione busta SPCoop

Questa funzionalità viene utilizzata dalla parte 'Sbustamento' del modulo di controllo, poiché solo essa riceve buste in formato e-Gov. Viene effettuato un controllo di conformità del messaggio con lo standard busta SPCoop e verificato che il contenuto del messaggio sia ben formato e sintatticamente corretto. Ogni qualvolta che la parte 'Sbustamento' riceve un messaggio e-Gov, viene effettuato prima di ogni altra cosa la Validazione del messaggio, come è definito nella standard CNIPA, in cui è specificato che il destinatario del messaggio non è autorizzato a intraprendere alcuna operazione di trattamento, prima di aver completato la verifica di formato, sintattica e semantica.

I controlli sono eseguiti sull'header della busta e-Gov e su alcuni campi della busta Soap. I requisiti sull'header SOAP sono minimi, e riguardano un utilizzo di namespaces corretti, di una proprietà 'actor' correttamente impostata ad un valore fornito dal CNIPA e dell'attributo mustUnderstand che come illustrato da specifica, deve essere impostato al valore 'true'. Dopodichè sono controllati ogni singola parte dell'header E-Gov:

- ***Mittente e Destinatario*** *(obbligatorio).* Deve essere obbligatoriamente presente e riconosciuto dal registro dei servizi poiché utile per la risoluzione nell'indirizzo fisico della corrispondente parte.
- ***Profilo di Collaborazione*** *(opzionale).* Se presente deve essere un profilo riconosciuto dal registro dei servizi, e se è presente la proprietà 'Tipo', riferita al tipo di 'ServizioCorrelato' (presente anch'essa come proprietà nei profili asincroni), anche il tipo deve essere un valore valido.
- ***Collaborazione e Sequenza*** *(opzionale).* La collaborazione deve contenere l'identificativo del messaggio capostipite che ha originato i successivi messaggi e deve essere riconosciuta dalla porta di dominio destinataria.

  La presenza della sequenza, oltre alla presenza della collaborazione comporta anche la presenza di un profilo di Trasmissione correttamente impostato a valori specifici; l'attributo 'Inoltro' deve assumere il valore EGOV_IT_ALPIUUNAVOLTA e l'attributo 'ConfermaRicezione' deve assumere il valore 'true'.

- ***Servizio e Azione*** *(opzionale).* Entrambi devono possedere valori validi riconosciuti dal registro dei servizi.
- ***Messaggio*** *(obbligatorio).* Negli elementi obbligatori, l'identificatore del messaggio deve avere una struttura valida e ben definita, e l'ora di registrazione deve contenere Data e Orario di creazione dell'header e-Gov in un formato valido. Il RiferimentoMessaggio (opzionale), se presente, deve essere valido rispetto allo stato del profilo di collaborazione in corso, e la Scadenza (opzionale), deve indicare il periodo di validità del messaggio attraverso una Data valida.
- ***Profilo Trasmissione*** *(opzionale).* I valori degli attributi 'Inoltro' e 'ConfermaRicezione' di questo profilo devono essere valori validi.
- ***Lista Riscontri e Trasmissioni*** *(opzionale).* La lista dei riscontri presenti deve essere controllata per verificare la validità sintattica e semantica di ogni riscontro.

2.2.4.3 Tracciamento

Questa funzionalità è presente sia nella parte 'Imbustamento' che nella parte 'Sbustamento' del modulo di controllo. Il tracciamento viene effettuato sia localmente, su files di log e Database, che direttamente all'interno di un header della busta di e-Gov. L'inserimento di informazioni, dentro l'header e-Gov, è compito solo della parte 'Imbustamento' del modulo di controllo, che si occupa di inserire nella lista trasmissioni le informazioni necessarie per determinare che tale busta è transitata attraverso la Porta di Dominio a cui appartiene il Modulo.
Le informazioni accluse sono:

- *Origine.* Contiene l'identificativo del mittente del messaggio.
- *Destinazione.* Contiene l'identificativo del destinatario del messaggio.
- *OraRegistrazione*. Rappresenta l'ora in cui è stata creata la traccia.

2.2.4.4 Gestione dei Riscontri

Questa sottoparte del modulo di controllo si occupa di implementare la gestione della spedizione di un acknowledgement riguardante una conferma di ricezione di una busta SPCoop, specificata attraverso l'attributo 'ConfermaRicezione' del campo profilo trasmissione della busta, che può assumere i valori 'True' (richiesta di ricevuta) o 'False'(nessuna richiesta di ricevuta, valore di default). Gli aspetti di gestione dei riscontri sono suddivisibili in due grandi componenti:

- **Gestione della notifica di riscontri verso altre porte di dominio,** effettuata dalla parte di Sbustamento. Se una busta e-Gov, processata da un modulo di controllo di una porta di dominio, possiede l'attributo impostato al valore 'true', il modulo di controllo dovrà inviare al mittente un messaggio apposito contenente nell'elemento lista riscontri, il riscontro relativo al messaggio ricevuto. Adottando questo approccio, si ha una inefficienza di banda, poiché per ogni busta e-Gov inviata (con confermaRicezione uguale a 'true') viene generato una ulteriore busta e-Gov per il riscontro.

  Esiste un diverso approccio per la gestione dei riscontri, oltre a quello precedentemente descritto, in cui si può risparmiare banda di trasmissione. Quando una busta con mittente A, richiede una conferma di ricezione, il riscontro non viene immediatamente spedito, ma viene inserito nella lista riscontri di un successivo messaggio da inviare al mittente A (modalità **piggy-backing**). Nel caso in cui ciò non fosse possibile entro un limite temporale (timeout predefinito), deve essere inviato un messaggio apposito contenente nella header il riscontro alla Porta di Dominio mittente A.
- **Gestione della ricezione di riscontri per la porta di dominio**, effettuata dalla parte di Imbustamento e Sbustamento. Quando l'attributo 'confermaRicezione' di una busta, appena costruita, assume il valore *true,* la Porta di Dominio Mittente deve gestire la persistenza dei messaggi per effettuarne il re-inoltro in caso di mancato ricevimento dell'*acknowledgment* generato dalla Porta di Dominio destinataria.

Verranno esaminate nel dettaglio progettuale entrambe le gestioni dei riscontri.

Gestione della notifica di riscontri verso altre porte di dominio.

Iniziamo a considerare la notifica di riscontri verso altri destinatari in OpenSPCoop PdD. Questa funzionalità interessa il componente 'Riscontri' della parte 'Sbustamento' del modulo di controllo.

La scelta della modalità non piggy-backing / piggy-backing è impostabile attraverso un file di configurazione. Nel caso in cui viene scelta la ***modalità non piggy-backing***, essa viene semplicemente realizzata dal componente 'riscontri' creando una apposita busta e-Gov di risposta.

Più interessante è la gestione con ***piggy-backing***. Per realizzarla è necessaria una tabella 'RiscontriDaSegnalare', dove ogni entry ha come chiave di accesso un destinatario, e come valore associato una lista di riscontri da inviare a quel destinatario. La tabella viene gestita dal componente 'Riscontri' della parte 'Sbustamento'. All'arrivo di un messaggio e-Gov, contenente la proprietà 'confermaRicezione = true', il componente inserisce una coppia di valori (IDMsg, ora) nella tabella inserendola nella lista presente nell'entry con chiave di accesso il mittente del messaggio e-Gov arrivato. Sarà poi il modulo 'Imbustamento' che nel momento in cui dovrà inviare ad un destinatario una busta e-Gov prelevata dalla sua coda, inserirà nella lista riscontri della busta creata tutti i riscontri presenti nell'entry della tabella RiscontriDaSegnalare, acceduta con chiave di accesso il destinatario del pacchetto. Dopodichè cancellerà l'intera entry.

Esisterà, inoltre, un **Thread di servizio** il cui compito sarà controllare il timeout associato ad ogni riscontro presente nella tabella RiscontriDaSegnalare. Quando un timeout è scaduto, il thread si occuperà, come da specifica CNIPA, di inviare un messaggio e-Gov apposito contenente nella header il riscontro alla Porta di Dominio associata all'entry nella tabella (modalità non-piggy-backing).

Le figure seguenti illustrano un esempio di questo scenario (**piggy-backing**).

*Step 1 (*Figura 2-18*): Ricezione di una busta e-Gov, con 'ConfermaRicezione = true' dal SIL2.* Non esistendo ancora una entry con chiave di accesso 'SIL2' nella tabella, il componente 'Sbustamento' crea una apposita entry, ed inserisce l'informazione del riscontro da inviare.

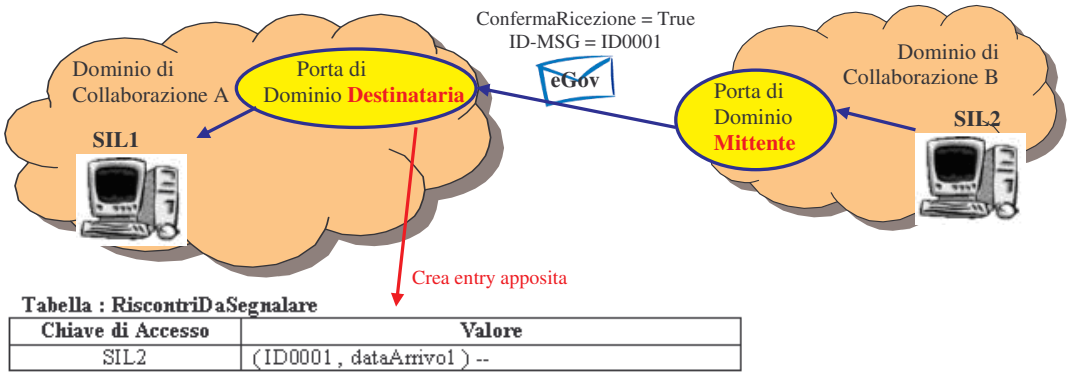


Figura 2-18, Ricezione prima busta SPCoop contenente una richiesta di riscontro

*Step 2 (*Figura 2-19*): Ricezione di una busta e-Gov, con 'ConfermaRicezione = true' dal SIL3.* Non esistendo ancora una entry con chiave di accesso 'SIL3' nella tabella, il componente 'Sbustamento' crea una apposita entry, ed inserisce l'informazione del riscontro da inviare.

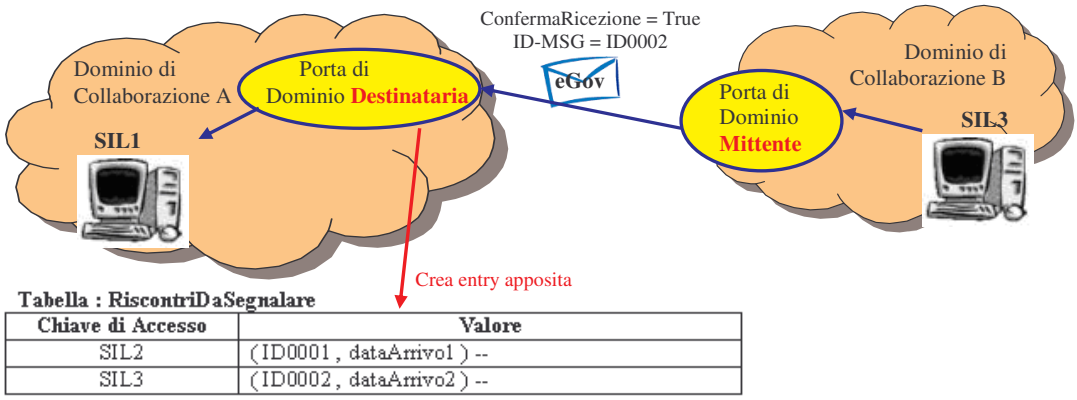


Figura 2-19, Ricezione seconda busta SPCoop contenente una richiesta di riscontro

*Step 3 (*Figura 2-20*): Ricezione di una busta e-Gov, con 'ConfermaRicezione = true' dal SIL2.* Siccome l'entry con chiave di accesso 'SIL2' esiste già nella tabella, il componente 'Sbustamento' appende nell'apposita entry, l'informazione di un ulteriore riscontro da inviare.

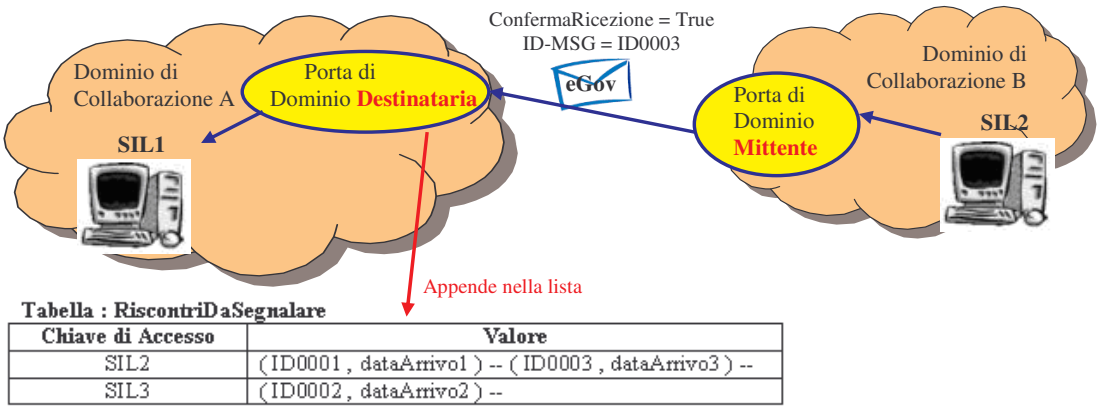


Tabella : RiscontriDaSegnalare

Chiave di Accesso	Valore
SIL2	(ID0001 , dataArrivo1) -- (ID0003 , dataArrivo3) --
SIL3	(ID0002 , dataArrivo2) --

Figura 2-20, Ricezione terza busta SPCoop contenente una richiesta di riscontro

*Step 4 (*Figura 2-21*): Invio di una busta e-Gov al SIL2.* Il modulo 'Imbustamento', durante la creazione della busta SPCoop cerca nella tabella 'RiscontriDaSegnalare', se vi è presente una entry con chiave di accesso 'SIL2'. Siccome l'entry nella tabella esiste, il modulo di OpenSPCoop appende nella lista Riscontri della busta e-Gov creata tutti i riscontri presenti nella riga della tabella acceduta. Dopodichè cancella l'entry associata con chiave di accesso 'SIL2'.

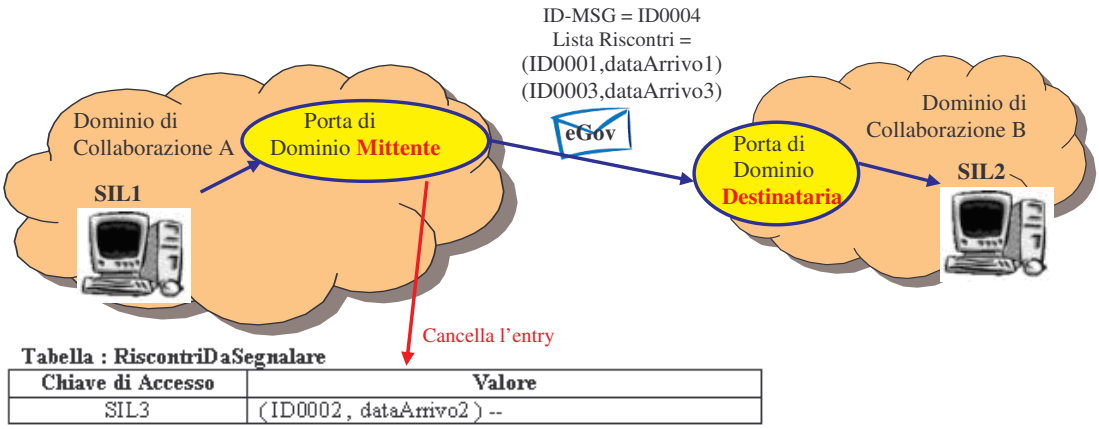


Tabella : RiscontriDaSegnalare

Chiave di Accesso	Valore
SIL3	(ID0002 , dataArrivo2) --

Figura 2-21, Spedizione di un Riscontro (modalità Piggy-Backing)

Gestione della ricezione di riscontri per la porta di dominio

La gestione della ricezione di riscontri, in OpenSPCoop PdD viene effettuata dallo componente 'Riscontri' sia della parte 'Imbustamento' che della parte 'Sbustamento' del modulo di controllo appoggiandosi ad un tabella 'RiscontriDaRicevere'. Vediamo cosa devono fare le due parti.

Imbustamento. La parte 'Imbustamento' del modulo di controllo inserisce informazioni nella tabella 'RiscontriDaRicevere', se al servizio invocato dal SIL mittente è associato l'header e-Gov profiloDiTrasmissione con l'attributo 'confermaRicezione=true' (informazioni prese dal registro dei servizi). Nel qual caso viene creata una entry nella tabella, con chiave di accesso l'ID della busta e-Gov che sarà inviata, e con valore associato l'ora e la data di invio.

Sbustamento. La parte 'Sbustamento' effettua il controllo della ricezioni di riscontri, ogni qualvolta riceve una busta e-Gov. Se sono presenti riscontri nella busta, vengono utilizzati gli ID di ogni riscontro come chiave di accesso alla tabella RiscontriDaRicevere, in modo da cancellare i riscontri avvenuti (entry esistente nella tabella).

Un **Thread di servizio**, si occuperà di verificare che i riscontri attesi, non superino un timeout specificato. In caso di superamento del timeout, il thread si occuperà di reinviare la busta e-Gov precedentemente elaborata e salvata in una history apposito dei messaggi inviati (**persistenza dei messaggi**).

Le figure seguenti illustrano un esempio di questo scenario.

*Step 1 (*Figura 2-22*): Invio di una busta e-Gov al SIL2.* La parte 'Imbustamento' deve creare una busta e-Gov (con ID=ID0001), che include l'header 'ConfermaRicezione=true'. Verrà creata una entry nella tabella 'RiscontriDaSegnalare' con chiave di accesso ID0001 e con valore la data e l'ora di creazione della busta.

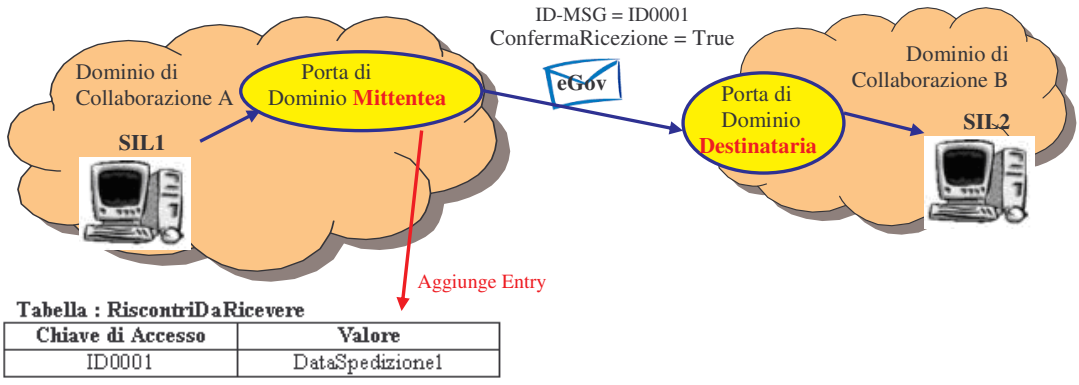


Figura 2-22, Spedizione prima busta contenente una richiesta di riscontro

*Step 2 (*Figura 2-23*): Invio di una busta e-Gov al SIL2.* La parte 'Imbustamento' deve creare una busta e-Gov (con ID=ID0002) associata allo stesso servizio invocato nello step 1. Verrà creata una entry nella tabella 'RiscontriDaSegnalare' come nello step 1.

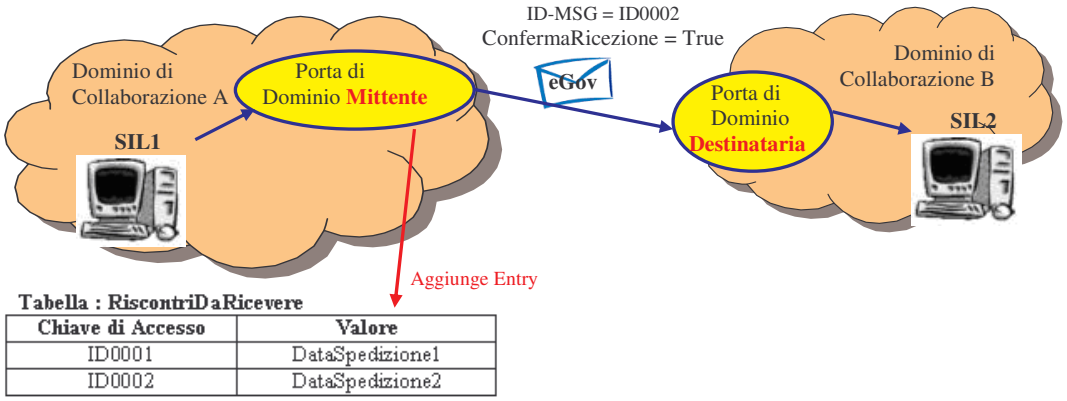


Figura 2-23, Spedizione seconda busta contenente una richiesta di riscontro

*Step 3 (*Figura 2-24*): Ricezione di una busta e-Gov, con una lista Riscontri contenente i riscontri da parte della porta di dominio destinataria delle buste inviatogli negli step 1 e 2.* La parte 'Sbustamento', nel componente 'Riscontri', verifica la presenza dei riscontri presenti nella lista della busta e-Gov con quelli presenti nella tabella RiscontriDaSegnalare. Ogni riscontro presente nella tabella, comporta la sua cancellazione nella tabella.

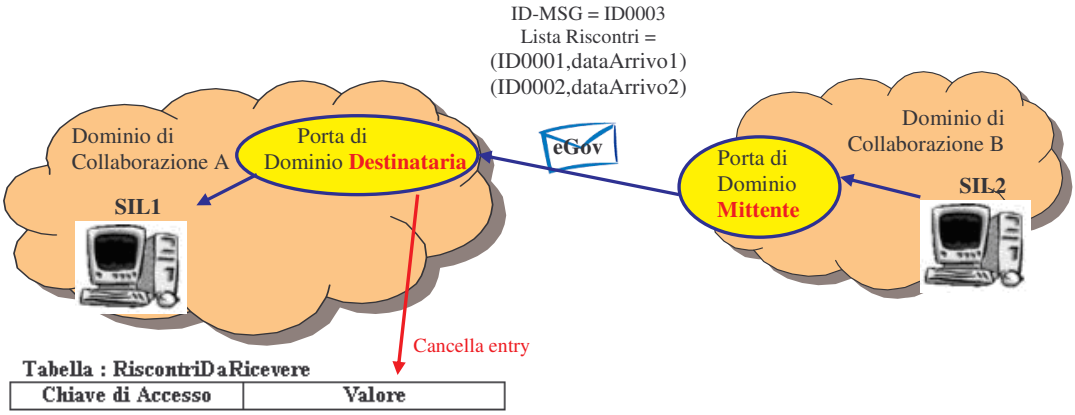


Figura 2-24, Ricezione di una busta SPCoop contenente riscontri di buste precedentemente inviate (Modalità Piggy-Backing)

2.2.4.5 Gestione dei Duplicati

Questa funzionalità si occupa di implementare la gestione dei duplicati descritta nel Profilo di Trasmissione di una busta SPCoop. La gestione viene attivata o meno a seconda del valore presente nell'attributo 'Inoltro' che può contenere:

- EGOV_IT_ALPIUUNAVOLTA
- EGOV_IT_PIUDIUNAVOLTA (default)

Ad un determinato servizio, nel registro dei servizi, viene associato uno dei due valori. Quindi al momento della creazione della busta e-Gov, compito della parte 'Imbustamento' del modulo di controllo, verrà inserito anche l'header Profilo di Trasmissione che avrà l'attributo 'Inoltro' correttamente configurato.
La gestione dei duplicati viene effettuata, dalla parte 'Sbustamento', solo nel caso in cui la busta e-Gov pervenuta alla porta di dominio, contenga l'attributo 'Inoltro = EGOV_IT_ALPIUUNAVOLTA'. Per gestire tale funzionalità le porte di dominio devono opportunamente gestire la **persistenza** delle buste e-Gov ricevute o parte di esse.
L'implementazione di questa funzionalità, in OpenSPCoop, si basa sul campo "identificatore", previsto nella busta, che identifica univocamente ogni messaggio. Il componente 'Duplicati', si occupa di verificare che l'identificatore della busta non risulti già ricevuto, interagendo con un history delle buste precedentemente elaborate. Nel caso in cui il messaggio risulti già ricevuto, viene scartato.

Componendo la funzionalità ({inoltro}=EGOV_IT_ALPIUUNAVOLTA) con quella illustrata precedentemente ({confermaRicezione}=*true*), è possibile implementare una semantica di tipo **ESATTAMENTE UNA VOLTA (ONCE AND ONLY ONCE).**
Nella figure seguenti viene illustrato un esempio di perdita di buste SPCoop. Nella Figura 2-25 viene descritto il caso in cui una porta di dominio mittente, dopo la scadenza di un opportuno intervallo di tempo in cui non ha ricevuto un riscontro di una busta precedentemente inviata, provvede a rispedire il messaggio originario.

Nella Figura 2-26 viene invece raffigurato un contesto dove una porta di dominio, riceve una busta precedentemente già elaborata, e di cui aveva già effettuato un riscontro. Il filtraggio dei duplicati permetterà di non rielaborare la busta per la consegna al destinatario finale, ma comunque la porta di dominio re-invierà un messaggio contenente un nuovo acknowledgment di riscontro alla Porta di Dominio Mittente.

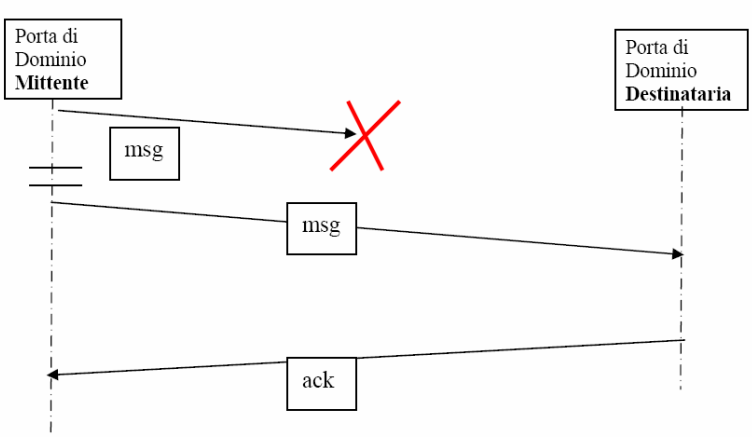


Figura 2-25, Perdita di una busta SPCoop spedita da una porta di dominio.

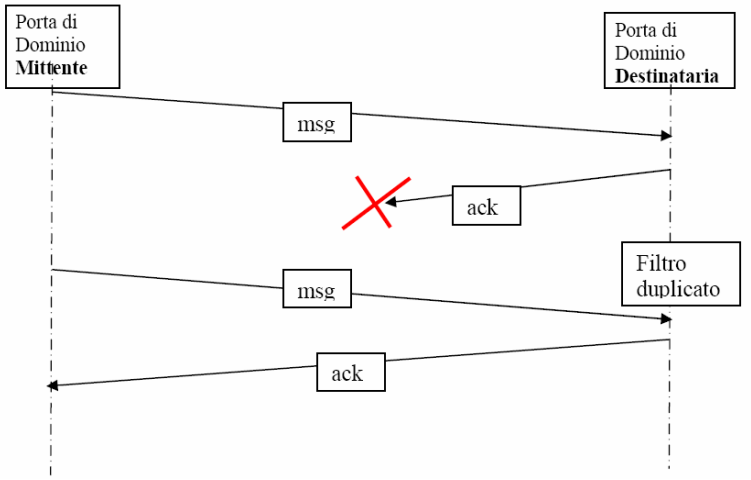


Figura 2-26, Perdita di un riscontro

2.2.4.6 Consegna in Ordine

Nelle specifiche della busta SPCoop sono introdotti i concetti di Collaborazione e Sequenza. La collaborazione identifica una conversazione. Se un servizio prevede più scambi tra le parti (modalità conversazionale), nell'header della busta e-Gov deve essere presente l'elemento Collaborazione, che contiene l'identificativo del messaggio contenente la richiesta che ha originato i successivi messaggi (identificazione a capostipite). In tal modo è possibile identificare i messaggi relativi ad una stessa istanza di interazione (es. messaggi necessari in un profilo asincrono) ed inoltre è possibile anche correlare più interazioni in una stessa conversazione.
Quando è presente la collaborazione, in un messaggio e-Gov, è possibile anche la presenza di un altro elemento dell'header della busta: la sequenza. Essa permette di identificare il numero di sequenza del messaggio nella Collaborazione (conversazione) di appartenenza, e quindi viene utilizzato dalla porta di dominio destinataria per consegnare i messaggi ricevuti, appartenenti ad una stessa conversazione, nell'ordine in cui sono stati spediti.
La gestione del modulo di controllo per identificare i messaggi relativi ad una stessa istanza di interazione (es. messaggi necessari in un profilo asincrono) viene semplicemente attivata nel caso venga richiesta (tramite informazioni associate ad un servizio presente nel registro dei servizi), aggiungendo il campo conversazione nei vari messaggi scambiati per il profilo gestito (vedi paragrafi 2.2.4.8 e 2.2.4.9, per quanto riguarda i profili di collaborazione asincroni).
La gestione del modulo di controllo per correlare più interazioni in una stessa conversazione è più complessa e differisce nella parte 'Imbustamento' o 'Sbustamento' del modulo di controllo stesso.

Imbustamento. In questa fase, il modulo di controllo si occupa, tra le altre cose, di creare la busta e-Gov, imbustando la richiesta SOAP, ottenendo le informazioni necessarie per l'imbustamento dal registro dei servizi. Tra queste informazioni ne sarà presente anche una che richiederà la presenza di una collaborazione verso un servizio. La collaborazione verso un servizio viene identificata dalla quadrupla:

[ServiceProvider, Servizio, Azione, MittenteCollaborazione]

Nel caso che la richiesta verso il servizio applicativo sia la prima di una sequenza di richieste successive, la collaborazione dovrà essere creata. Altrimenti sarà effettuata la continuazione di una collaborazione precedentemente instaurata.
Se la collaborazione deve essere creata, verrà creata una entry in una tabella apposita 'CollaborazioneInSpedizione' utilizzata per la gestione della Collaborazione. La chiave di accesso alla tabella sarà una chiave composta da tutte e quattro le informazioni che identificano una collaborazione. Oltre alla chiave di accesso, verrà inserita, nell'entry appena creata il valore dell'ID della busta e-Gov che è stata costruita, che è la busta capostipite della conversazione. Quindi la busta e-Gov si troverà a possedere l'identificativo messaggio e l'elemento Conversazione con lo stesso valore.
Se la collaborazione è gia stata creata, verrà acceduta la tabella 'CollaborazioneInSpedizione', con la chiave di accesso composta, formata dalle quattro informazioni elencate precedentemente, per prelevare l'ID della busta capostipite della conversazione, e dopodichè questo identificativo sarà utilizzato nell'elemento Collaborazione della busta e-Gov creata. Quindi l'elemento Collaborazione sarà diverso dall'identificativo della busta stessa.

Per gestire la consegna in ordine, i valori della sequenza sono gestiti attraverso un'altra apposita tabella, 'Sequenza', che sarà acceduta con l'ID della conversazione, e che conterrà un contatore che indica il prossimo numero di sequenza da utilizzare. Quando una collaborazione deve essere creata, verrà creata anche l'apposita entry dentro la tabella 'Sequenza' con il contatore inizializzato ad 1. Mentre per successivi messaggi della collaborazione, viene utilizzato il contatore, dopo averlo incrementato, per ottenere il numero di sequenza da utilizzare nella busta e-Gov da inviare.

Sbustamento. In questa fase, il modulo di controllo si occupa, tra le altre cose, di sbustare la busta e-Gov pervenuta alla porta di dominio, per poi consegnarla all'applicazione destinataria. La gestione di questa consegna cambia a seconda che la busta arrivata sia la busta capostipite della conversazione, oppure un'altra busta.

La busta capostipite è riconoscibile attraverso un semplice controllo, poiché sia l'identificativo del messaggio che l'elemento Collaborazione nell'header e-Gov posseggono lo stesso identificativo.

Se la busta è la capostipite della conversazione, verrà creata una entry, con chiave di accesso il valore dell'ID della busta pervenuta, in una tabella 'CollaborazioneInRicezione' utilizzata per la gestione della Collaborazione, che conterrà il valore 2. Questo valore indicherà il prossimo elemento Sequenza aspettato ed è significante solo nel caso in cui le buste e-Gov della conversazione posseggano l'elemento Sequenza. Altrimenti il valore a se stante è insignificante, ma è importante la presenza della entry con chiave di accesso l'identificativo della busta capostipite per riconoscere la collaborazione nelle buste successive alla capostipite, durante la validazione della busta.
Se la busta è una tra le successive alla capostipite, verrà acceduta la tabella CollaborazioneInRicezione, con chiave di accesso il valore dell'elemento Conversazione (ID messaggio capostipite) presente nell'header e-Gov, per verificare innanzitutto l'esistenza della collaborazione richiesta. Dopodichè, se presente l'elemento sequenza nell'header e-Gov, viene confrontato con il valore presente nell'entry della tabella CollaborazioneInRicezione precedentemente acceduta. Se il valore è uguale, la busta può essere sbustata e consegnata all'applicazione destinataria (tramite una apposita porta applicativa), ed il valore nella corrispondente entry della tabella deve essere incrementato. Altrimenti, se il valore è maggiore, la consegna della busta non può essere effettuata, poiché arrivata fuori ordine, e deve essere congelata, fino all'arrivo di successive buste che portano il contatore uguale al numero di sequenza della busta congelata. Solo dopo che queste buste sono state elaborate e consegnate all'applicazione destinataria, la busta congelata può essere ripresa in considerazione per la consegna finale.

Le figure nelle prossime pagine illustrano un esempio di conversazione con sequenza.

Step 1 (Figura 2-27): Messaggio Capostipite.

La parte 'Imbustamento' (porta di dominio mittente) deve creare una busta e-Gov (con ID=ID0001) associata ad un servizio invocato dal SIL1 che necessita sia della collaborazione che della sequenza. Verrà creata una entry nella tabella 'ConversazioneInSpedizione' con chiave di accesso [SIL2,anagrafe,modifica, SIL1] e con valore l'ID della busta e-Gov spedita (ID0001). Inoltre verrà creata una entry nella tabella 'Sequenza' con chiave di accesso l'ID della busta e-Gov (ID0001) e con valore il prossimo numero di sequenza da utilizzare (2).

La parte 'Sbustamento' (porta di dominio destinataria), all'arrivo della busta di e-Gov, vedrà che si tratta di una busta e-Gov capostipite di una conversazione, e quindi creerà una entry nella tabella 'ConversazioneInRicezione', con chiave di accesso l'ID della collaborazione e con valore il successivo numero di sequenza aspettato (2).

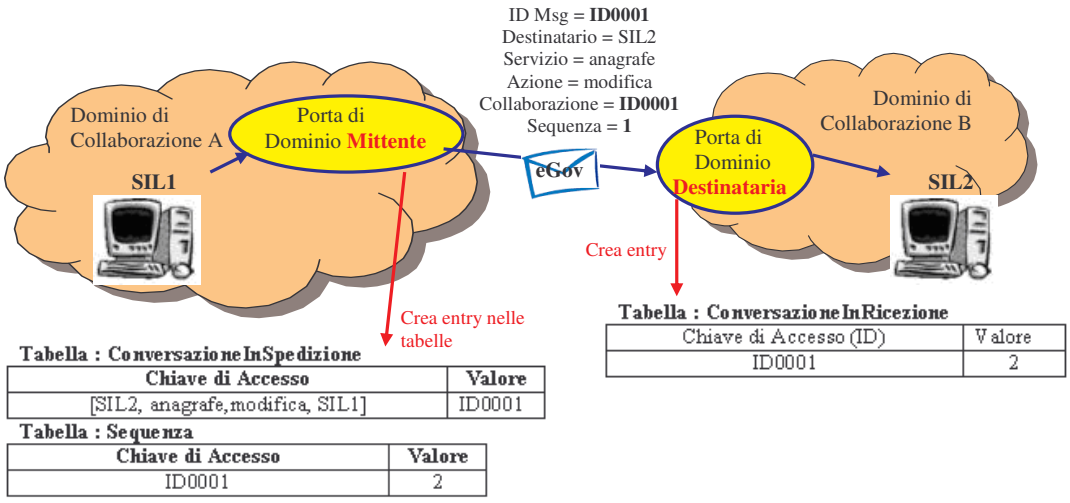


Figura 2-27, Spedizione busta capostipite di una conversazione

Step 2 (Figura 2-28): Messaggio non capostipite, in ordine. Nello step 1 è stato spedito il messaggio capostipite dalla porta di dominio mittente alla porta di dominio destinataria, e adesso viene inviato il secondo messaggio della conversazione allo stesso servizio applicativo dello step1.

La parte 'Imbustamento' (porta di dominio mittente) vede l'esistenza dell'entry [SIL2,anagrafe,modifica,SIL1] nella tabella 'ConversazioneInSpedizione', ed utilizza l'ID associato per ottenere il valore, presente nella tabella Sequenza, da associare all'header Collaborazione della busta e-Gov da creare. Dopo averne utilizzato il valore, incrementa il contatore.

La parte 'Sbustamento (porta di dominio destinataria), vede arrivare una busta e-Gov, che possiede una collaborazione esistente (entry ID0001 nella tabella ConversazioneInRicezione) e con un valore di Sequenza corretto (valore associato all'entry è esattamente 2). La porta di dominio destinataria, può quindi consegnare la busta arrivata alla porta applicativa, poiché risulta in ordine di consegna, incrementando il valore presente nella tabella 'ConversazioneInRicezione' associato all'entry con chiave di accesso [ID0001].

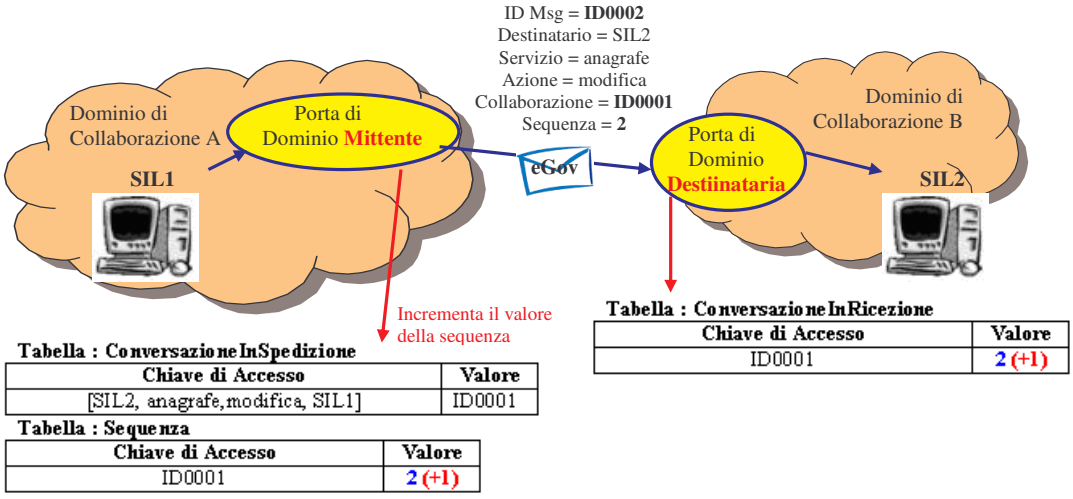


Figura 2-28, Spedizione seconda busta di una conversazione

Step 3 (Figura 2-29): Messaggio non capostipite e fuori ordine. Negli step precedenti sono stati spediti dalla porta di dominio di collaborazione A due messaggi di una stessa collaborazione con numero di sequenza uguale rispettivamente ad 1 e 2 (step1 e 2). A differenza di quanto visto nello step 2, supponiamo invece che la porta di dominio applicativa non abbia ricevuto ancora la busta e-Gov con sequenza uguale a 2. Supponiamo quindi che l'entry nella tabella 'ConversazioneInRicezione', con chiave di accesso 'ID0001', non abbia ancora subito l'incremento. Lo stato dell'entry sarà quindi quello dello step 1, quando la entry è stata appena creata [ID0001, 2]).

In questo step viene inviato il 3° messaggio della conversazione allo stesso servizio applicativo dello step1 e 2. Supponiamo che la porta di dominio destinataria riceva normalmente la busta e-Gov con sequenza uguale a 3 (quindi prima della busta con sequenza uguale a 2).

La parte 'Sbustamento' della porta in questione, noterà che la busta include una collaborazione esistente (entry ID0001 nella tabella ConversazioneInRicezione) ma con un valore di Sequenza scorretto (valore associato all'entry è 2, mentre la busta possiede sequenza uguale a 3). La busta e-Gov non potrà quindi essere consegnata alla porta applicativa destinataria, ma deve essere congelata fino all'arrivo della busta con e-Gov con sequenza uguale a 2, poiché arrivata fuori ordine.

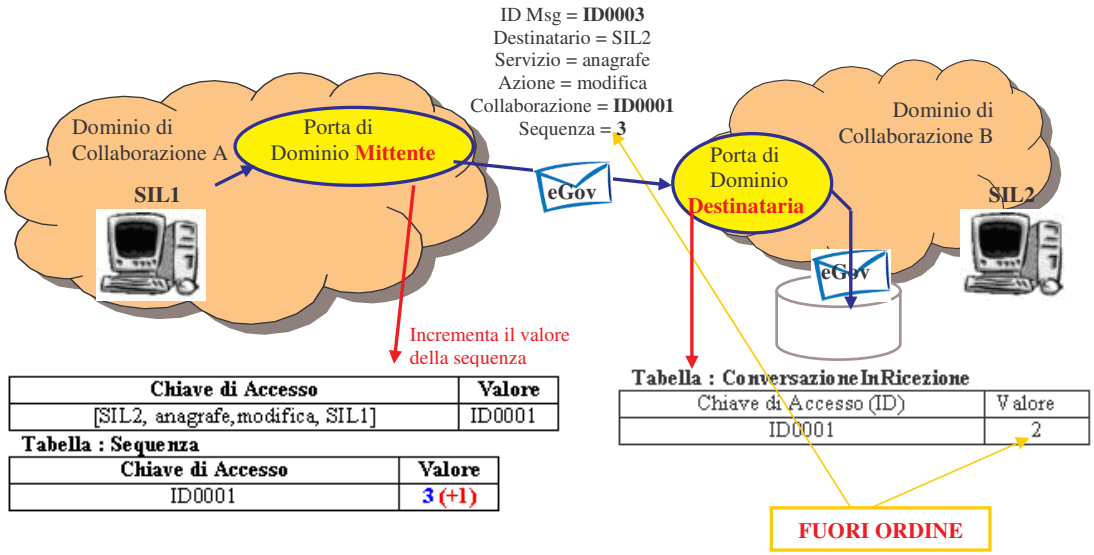


Figura 2-29, Scenario in cui una porta di dominio riceve una busta, appartenente ad una conversazione, non in ordine.

Step 4 (Figura 2-30 e Figura 2-31): Messaggio non capostipite, in ordine, che scongela anche altri msg precedentemente congelati. Nei precedenti step sono stati spediti dalla porta di dominio mittente tre messaggi di una stessa collaborazione con numero di sequenza uguale rispettivamente ad 1, 2 e 3, mentre la porta di dominio destinataria ha ricevuto solo il messaggio capostipite (ed è quindi in attesa di una busta con sequenza 2) e il messaggio con sequenza uguale a 3 (che quindi è stato congelato come è descritto nello step 3). Supponiamo in questo step, che la porta di dominio destinataria riceva il messaggio con sequenza 2, che è in ritardo per problemi ad es. di routing. La parte 'Sbustamento della porta di dominio in questione, esaminerà la busta e-Gov ricevuta, notando che la collaborazione inclusa nella busta è già esistente (entry ID0001 nella tabella ConversazioneInRicezione) e che il valore di Sequenza è corretto (valore associato all'entry 2).

La porta di dominio destinataria, può quindi consegnare la busta arrivata, poiché risulta in ordine, incrementando il valore presente nella tabella 'ConversazioneInRicezione' associato all'entry con chiave di accesso ID0001. Quanto dette viene illustrato nella Figura 2-30.

La parte 'Sbustamento, inoltre, deve controllare se sono state precedentemente congelate buste e-Gov con collaborazione 'ID0001' e sequenza '3' (valore del prossimo numero di sequenza aspettato). Poiché il controllo ha esito positivo, può essere effettuato lo scongelamento, e quindi viene consegnata la busta precedentemente congelata, poiché risulta adesso in ordine, incrementando il valore presente nella tabella 'ConversazioneInRicezione' associato all'entry con chiave di accesso ID0001. Lo scongelamento andrà avanti fino a che la ricerca di buste nel DB di buste congelate ha successo. Quanto detto viene illustrato nella Figura 2-31.

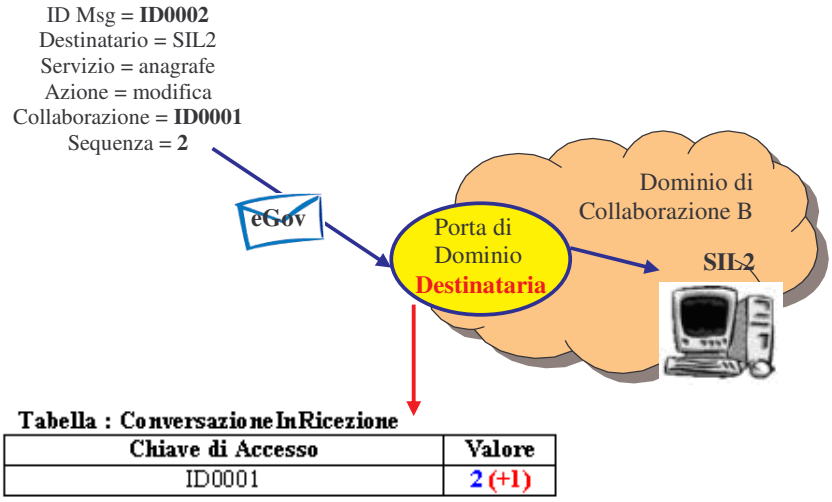


Figura 2-30, Scenario in cui una porta di dominio riceve una busta, appartenente ad una conversazione, il cui ritardo ha causato precedentemente il congelamento di altre buste.

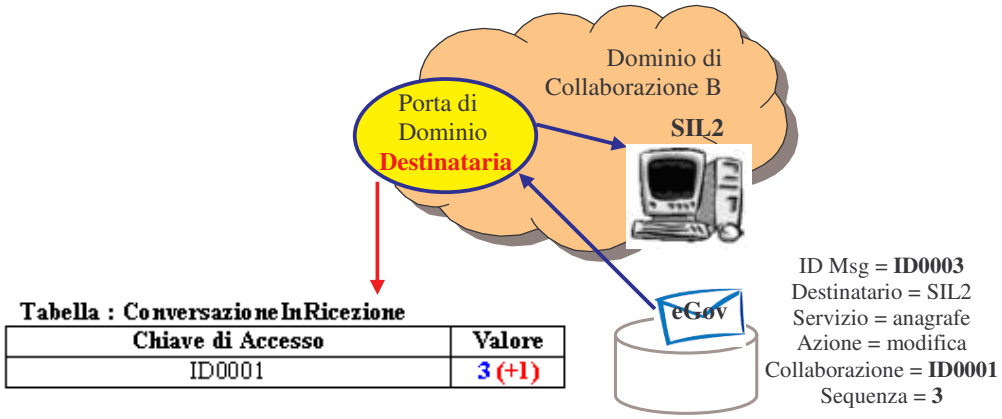


Figura 2-31, Scongelamento di buste precedentemente congelate poiché arrivate fuori ordine.

2.2.4.7 Profilo di Collaborazione Sincrono

Il profilo di collaborazione definisce lo schema di interscambio dei messaggi tra porte di dominio. Esistono quattro possibili profili di collaborazione. Gli stadi di questi profili sono gestiti da apposite strutture dati modificate dal componente 'Profilo di Collaborazione' presente sia nella parte 'Imbustamento' che nella parte 'Sbustamento' del modulo di controllo.

Il profilo di collaborazione MessaggioSingoloOneWay non necessità di strutture dati per essere gestito, essendo composta da una singola richiesta, e quindi non viene studiato in questa progettazione.

Il profilo di collaborazione Sincrono necessità invece di alcune strutture dati. Con questo profilo la porta di dominio mittente invia una busta SPCoop, su una connessione da lei attivata, alla porta di dominio destinataria, rimanendo in attesa (attiva) della busta di risposta. Presso il dominio dove risiede la porta applicativa viene ricevuto ed elaborata la busta e-Gov, invocando il servizio applicativo richiesto. La risposta del servizio applicativo viene re-imbustata e trasmessa alla porta di dominio mittente. La Figura 2-32 illustra lo scambio delle buste e-Gov che avvengono in questo profilo di trasmissione.

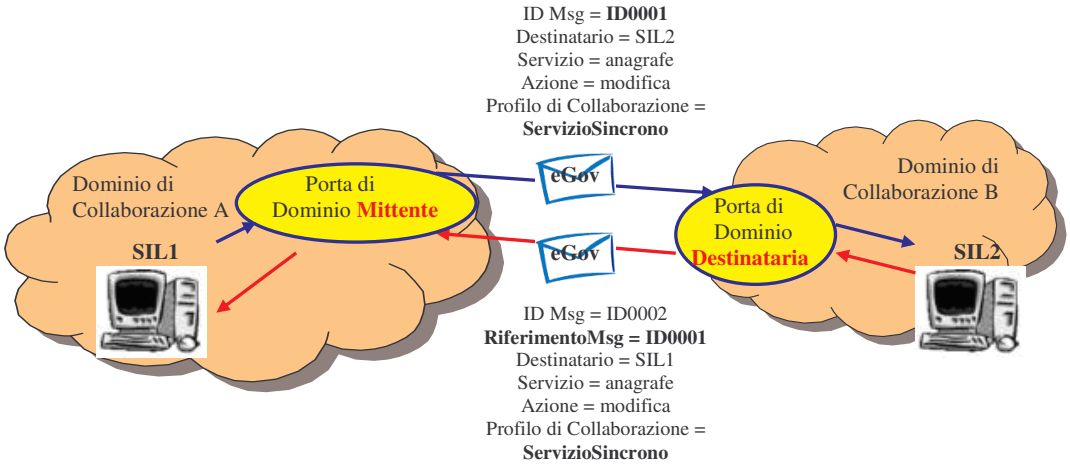


Figura 2-32, Buste SPCoop scambiate in un profilo di collaborazione Sincrono

E' interessante illustrare il funzionamento di questo profilo, direttamente dentro l'architettura OpenSPCoop PdD, per evidenziare l'attesa attiva effettuata dai moduli dell'architettura sulla connessione attivata per inviare e ricevere le buste.

E' utile dividere la gestione di questo profilo in due parti, la gestione dentro l'architettura della porta di dominio mittente e dentro quella della porta di dominio destinataria.

Porta di dominio mittente.

La Figura 2-33 illustra i componenti coinvolti nell'architettura della porta di dominio mittente. Le componenti attivate nella gestione della creazione della busta e-Gov e spedizione verso la porta di dominio destinataria sono evidenziate con frecce rosse. Le componenti coinvolte nella ricezione della risposta sincrona, sbustamento e consegna al SIL1 sono evidenziate con frecce blu.

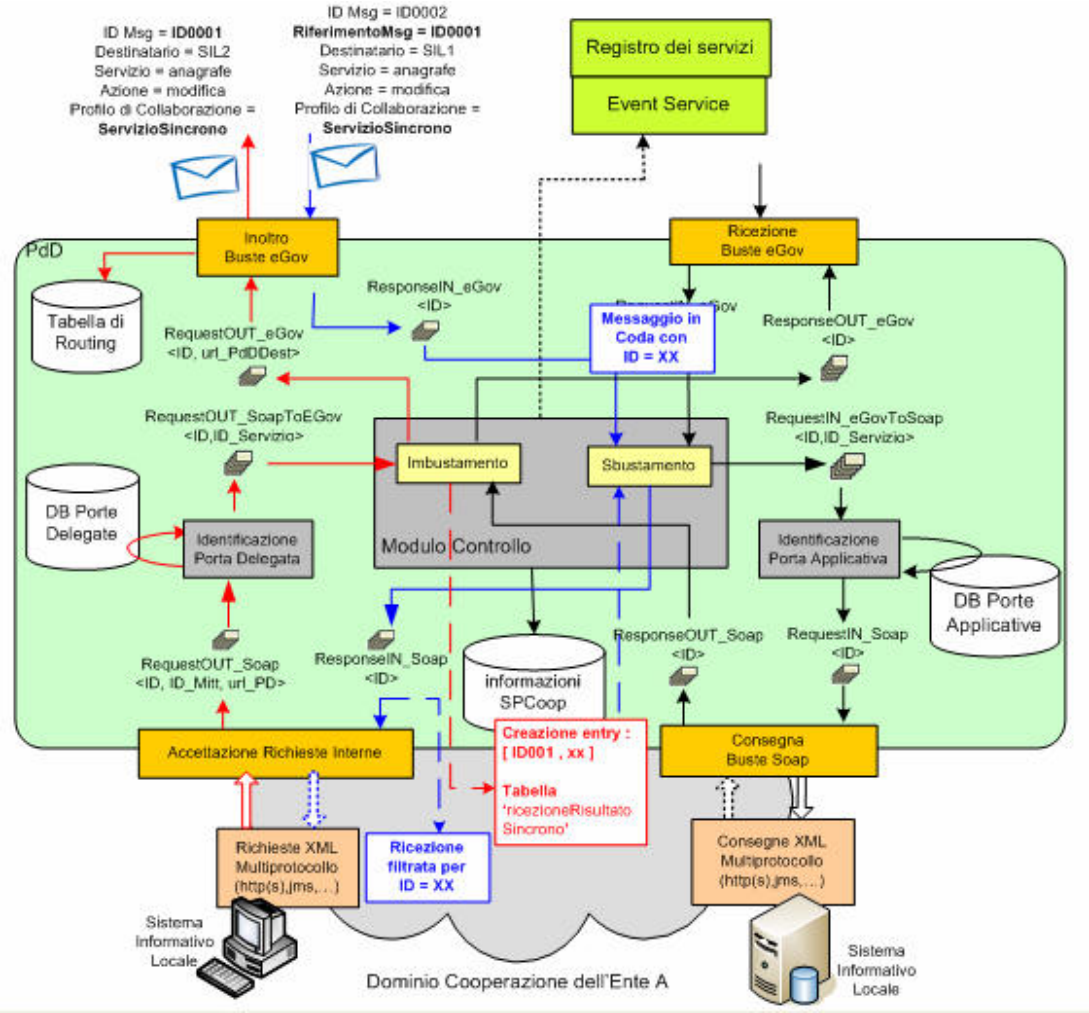


Figura 2-33, Gestione del profilo sincrono all'interno dell'architettura di una porta di dominio che riveste il ruolo 'mittente'

Il componente 'Profili di Collaborazione', della parte **Imbustamento** del modulo di controllo, dovrà gestire una tabella 'ricezioneRisultatoSincrono' dove verrà creata una entry con chiave di accesso uguale all'ID della busta e-Gov che sarà spedita e un valore associato comprendente l'identificatore utilizzato per il messaggio scambiato nelle code persistenti dell'architettura, creato dal modulo 'Accettazione Richeste Interne' al momento dell'inserimento della richiesta Soap nella coda RequestOUT_Soap. Questo ID del messaggio inserito nelle code rimane inalterato per tutto il ciclo di vita di gestione della porta delegata invocata dal SIL1, all'interno dell'architettura; quindi i passaggi tra le varie code persistenti, non modificheranno il valore dell'identificatore. Il creatore di questo identificatore ne ha bisogno alla fine del ciclo vitale di processamento della richiesta gestita, poiché il thread del modulo 'Accettazione Richieste Interne' che si sta occupando della gestione della richiesta, potrà recuperare la risposta applicativa dalla coda ResponseIN_Soap, filtrando la ricezione con l'ID precedentemente creato. Quando riceverà la risposta Soap, la inoltrerà all'applicazione mittente secondo il protocollo SIL-to-PdD. L'entry nella tabella 'ricezioneRisultatoSincrono' è creata ogni qualvolta viene imbustata una richiesta di servizio con profilo di collaborazione Sincrono.
Il componente 'Profili di Collaborazione', della parte **Sbustamento** entra in azione quando riceve una busta e-Gov, con profilo Sincrono, che è una risposta ad una richiesta e-Gov Sincrona precedentemente inviata. Il componente riesce a riconoscere che la busta e-Gov arrivata è una risposta, grazie alla presenza del campo 'RiferimentoMessaggio' all'interno della busta e-Gov. Il valore del campo stesso sarà utilizzato per accedere nella tabella 'ricezioneRisultatoSincrono', ed ottenere l'ID del messaggio scambiato nelle code, ed associarlo alla risposta Soap (dopo averla sbustata dalle informazioni SPCoop) da inserire nella coda ResponseIN_Soap.

Porta di dominio destinataria. La Figura 2-34 illustra i componenti coinvolti nell'architettura della porta di dominio destinataria. Le componenti attivate nella gestione della ricezione, sbustamento, e consegna del contenuto della busta alla porta applicativa sono illustrate con frecce rosse. Le componenti coinvolte nella gestione della risposta (imbustamento del risultato ritornato dal servizio applicativo ed invio alla porta di dominio mittente) sono mostrate con frecce blu.

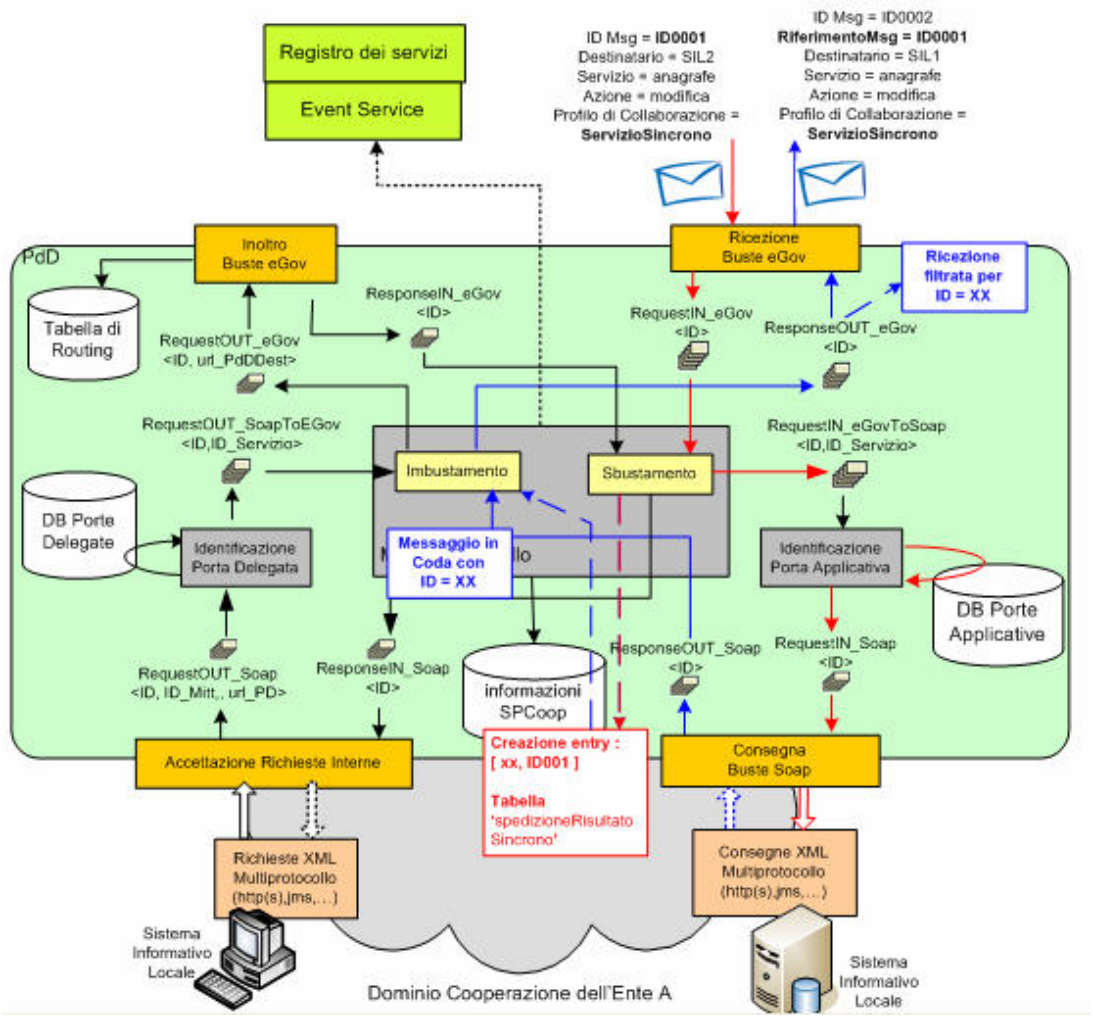


Figura 2-34, Gestione del profilo sincrono all'interno dell'architettura di una porta di dominio che riveste il ruolo 'destinatario'

Il componente ‘Profili di Collaborazione’, della parte **Sbustamento** del modulo di controllo, deve gestire una tabella ‘spedizioneRisultatoSincrono’ dove verrà creata una entry contenente chiave di accesso uguale all’ID del messaggio prelevato dalla coda RequestIN_eGov ed inserito (con stesso ID) nella coda RequestIN_eGovToSoap, dopo avere effettuato lo sbustamento. Il valore associato all’entry creata possiederà l’ID della busta e-Gov arrivata, oltre alle altre varie informazioni di e-Gov della busta stessa (utili per la costruzione della busta e-Gov di risposta). L’ID del messaggio scambiato nelle code persistenti, è importante, poiché il thread del modulo ‘Ricezione Buste e-Gov’ che si è occupato della gestione della richiesta, si è messo in **attesa attiva** di ricezione dalla coda ResponseOUT_eGov, filtrando la ricezione con l’ID associato al messaggio precedentemente inserito nella coda RequestIN_eGov.

Quando riceverà la risposta e-Gov, la inoltrerà alla porta di dominio mittente, che è a sua volta in attesa attiva della risposta. L'entry nella tabella 'spedizioneRisultatoSincrono' è creata ogni qualvolta viene sbustata una busta SPCoop contenente una richiesta di servizio con profilo di collaborazione Sincrono. E' possibile riconoscere che la busta e-Gov arrivata è una richiesta semplicemente controllando che non sia presente l'header 'RiferimentoMessaggio' nella busta e-Gov.

Il componente 'Profili di Collaborazione', della parte **Imbustamento** entra in azione quando preleva dalla coda ResponseOUT_Soap una busta Soap e l'associato ID del messaggio (non cambiando valore nei vari passaggi tra code, è rimasto quello creato in precedenza dal modulo 'Ricezione Buste e-Gov'). Il componente utilizza l'ID del messaggio nella coda per accedere alla tabella 'spedizioneRisultatoSincrono' e prelevare l'ID della precedente busta e-Gov che conteneva la richiesta di servizio (oltre alle altre informazioni e-Gov). Questa informazione (ID della busta e-Gov) sarà utilizzata all'interno della busta di risposta impostandola come valore del campo 'riferimentoMessaggio' (come si vede dalla Figura 2-34). La risposta e-Gov creata viene inserita nella coda ResponseOUT_eGov assieme all'ID scambiato nelle varie code dell'architettura. A questo punto il nodo 'Ricezione Buste e-Gov' potrà prelevare la busta contenente la risposta e-Gov ed inoltrarla alla porta di dominio mittente (rimasta in attesa attiva).

2.2.4.8 Profilo di Collaborazione Asincrono Simmetrico

Il profilo di collaborazione sincrono, comporta, come specificato dettagliatamente nel precedente paragrafo, una attesa attiva tra le porta di dominio per un intervallo di tempo necessario alla porta di dominio destinataria a gestire la busta ricevuta, redirigere il contenuto ad una porta applicativa, e ritornare indietro alla porta di dominio mittente una busta contenente un risultato applicativo. Questo scenario diventa improponibile in un contesto dove la porta applicativa necessita di ore o giorni per elaborare una risposta. Per superare questo svantaggio è necessario un altro profilo di collaborazione: il profilo asincrono simmetrico.

In questo profilo la porta di dominio mittente invia un messaggio contente una richiesta applicativa senza attendere una immediata risposta; la risposta potrà essere inviata, dalla porta di dominio destinataria, in un tempo successivo.

La differenza principale con il profilo sincrono la si ha nel fatto che la porta di dominio mittente non rimane in attesa attiva di una risposta, ma solo di una ricevuta che garantisce la presa in carico della richiesta da parte della porta di dominio mittente. La busta SPCoop contenente la richiesta di servizio effettuata dalla porta delegata nel dominio mittente, deve contenere il riferimento al servizio di ricezione delle risposte che la porta di dominio mittente ha attivato per la gestione della ricezione di una risposta applicativa. Questa informazione, utilizzata dalla porta di dominio destinataria per recapitare la risposta, è inclusa nell'attributo 'ServizioCorrelato' nell'elemento Profilo di Collaborazione della busta. Le figure seguenti illustrano i concetti appena specificati, fornendo un esempio di messaggi e-Gov scambiati durante una collaborazione asincrona simmetrica.

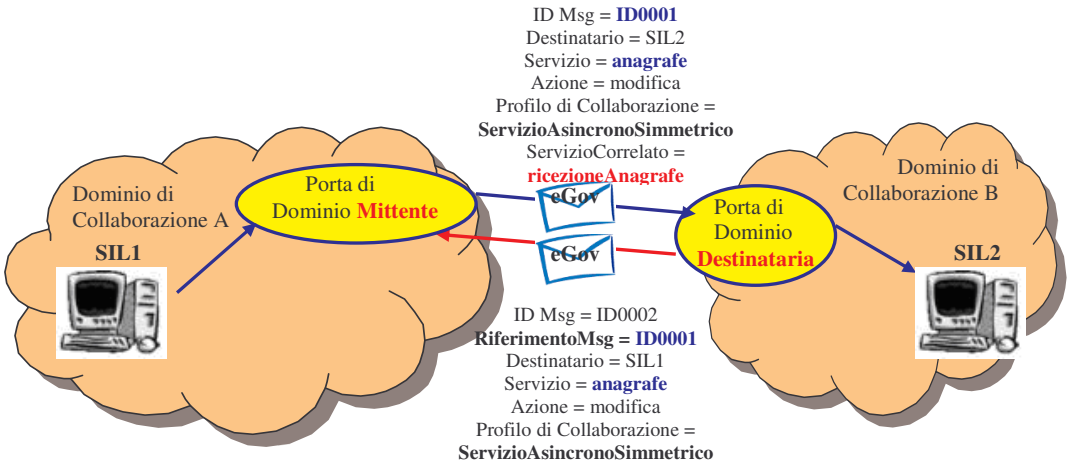


Figura 2-35, Buste SPCoop scambiate durante una richiesta di servizio con profilo Asincrono Simmetrico

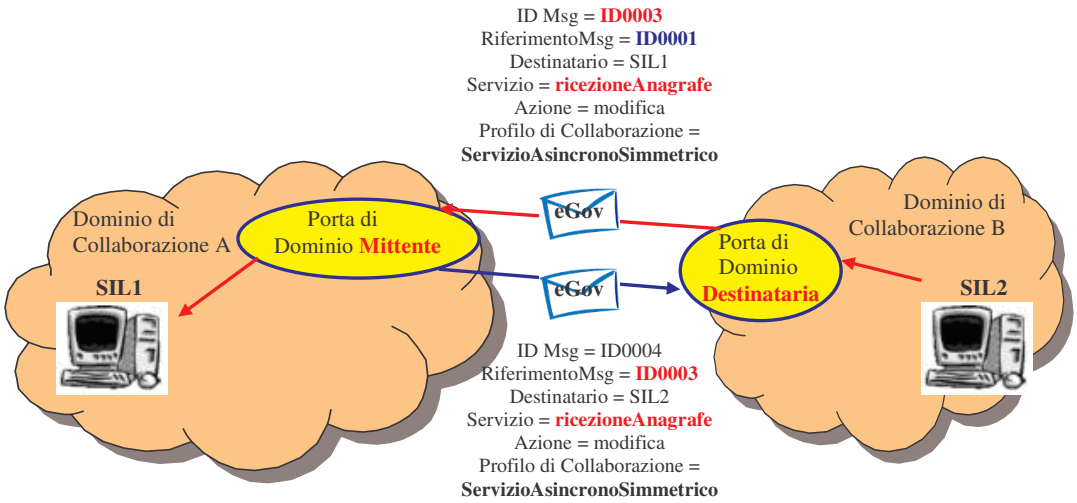


Figura 2-36, Buste SPCoop scambiate durante la consegna di una risposta applicativa associata ad un profilo Asincrono Simmetrico

Il flusso completo delle buste scambiate, con l'ordine cronologico, è visualizzato in Figura 2-37:

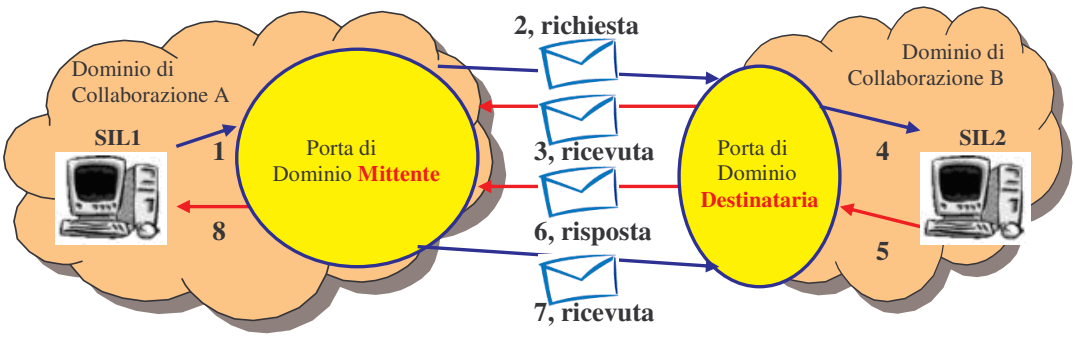


Figura 2-37, Flusso delle buste SPCoop scambiate durante un profilo asincrono simmetrico

Vediamo come **la gestione** vera e propria **dei vari stati di uno scambio di messaggi con profilo asincrono simmetrico** viene effettuata all'interno dell'architettura OpenSPCoop PdD dal modulo 'Profili di Collaborazione' sia della parte 'Imbustamento' che 'Sbustamento'. Come per il profilo Simmetrico è interessante illustrare il funzionamento di questo profilo dividendone la gestione in due parti, la gestione dentro la porta di dominio mittente e la gestione dentro la porta di dominio destinataria.

Porta di dominio mittente.

Il modulo 'Profili di Collaborazione', della parte **Imbustamento** del modulo di controllo, dovrà semplicemente creare la busta di e-Gov contenente le varie impostazioni associate al servizio richiesto (tra cui un identificativo di servizio correlato creato per la gestione dell'apposito profilo). Inoltre dovrà creare una entry in una tabella 'statoAsincronoSimmetrico', comprendente il valore dell'ID della busta e-Gov creata, il valore del servizio correlato associato e l'ID utilizzato per lo scambio dei messaggi tra le code persistenti dell'architettura, (ampiamente descritto nel paragrafo 2.2.4.7 relativo al profilo sincrono) creato dal modulo 'Accettazione Richieste Interne' al momento dell'inserimento della richiesta Soap nella coda RequestOUT_Soap. Questo ID interno è importante, poiché il thread del modulo 'Accettazione Richieste Interne' che si occupa della gestione della richiesta, potrà recuperare la risposta applicativa dalla coda ResponseIN_Soap, filtrando la ricezione sulla coda per ricevere il messaggio contenente l'ID in questione.. Quando riceverà la risposta Soap, la inoltrerà all'applicazione mittente secondo il protocollo SIL-to-PdD (funzionamento simile a quanto visto per il profilo sincrono).
Il modulo 'Profili di Collaborazione', della parte **Sbustamento** entra in gioco quando riceve una busta e-Gov, con profilo Asincrono Simmetrico. In questo caso si può trattare di una richiesta o di una risposta e-Gov con profilo Asincrono Simmetrico. Viene riconosciuto che si tratta della risposta (siamo nella porta di dominio mittente) poiché in essa è presente il campo riferimentoMessaggio e proprio grazie a questo campo viene validata la busta, controllando che si tratti di una risposta arrivata a causa di una richiesta precedente effettuata, utilizzando il campo insieme al valore del Servizio presente nella busta. Questi valori vengono utilizzati per accedere alla tabella 'statoAsincronoSimmetrico', e verificare l'esistenza di una entry che li contenga (validazione con successo). A questo punto la gestione viene lasciata al protocollo SIL-to-PdD dopo avere sbustato la busta, ed averla inserita nella coda ResponseIN_Soap insieme all'ID utilizzato per lo scambio dei messaggi nelle code, prelevato dall'entry acceduta (la entry sarà poi cancellata).

Dopo avere gestito la risposta pervenuta, il modulo della parte Sbustamento deve anche generare una ricevuta di conferma, per avvisare la porta del dominio di collaborazione B della presa visione della risposta. La ricevuta viene inviata come risposta nella connessione attivata dall'altra porta di dominio.

La Figura 2-38 illustra i concetti appena descritti (tralasciando la gestione delle ricevute). Le componenti attivate nella gestione della creazione della richiesta e-Gov e spedizione verso l'altra porta di dominio sono illustrate con frecce rosse. Le componenti coinvolte nella ricezione della risposta asincrona, sbustamento e sua consegna al SIL sono illustrate con frecce blu.

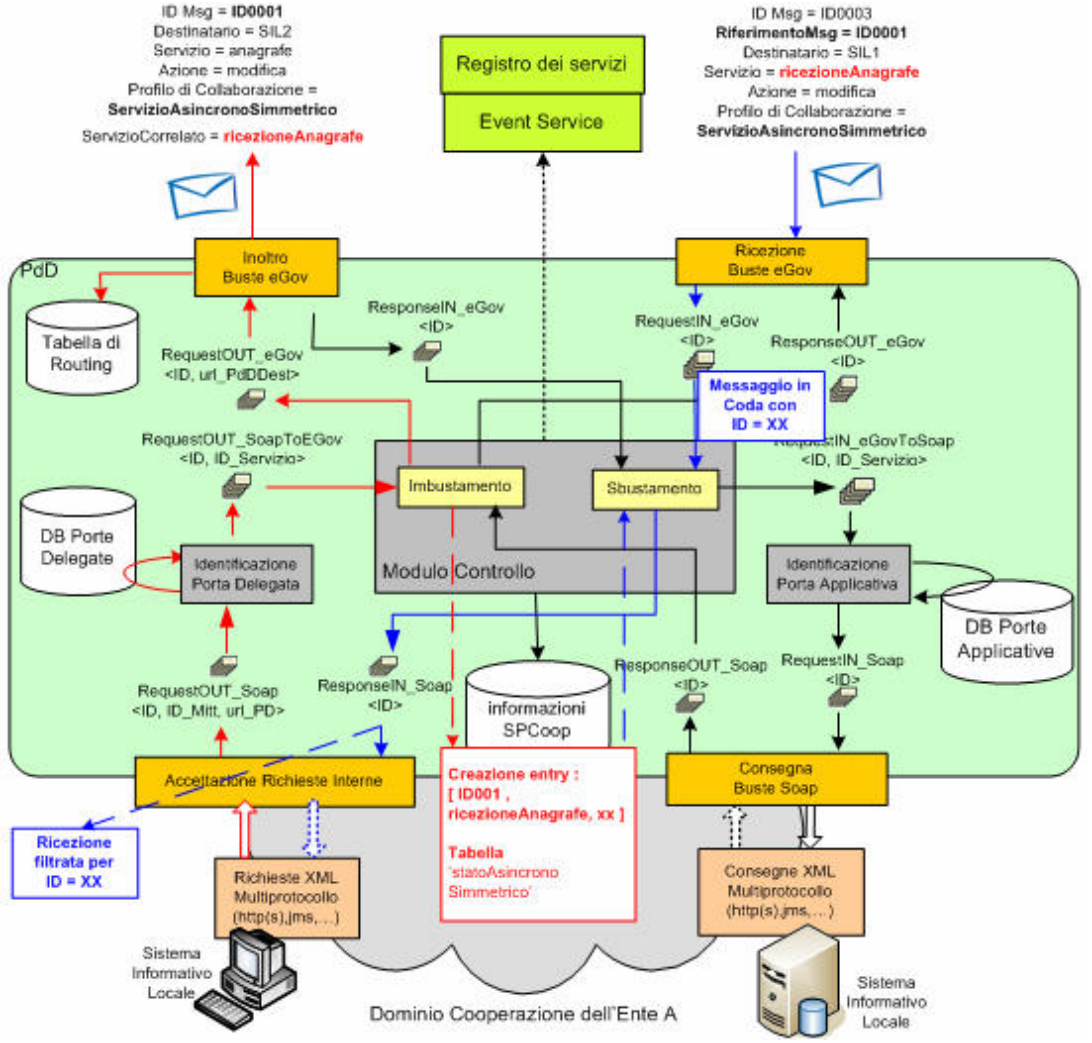


Figura 2-38, Gestione del profilo asincrono simmetrico all'interno dell'architettura di una porta di dominio che riveste il ruolo 'mittente'

Porta di dominio destinataria.

Il componente 'Profili di Collaborazione', della parte **Sbustamento** entra in azione quando la porta di dominio riceve una busta e-Gov, con profilo Asincrono Simmetrico. In questo caso si può trattare di una richiesta o di una risposta e-Gov con profilo Asincrono Simmetrico. Viene riconosciuto che si tratta della richiesta (siamo nella porta di dominio destinataria) poiché in essa non è presente il campo riferimentoMessaggio. La busta pervenuta sarà sbustata e consegnata alla porta applicativa invocata secondo il protocollo PdD-to-SIL.

Prima dello sbustamento, il componente 'Profilo di Collaborazione' si occupa di creare una entry in una tabella apposita 'risposteAsincroneSimmetriche' con chiave di accesso l'ID del messaggio prelevato dalla coda RequestIN_eGov ed inserito (con stesso ID) nella coda RequestIN_eGovToSoap, dopo avere effettuato lo sbustamento. L'entry creata possiederà il valore del ServizioCorrelato e dell'identificatore della busta e-Gov ricevuta, oltre alle altre informazioni e-Gov che saranno necessarie per creare la busta e-Gov contenente la risposta applicativa.

Dopo avere gestito la richiesta pervenuta, il modulo della parte Sbustamento deve anche generare una ricevuta di conferma, per avvisare la porta del dominio di collaborazione A della presa visione della richiesta. La ricevuta viene inviata come risposta nella connessione instaurata dall'altra porta di dominio.

Il componente 'Profili di Collaborazione', della parte **Imbustamento** entra in azione quando preleva dalla coda ResponseOUT_Soap una busta Soap e l'associato ID (il cui valore è rimasto quello creato in precedenza dal modulo 'Ricezione Buste e-Gov' e ri-associato ad ogni messaggio inserito nelle varie code). Il componente utilizza l'ID per accedere alla tabella 'risposteAsincroneSimmetriche' e prelevare l'ID e-Gov e il servizio correlato (oltre alle altre varie informazioni e-Gov) appartenente alla precedente busta SPCoop gestita che conteneva la richiesta di servizio. Queste informazioni (ID e servizio correlato) sono utilizzate durante la creazione della busta e-Gov di risposta rispettivamente per i valori dei campi 'riferimentoMessaggio' e 'Servizio'.

Per avere una visione più chiara di questa parte, è bene entrare direttamente nell'architettura della porta di dominio applicativa attraverso la Figura 2-39, dove viene evidenziato la ricezione di una richiesta (in rosso) e la gestione della risposta (in blu) tralasciando la gestione delle ricevute.

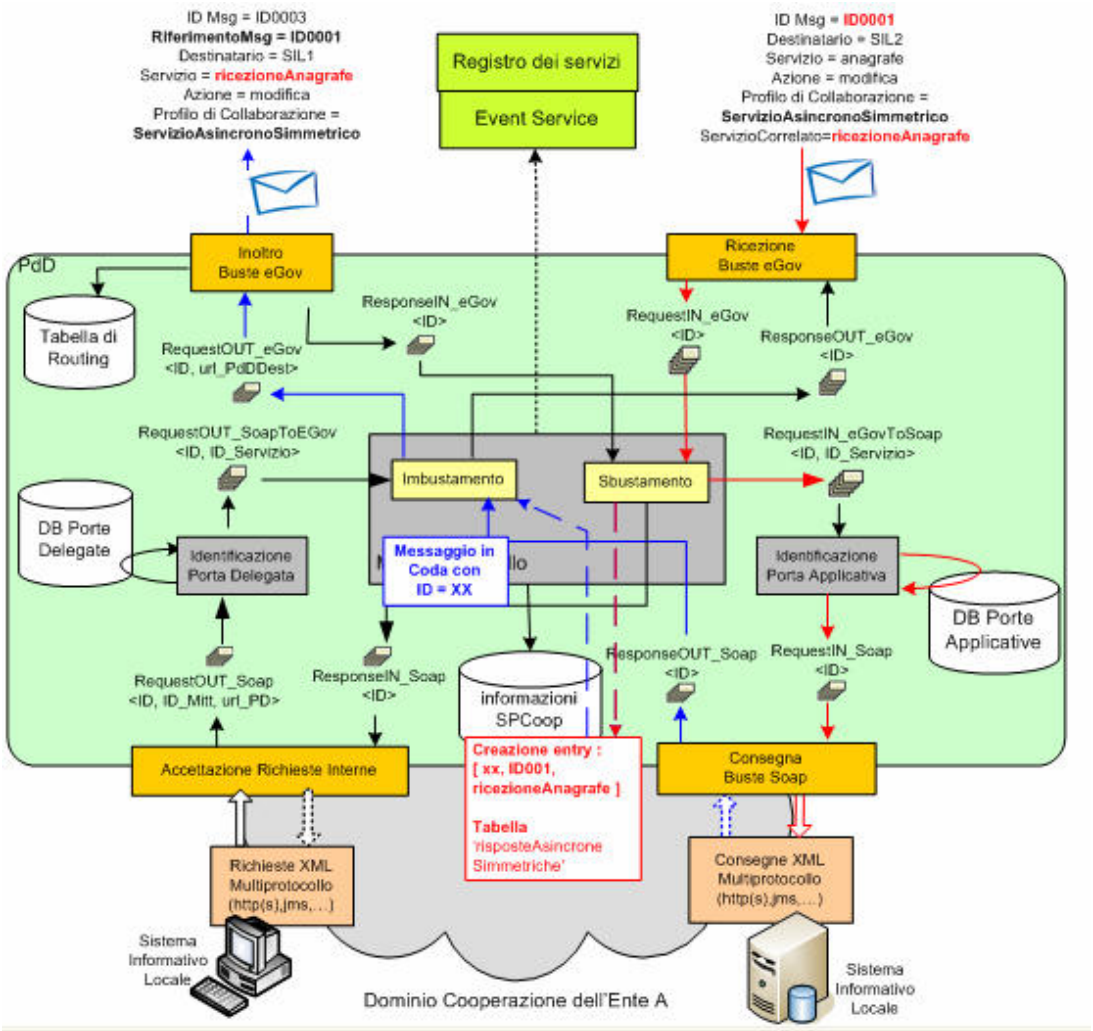


Figura 2-39, Gestione del profilo asincrono simmetrico all'interno dell'architettura di una porta di dominio che riveste il ruolo 'destinatario'

2.2.4.9 Profilo di Collaborazione Asincrono Asimmetrico

Il profilo di collaborazione asincrono simmetrico, descritto nel precedente paragrafo, potrebbe non essere attuabile in un contesto in cui la porta di dominio mittente sia protetta da un firewall che non accetta connessioni in ingresso. Il profilo sincrono permetterebbe di risolvere il problema, ma come è già stato precedentemente descritto, non è utilizzabile in un contesto dove la risposta applicativa è disponibile dopo ore o giorni. Per risolvere entrambi i problemi è presente un ulteriore profilo di collaborazione: il profilo asincrono asimmetrico.

La porta di dominio mittente invia un messaggio contenente una richiesta applicativa senza restare in attesa di una risposta; in un tempo successivo, la porta di dominio mittente richiede lo stato di esecuzione della propria richiesta alla porta di dominio destinataria rimanendo in attesa della risposta. Se il servizio è stato eseguito nel dominio destinatario, la relativa porta inoltrerà alla Porta di dominio mittente la risposta applicativa, altrimenti segnala che il processamento non è stato ancora completato e l'interrogazione sullo stato si ripeterà dopo un intervallo di tempo. Entrambe le richieste effettuate dalla porta di dominio mittente, saranno susseguite da una ricevuta di conferma (contenente il risultato nel caso di richiesta dello stato dell'operazione) generata dalla porta di dominio destinataria. Inoltre, l'attributo 'ServizioCorrelato' dell'elemento Profilo di Collaborazione, presente nell'header e-Gov della ricevuta alla richiesta, contiene il riferimento al servizio (esposto dalla porta di dominio destinataria) su cui effettuare il Polling della risposta.

Le figure sottostanti illustrano i concetti appena descritti, ed esempi di messaggi SPCoop di una collaborazione asincrona simmetrica.

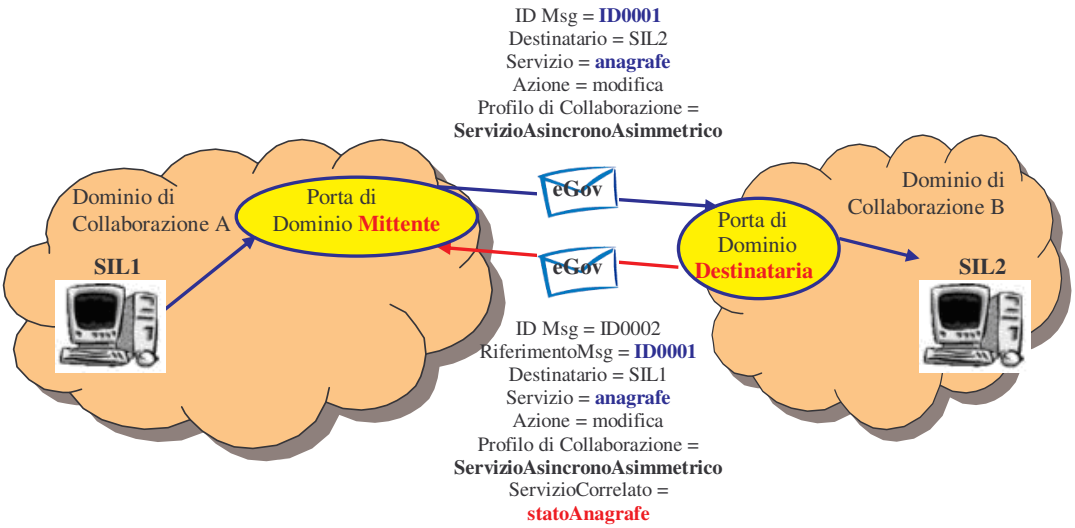


Figura 2-40, Buste SPCoop scambiate durante una richiesta di servizio con profilo Asincrono Asimmetrico

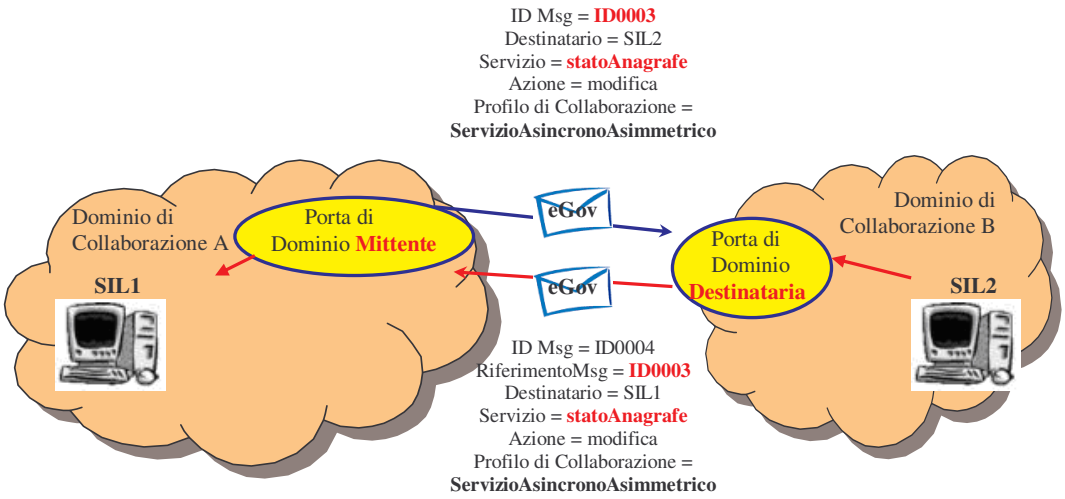


Figura 2-41, Buste SPCoop scambiate durante una richiesta dello stato di un operazione, con profilo Asincrono Asimmetrico, precedentemente richiesta

Il flusso completo delle buste scambiate, con l'ordine cronologico, è visualizzato in Figura 2-42.

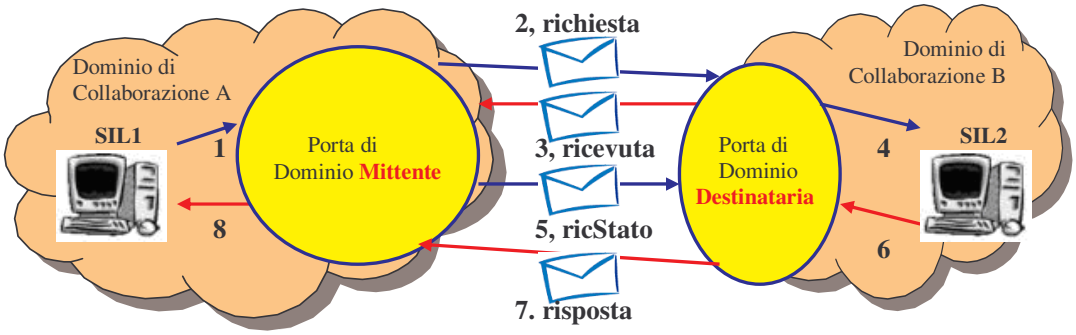


Figura 2-42, Flusso delle buste SPCoop scambiate durante un profilo asincrono asimmetrico

Vediamo come **la gestione** vera e propria **dei vari stati di uno scambio di messaggi con profilo asincrono asimmetrico** viene effettuata all'interno dell'architettura OpenSPCoop PdD dal modulo 'Profili di Collaborazione' sia della parte 'Imbustamento' che 'Sbustamento'. Come per il profilo Simmetrico e per il profilo Asincrono Simmetrico è interessante illustrare il funzionamento di questo profilo dividendone la gestione in due parti: la gestione dentro la porta di dominio mittente e la gestione dentro la porta di dominio destinataria.

Porta di dominio mittente.

1° Fase. Il componente 'Profili di Collaborazione', della parte **Imbustamento** entra in gioco quando riceve una richiesta di servizio a cui è legato un profilo Asincrono Asimmetrico. Questa parte crea una entry in una tabella 'RicezioneRisultatoAsincronoAsimmetrico' con chiave di accesso l'ID della busta e-Gov ricevuta e con valore l'ID, utilizzato nello scambio dei messaggi nelle code dell'architettura, creato dal modulo 'Accettazione Richieste Interne' al momento dell'inserimento della richiesta Soap nella coda RequestOUT_Soap. Questo ID architetturale è importante, poiché il thread del modulo 'Accettazione Richieste Interne' che si sta occupando della gestione della richiesta, potrà recuperare la risposta applicativa dalla coda ResponseIN_Soap, filtrando la ricezione con l'ID in questione. Quando riceverà la risposta Soap, la inoltrerà all'applicazione mittente secondo il protocollo SIL-to-PdD (funzionamento simile a quanto visto per il profilo sincrono e asincrono simmetrico).

2° Fase. Il componente 'Profili di Collaborazione', della parte **Sbustamento** ha bisogno di una tabella 'StatoAsincronoAsimmetrico'. Questa tabella è modificata all'arrivo di una ricevuta di una richiesta, inviata dalla porta di dominio destinataria sulla reply della connessione attivata dalla porta di dominio mittente. Viene creata una entry nella tabella 'StatoAsincronoAsimmetrico' con i valori invertiti presenti nei campo Mittente e Destinatario della ricevuta, il valore del campo 'servizioCorrelato', del campo 'Azione' e del campo 'RiferimentoMessaggio'. Tutte queste informazioni serviranno per conoscere il servizio su cui redirigere le successive richieste di polling.

3° Fase. Le informazioni salvate al passaggio della ricevuta alla richiesta, saranno poi utilizzate da un **Thread di servizio** che dopo un intervallo di tempo prefissato, legge tutte le entry della tabella e crea le apposite richieste di stato ai vari servizi di polling presenti nelle varie entry, creando apposite buste e-Gov. Inoltre per ogni busta e-Gov creata ed inviata, utilizza anche il campo RiferimentoMessaggio per accedere alla tabella 'RicezioneRisultatoAsincronoAsimmetrico' e cambiare l'ID precedente con l'ID della busta e-Gov contenente la richiesta stato appena inviata.

4° Fase. Il componente 'Profili di Collaborazione', della parte **Sbustamento,** utilizza la tabella 'RicezioneRisultatoAsincronoAsimmetrico' per effettuare la consegna della risposta applicativa contenuta in una ricevuta di una richiesta stato,

inviata dalla porta di dominio destinataria sulla reply della connessione attivata dalla porta di dominio. Viene acceduta l'entry nella tabella con il valore del campo RiferimentoMessaggio presente nella ricevuta pervenuta, per conoscere l'ID architetturale con cui il modulo 'Accettazione Richieste Interne' sta effettuando il filtro per la ricezione della risposta applicativa secondo il protocollo SIL-to-PdD.
Le figure sottostanti illustrano i concetti appena descritti, fornendo una astrazione dell'architettura OpenSPCoop.

*Fase 1 (*Figura 2-43*).* Creazione della richiesta e-Gov.

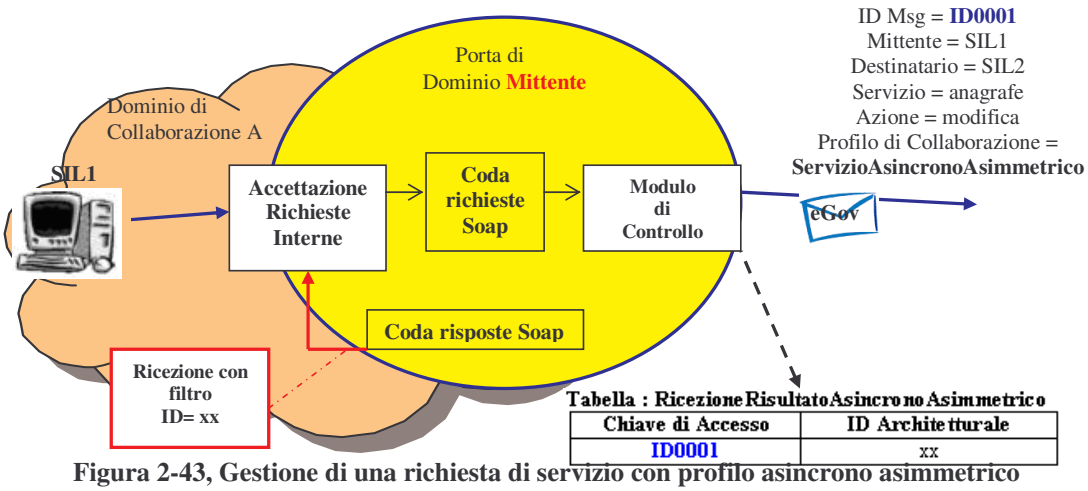


Figura 2-43, Gestione di una richiesta di servizio con profilo asincrono asimmetrico

*Fase 2 (*Figura 2-44*).* Ricezione della ricevuta relativa alla richiesta precedentemente effettuata.

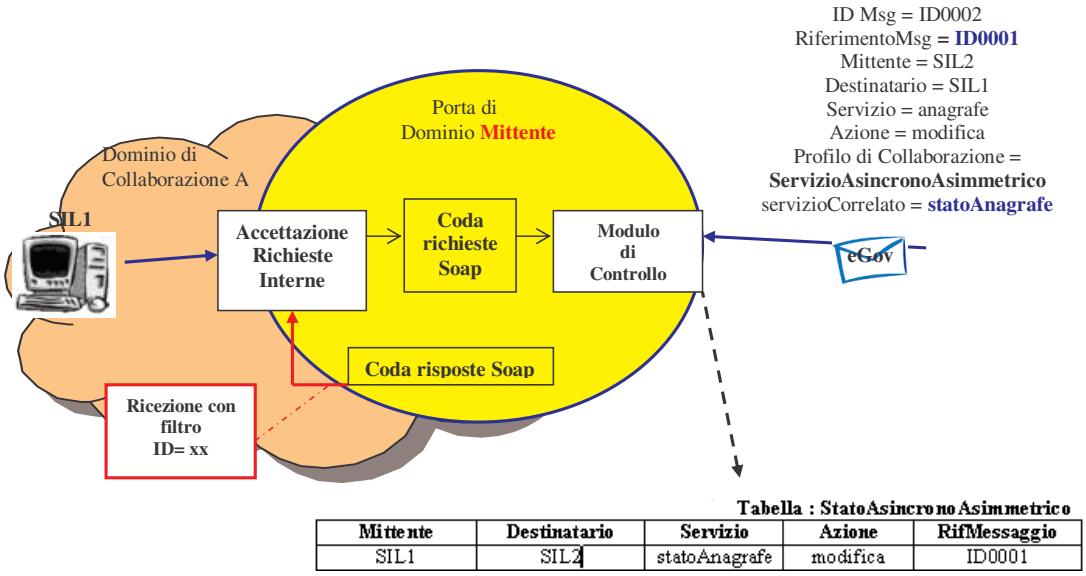


Figura 2-44, Ricezione della ricevuta relativa ad una richiesta, con profilo asincrono asimmetrico, precedentemente inviata

*Fase 3 (*Figura 2-45*).* Invio della richiesta stato con profilo AsincronoAsimmetrico da parte del Thread di Servizio ed aggiornamento della tabella 'RicezioneRisultatoAsincronoAsimmetrico', nell'entry con ID=RifMessaggio.

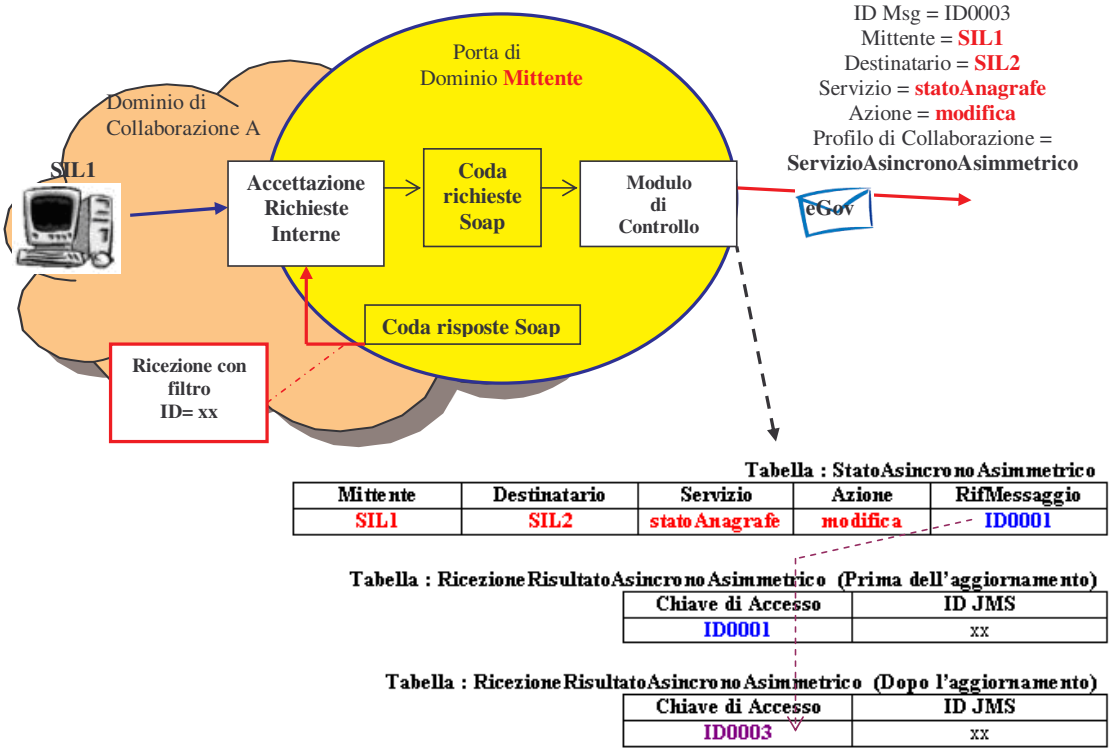


Figura 2-45, Invio di una richiesta di stato con profilo asincrono asimmetrico

*Fase 4 (*Figura 2-46*).* Ricezione della risposta e-Gov con profilo AsincronoAsimmetrico contenente la risposta applicativa Soap.

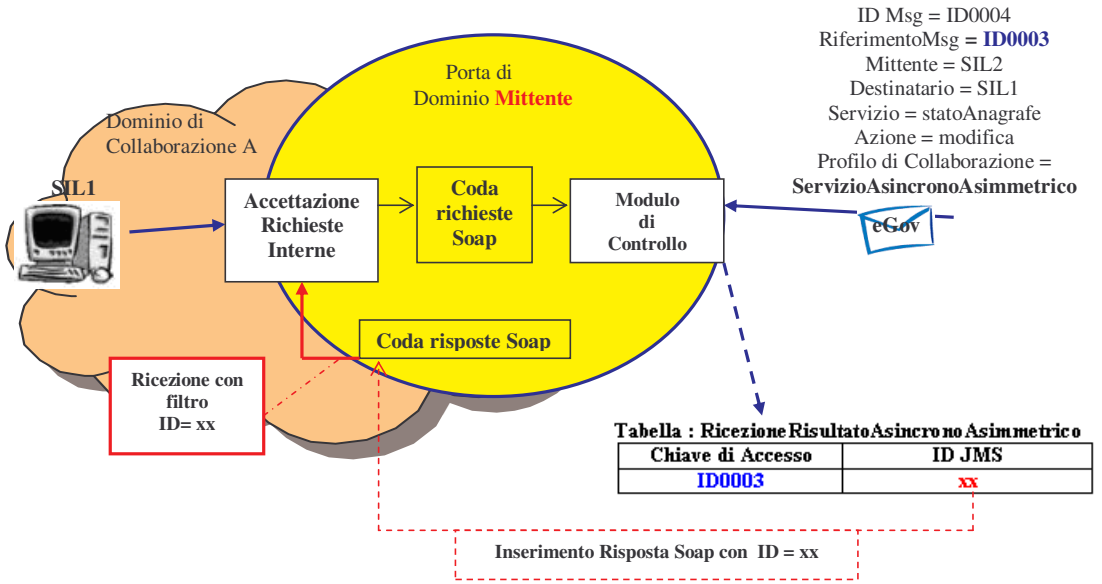


Figura 2-46, Ricezione di una ricevuta con profilo asincrono asimmetrico, relativa ad una richiesta stato precedentemente effettuata

Porta di dominio destinataria.

1° Fase. Il componente 'Profili di Collaborazione', della parte **Sbustamento,** se riceve una richiesta, deve semplicemente generare una ricevuta alla richiesta inserendo un identificativo di ServizioCorrelato, creato appositamente per la risposta, nella ricevuta e-Gov. Inoltre dovrà creare una entry in una tabella apposita 'risposteAsincroneAsimmetriche' con chiave di accesso l'ID architetturale del messaggio prelevato dalla coda RequestIN_eGov ed inserito (con stesso ID) nella coda RequestIN_eGovToSoap, dopo avere effettuato lo sbustamento. Il valore associato all'entry creato possiederà il valore del ServizioCorrelato appositamente creato.

2° Fase. Il componente 'Profili di Collaborazione', della parte **Imbustamento** entra in gioco quando preleva dalla coda ResponseOUT_Soap una busta Soap e l'associato ID architetturale (il cui valore è rimasto lo stesso di quello creato in precedenza dal modulo 'Ricezione buste e-Gov' e ri-associato ad ogni messaggio passato nelle varie code). Il componente utilizza l'ID architetturale per accedere alla tabella 'risposteAsincroneAsimmetriche' e prelevare l'identificativo del servizio correlato creato e salvato nella fase 1 (oltre alle altre varie informazioni e-Gov). Questa informazione (servizio correlato) viene associata alla risposta applicativa e la coppia prodotta viene memorizzata in un repository di risposte asincrone asimmetriche.

3° Fase. Il componente 'Profili di Collaborazione', della parte **Sbustamento,** al momento della ricezione di una busta e-Gov con richiesta stato, utilizza il servizio, presente nella busta e-Gov, per vedere se è presente una risposta nel repository, con quel servizio associato. In caso di assenza della risposta viene costruita ed inviata una busta e-Gov con un messaggio di fallimento. In caso di presenza viene costruita una busta e-Gov contenete la risposta applicativa prelevata dal repository.

Le prime due fasi vengono illustrate nella Figura 2-47, esaminando i componenti coinvolti all'interno dell'architettura di OpenSPCoop. Le frecce blu mostrano la fase 1, di ricezione della richiesta e creazione della ricevuta. Le frecce rosse mostrano la fase 2, dove la risposta applicativa viene depositata in un repository.

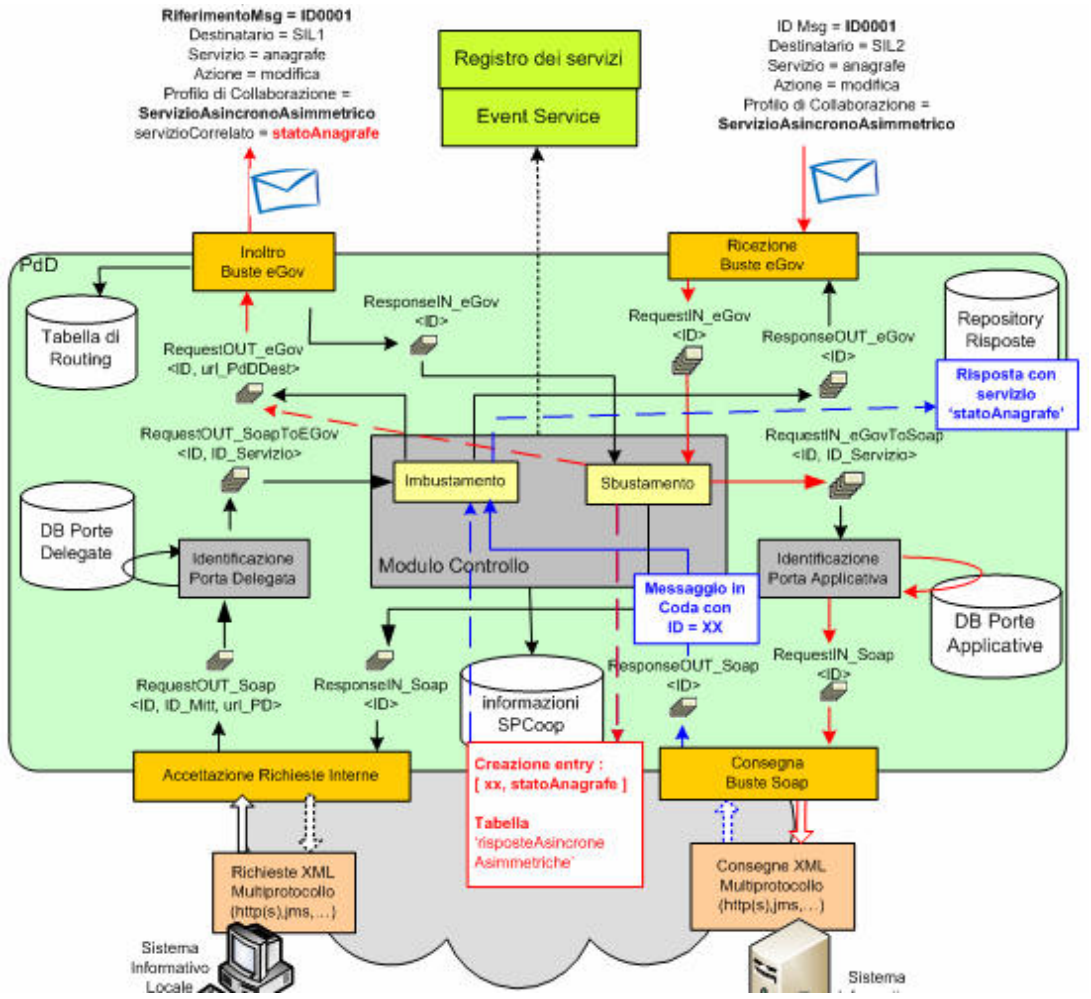


Figura 2-47, Gestione del profilo asincrono asimmetrico all'interno dell'architettura di una porta di dominio che riveste il ruolo 'destinatario'

2.2.5 Il Registro dei Servizi

La specifica CNIPA, come è stato illustrato nel paragrafo 1.3.3, evidenzia il bisogno di un componente architetturale dove sono memorizzati gli accordi di servizio e i diritti di utilizzo del servizio stesso. In particolare, la specifica CNIPA, cerca di mappare l'implementazione di un registro dei servizi attraverso l'utilizzo della tecnologia UDDI.

UDDI (Universal Description, Discovery and Integration) è un meccanismo di registrazione e consultazione di servizi all'interno di un registro. La funzione di registrazione permetterà agli enti regionali di registrarsi e registrare i servizi che intendono erogare (porte applicative), mentre la funzione di consultazione consente ai potenziali fruitori dei servizi di ottenere informazioni su di essi.
In particolare, tramite il registro UDDI è in generale possibile esaminare l'accordo di servizio con cui è possibile:

- registrare enti o altre aziende ed istituzioni;
- registrare il servizio offerto da un punto di vista descrittivo;
- registrare le informazioni necessarie ad invocare il servizio, come per esempio la URL della porta di dominio presso cui è esposto;
- ottenere informazioni su un soggetto che ha pubblicato un servizio;
- ottenere dettagli relativi al tipo di servizio;
- ottenere dettagli tecnici necessari per invocare il servizio (per esempio l'indirizzo del file WSDL).

L'architettura di OpenSPCoop, dovrà disporre di un registro dei servizi che raccoglie tutti gli accordi di servizio. Ogni servizio offerto è memorizzato in un businessService ed è caratterizzato da un profilo collaborativo che identifica il flusso di messaggi di una sua istanza di esecuzione. Nei paragrafi 2.2.4.7 – 2.2.4.9 sono stati ampiamente discussi i profili di collaborazione che saranno associati ad un servizio. Inoltre, nell'accordo, possono essere presenti altre informazioni utili alla gestione del servizio tra porte di dominio, come la collaborazione (Conversazione che racchiude più istanze di servizio), la sequenza (componente utilizzato per ottenere un'ordinamento temporale di più istanze di servizio) e il profilo di trasmissione (componente utilizzato per avere qualità di servizio come affidabilità e scarto di eventuali duplicati). Per ognuna di queste informazioni, illustrate nei paragrafi 2.2.4.4 - 2.2.4.6, è stato creato uno schema di categorizzazione in modo tale da utilizzare gli spazi appositi (categoryBag), all'interno dei businessService, per la catalogazione del servizio.
Un esempio di come sarebbe memorizzato un accordo di servizio associato ad un servizio erogato da un SIL (identificato da un codice AOO '04801700607' e da un codice PA 'm_dg') all'interno del registro e' dato dalla Figura 2-48.

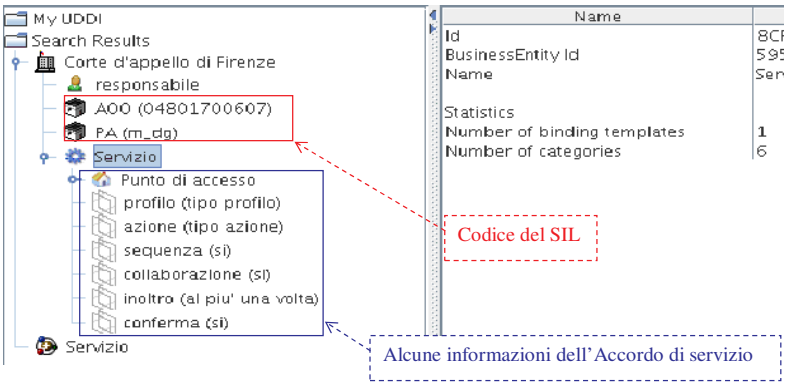


Figura 2-48, Esempio di registrazione di un servizio all'interno di un registro UDDI

In questo esempio abbiamo:

- "Corte d'appello di Firenze" è il dominio di cooperazione dell'esempio
- Il sistema informativo locale viene identificato attraverso:
  - "IPA(m_dg)", identificatore che memorizza un codice IPA;
  - "AOO(04801700607)", identificatore che memorizza un codice AOO;
- "responsabile" è il nome del publisher, colui che ha compiuto la registrazione del servizio;
- "Servizio" è l'identificativo del servizio pubblicato dalla Corte d'appello di Firenze;
- "Punto di accesso" rappresenta l'entry point del servizio;
- "Servizio" è una proprietà dove viene memorizzata la wsdl del servizio;
- Le seguenti funzionalità SPCoop sono raccolte nella catalogazione del servizio dal punto di vista della Cooperazione Applicativa:
  - "Profilo(tipo di profilo)" Il valore 'tipo di profilo' conterrà uno tra:
    - MessaggioSingoloOneWay
    - Sincrono
    - Asincrono Simmetrico
    - Asincrono Asimmetrico
  - "Azione(tipo di azione)" Rappresenta l'identificativo dell'azione effettuata dal servizio (può essere anche più di una);
  - "Sequenza(si)" Questa informazione, possiede un tipo booleano, ed indica l'utilizzo o meno di una consegna in ordine di messaggi SPCoop appartenenti ad una stessa conversazione;

- "Collaborazione(si) " Questa informazione, possiede un tipo booleano, ed indica l'utilizzo o meno di una correlazione tra più istanze del servizio;
- "Inoltro(al più una volta)" Questa informazione, possiede un tipo booleano, ed indica l'utilizzo o meno di un filtro per l'eliminazione di buste ricevute più di una volta;
- "Conferma(si)" Questa informazione, possiede un tipo booleano, ed indica l'utilizzo o meno di una consegna affidabile.

Nell'architettura OpenSPCoop dovrà quindi essere realizzato un registro dei soggetti che, in maniera conforme a quanto richiesto dalla specifiche SPCoop, permetta di identificare domini e SIL appartenenti. Inoltre dovrà schedare i servizi informatici da essi erogati.
Il registro dei servizi di OpenSPCoop, sarà quindi utilizzato dalla porta di Dominio per ottenere informazioni utili alla creazione o alla validazione della busta e-Gov, a partire dai dati identificativi della businessEntity (codice SIL, come ad es. <PA,AOO>), dal nome del servizio che vuole risolvere e l'azione da intraprendere;

Quanto visto risolve i problemi di categorizzazione degli accordi di servizio, ma non discute ancora i problemi riguardanti diritti ed autorizzazioni ad usufruire di un servizio da parte di SIL. Per questo compito, è auspicabile utilizzare un server apposito per questa funzionalità, il Server LDAP, che ospiterà tutte le informazioni relative alle policy di accesso dei Sistemi Informativi Locali ai Servizi (o Eventi) erogati nella Cooperazione Applicativa.
Nello scenario della cooperazione applicativa le policy sono piuttosto complesse, in quanto la possibilità di accesso o meno ad un servizio può anche dipendere dall'operazione da eseguire sul servizio (es. subscribe o publish), dal proxy che lo eroga (es. proxy Generico, anagrafe, …) e dal sistema ospitante della porta di dominio da cui la richiesta effettivamente viene gestita.
Nella soluzione che si intende realizzare, vi è la suddivisione dei soggetti in gioco in servizi, fruitori, e un insieme di entry che definiscono policy di accesso, come evidenziato nella Figura 2-49.

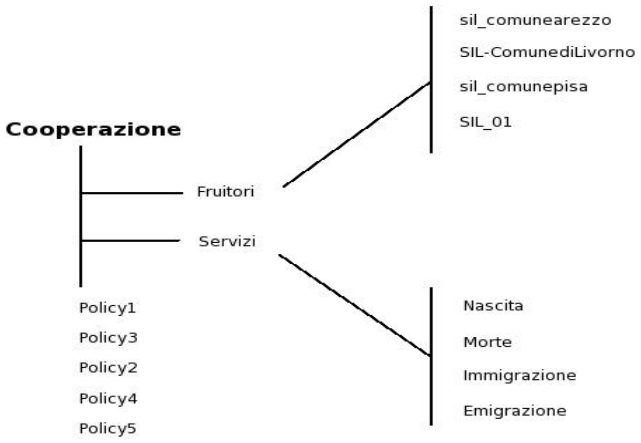


Figura 2-49, Suddivisione degli attori in gioco nella cooperazione applicativa in servizi e fruitori

Le caratteristiche di una policy legano un fruitore a un servizio, e definiscono una regola che indica in che modo il fruitore deve accedere al servizio. Per definire una policy verrà usato un meccanismo messo a disposizione dal server LDAP (es. di tecnologia utilizzabile è Fedora Directory Server) chiamato "ruolo". Un ruolo è un metodo per raggruppare entry all'interno di una directory; i modi in cui le entry vengono raggruppate sono diversi, a partire dall'assegnazione manuale al ruolo, fino alla determinazione dell'appartenenza sulla base di attributi posseduti. Una policy è un ruolo che è in grado di legare due entry, e la sua regola si può memorizzare sotto forma di attributi contenuti dal ruolo stesso. Ad esempio, nella Figura 2-50, la entry “policy4” definisce un ruolo a cui vengono attribuite manualmente le entry “Nascita” e “SIL-ComunediLivorno”.

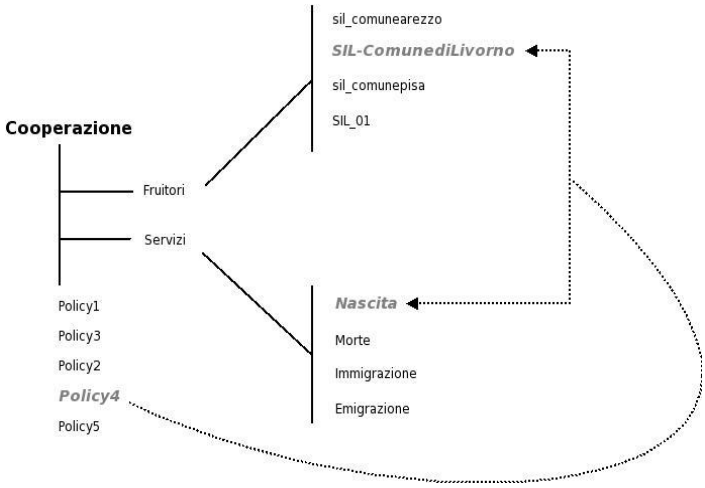


Figura 2-50, Esempio di definizione di una policy che regola la fruizione del servizio ‘Nascita’ utilizzato dal SIL del comune di Livorno

La entry “policy4” raggruppa un servizio ed un fruitore, e la regola che specifica come uno è legato all'altro è un insieme di attributi specificati al momento della definizione del ruolo. Supponiamo che la regola contenuta nella entry di definizione del ruolo sia rappresentata dalla Figura 2-51

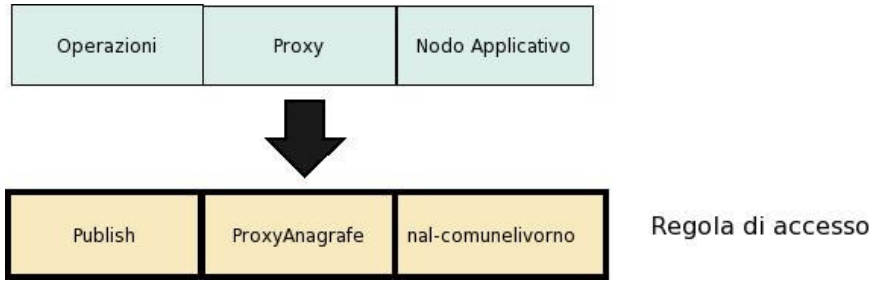


Figura 2-51, Esempio di Ruolo

Questo significa che “SIL_ComunediLivorno” potrà effettuare solo un operazione di tipo “Publish” esclusivamente per il Servizio “Nascita”, a condizione che vi acceda dal sistema informativo locale “nal-comunedilivorno” tramite il proxy “ProxyAnagrafe”.

La verifica avviene a partire dal nome del fruitore e del servizio di cui vuole usufruire (sono noti al momento della richiesta) in due fasi:

- si verifica che le entry che rappresentano il SIL ed il servizio richiesto posseggano il medesimo ruolo: se il ruolo è lo stesso significa che il fruitore ha richiesto un servizio che gli può essere erogato;
- si verifica se le modalità indicate dalla regola coincidono con quelle che il fruitore ha intrapreso/richiesto; la corrispondenza tra queste consente al fruitore di ottenere l'autorizzazione a procedere, il caso contrario implica delle limitazioni sul servizio, dunque revoca dell'autorizzazione.

Vantaggi della soluzione proposta

La directory di gestione delle policy è notevolmente semplice in quanto sia i servizi sia i fruitori sono elencati una volta sola nei rispettivi rami. La complessità di questa soluzione non è nella gestione della directory ma nella corretta definizione di una policy, che è un task molto meno dispendioso ed esposto ad errori.

Integrazione di LDAP con UDDI.

Lo schema di mappaggio delle informazioni UDDI rappresenta a livello logico i servizi, definendo entry di tipo BusinessService. Al di fuori del contesto UDDI, entry definite in questo modo rappresentano dei servizi erogabili. Tali entry dunque, possono essere utilizzate per determinare i diritti di accesso ad un servizio da parte di un fruitore. La directory viene modificata quindi, per includere le informazioni utilizzate dal registro dei servizi UDDI, come mostrato nella figura seguente.

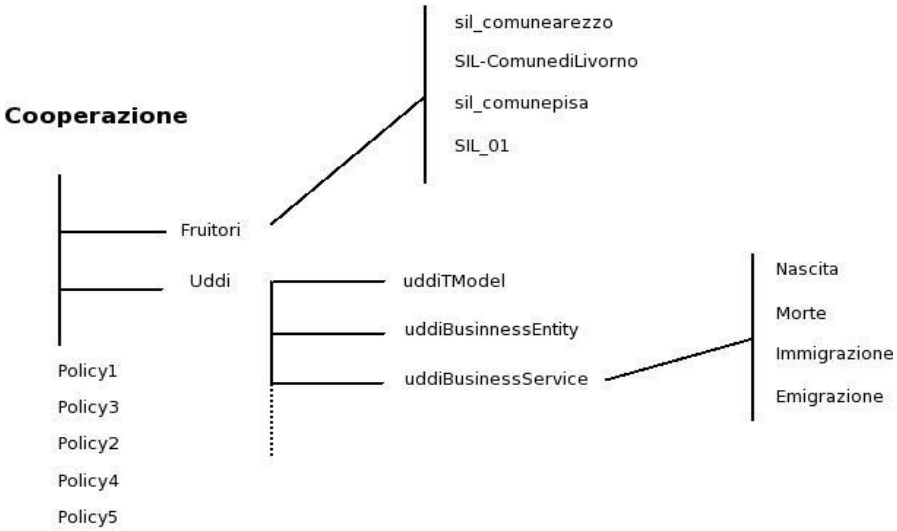


Figura 2-52, BusinessService e Fruitori in LDAP

2.3 Casi d'uso esemplificativi

In questa sezione saranno illustrati e descritti due casi d'uso esemplificativi, cercando di evidenziare la cooperazione dei nodi dell'architettura OpenSPCoop PdD, precedentemente descritti.

2.3.1 Invocazione di un Servizio

In questo scenario esemplificativo dell'architettura OpenSPCoop una porta di dominio invierà una singola richiesta di servizio ad un'altra porta dominio senza restare in attesa di alcuna risposta (profilo di collaborazione impostato a MessaggioSingoloOneWay). La Figura 2-53 illustra l'astrazione di questo particolare caso d'uso.

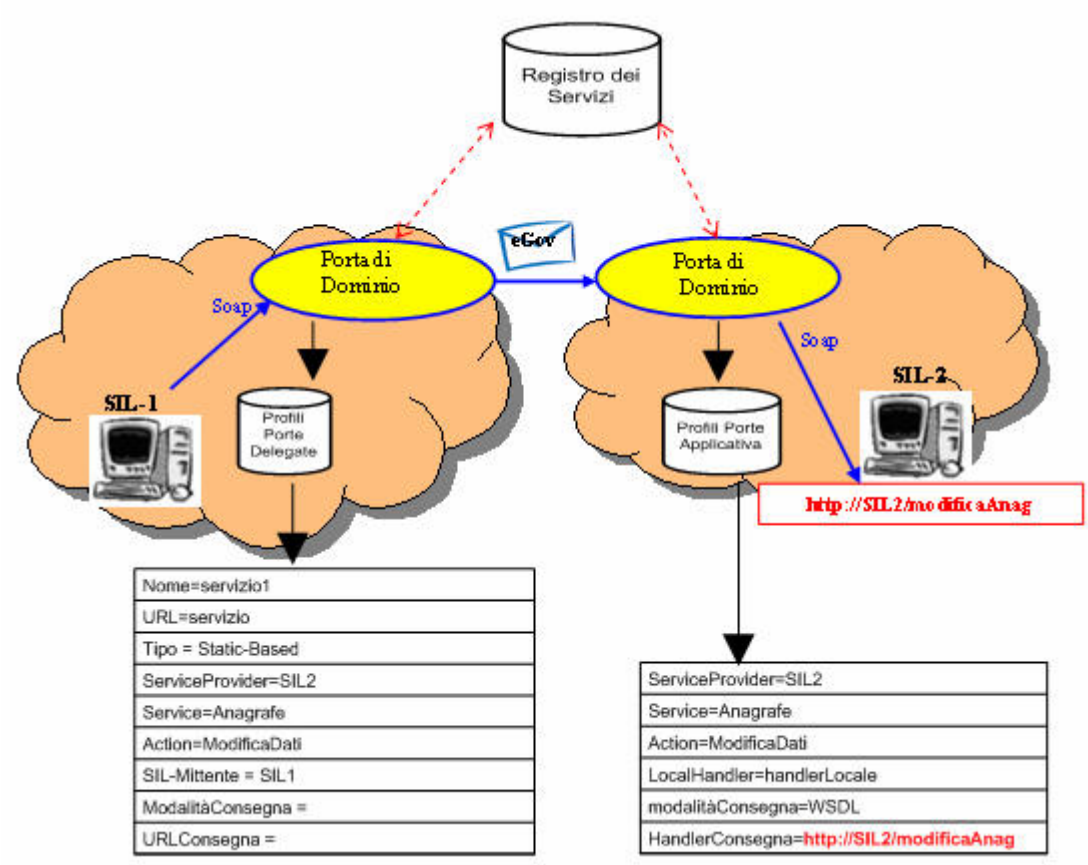


Figura 2-53, Scenario esemplificativo di una invocazione di servizio

La Figura 2-54 illustra la gestione della richiesta effettuata da un sistema informativo locale nell'architettura OpenSPCoop.

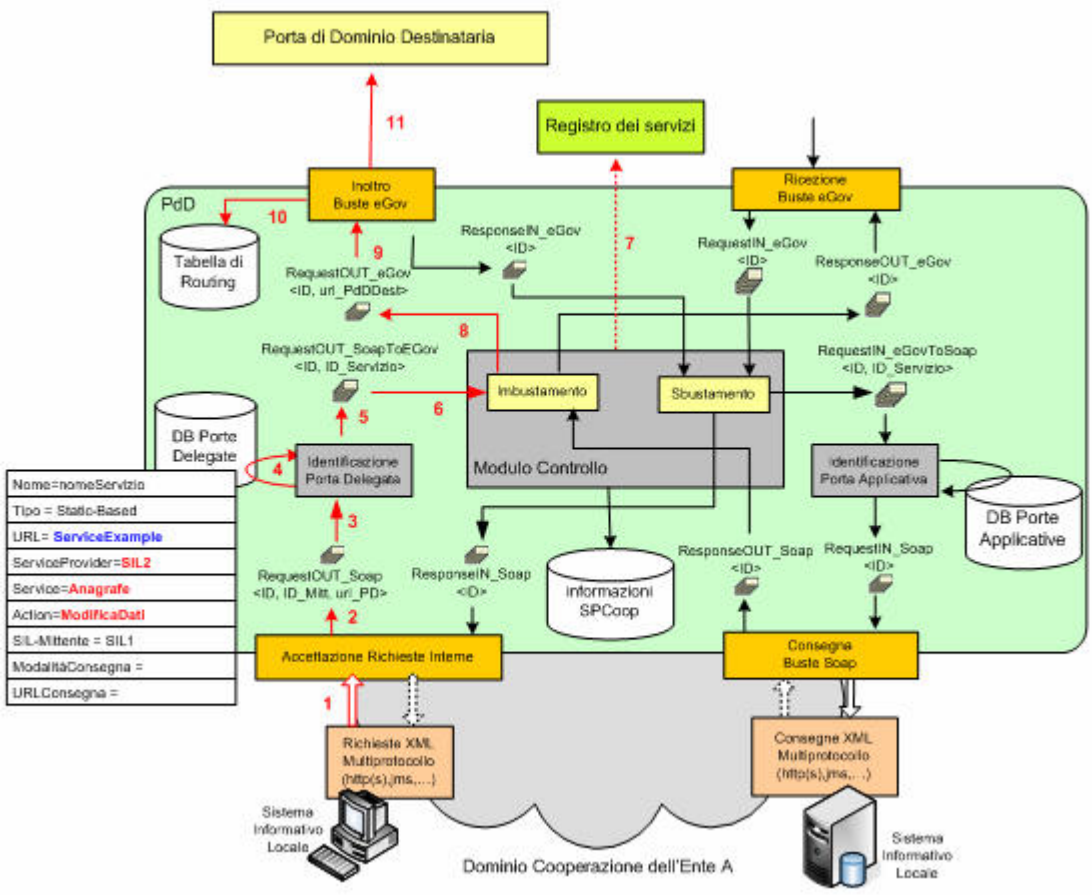


Figura 2-54, Gestione dell'invocazione di Servizio effettuata all'interno dell'architettura di una porta di dominio 'Mittente'

Il SIL1 effettuerà l'invocazione del servizio 'http://openspcoop/**ServiceExample**', semplicemente interagendo con il punto di accesso alla sua porta di dominio.

Step 1. La richiesta del servizio, effettuata dal SIL1, arriva al modulo di **Accettazione Richieste Interne,** come fosse una richiesta ad un servizio Web, e cioè sotto forma di un pacchetto Soap. La gestione della richiesta arrivata, comporta controlli di autenticazione ed autorizzazione a livello di trasporto del mittente.

Step 2. Se la richiesta supera i controlli, il componente si occupa di inserirla nella coda RequestOUT_Soap insieme con la parte finale dell'url invocata dal SIL (ServiceExample) e un identificativo del mittente. A questo punto il SIL1 ed il modulo hanno concluso il loro compito.

Viene a questo punto chiamato in causa il modulo di **Identificazione Porta Delegata.**

Step 3. Il modulo prende il pacchetto dalla coda RequestOUT_Soap condivisa con il modulo precedente e ne preleva il messaggio Soap, e l'identificativo del servizio richiesto (ServiceExample).

Step 4. Viene utilizzato l'identificativo del servizio richiesto per accedere alla tabella locale, che elenca le porte delegate presenti nella porta di dominio, e ottiene le informazioni necessarie per identificare il servizio richiesto nel registro dei servizi (Chiave di accesso data dai valori 'SIL2, Anagrafe, ModificaDati').

Step 5. Il pacchetto Soap viene infine posto nella coda RequestOUT_SoapToEGov condivisa con il modulo di controllo, insieme alle informazioni prelevate nella tabella locale (SIL2, Anagrafe, ModificaDati) ed a informazioni sul mittente.

Viene adesso il turno del **Modulo di Controllo**.

Step 6. Un pacchetto viene prelevato dalla coda RequestOUT_SoapToEGov condivisa con il modulo di controllo. Vengono effettuati controlli di sicurezza, quali ad es. l'autorizzazione del mittente nell'utilizzo della porta delegata richiamata. Inoltre vengono effettuati dei tracciamenti, attraverso la creazione di Log.

Step 7. Vengono prelevate le varie informazioni e-Gov, associate al servizio richiesto, nel registro dei servizi, utilizzando come chiave di accesso la tripla (SIL2,Anagrafe,ModificaDati) prelevata dalle proprietà presenti nel pacchetto. Sono quindi gestite le varie funzionalità e-Gov associate al servizio come per esempio l'affidabilità della consegna e la richiesta di un riscontro. Dal registro dei servizi viene prelevato anche l'url della porta di dominio destinataria a cui è destinata la busta.

Step 8. Una volta creata la busta e-Gov, viene inserita nella coda RequestOUT_eGov insieme all'url della porta di dominio a cui è destinata.

Una volta creata, la spedizione della busta e-Gov alla porta di dominio destinataria viene effettuata dal modulo **Inoltro Buste e-Gov**.

Step 9. Questo componente prende un pacchetto dalla coda RequestOUT_Egov, e ne preleva la busta e-Gov presente e l'url a cui è destinata.

Step 10. La Tabella di routing, in questo contesto potrebbe non venire utilizzata poiché si tratta di una invocazione di servizio verso una url precedentemente letta dal registro dei servizi (step 7). Potrebbe comunque essere forzato il suo utilizzo nel caso si voglia redirigere qualunque busta spedita dalla porta di dominio verso un proxy o relay intermedio.

Step 11. Viene spedita la busta e-Gov alla porta di Dominio Destinataria.

La Figura 2-55 illustra invece i passi effettuati, nell'architettura OpenSPCoop, da una busta SPCoop ricevuta da una porta di dominio che include un service provider che eroga il servizio richiesto nella busta (esiste una porta applicativa apposita).

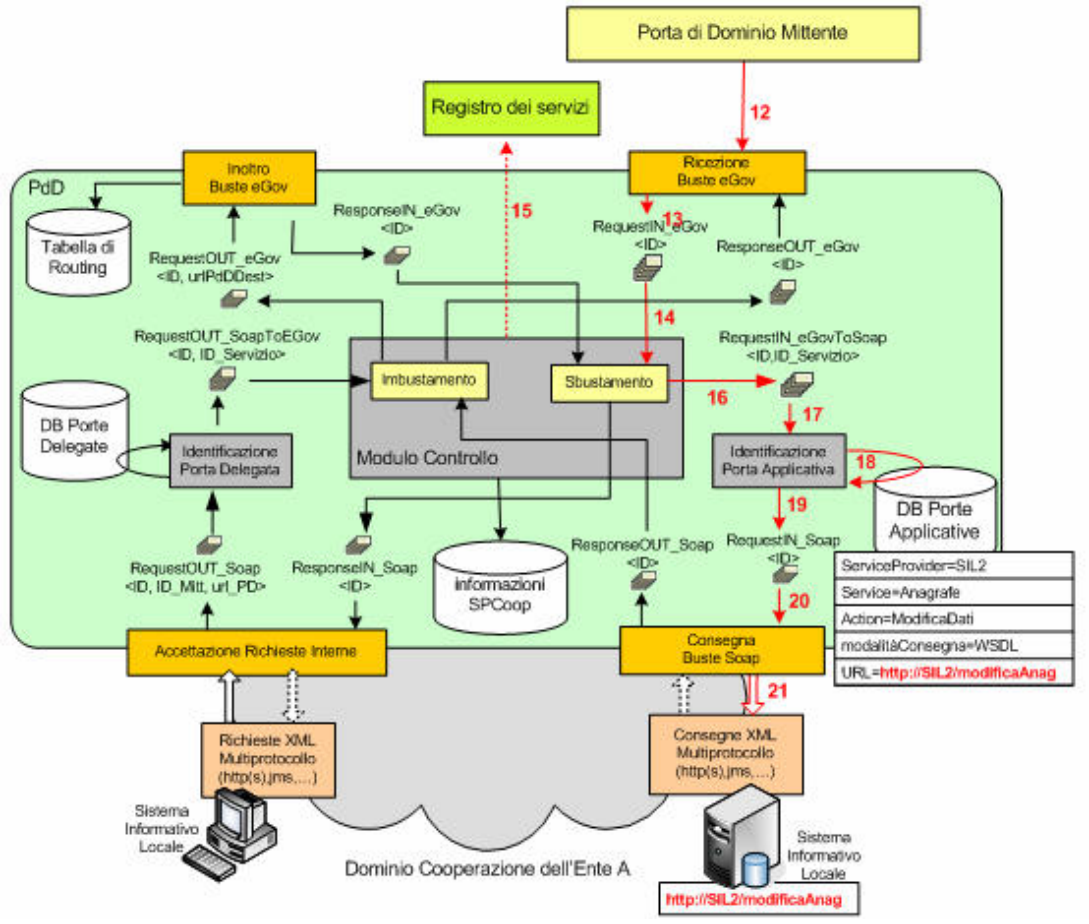


Figura 2-55, Gestione dell'invocazione di Servizio effettuata all'interno dell'architettura di una porta di dominio 'Destinataria'

Step 12. La busta e-Gov arriva al modulo di **Ricezione Buste e-Gov,** come fosse una richiesta ad un servizio Web, e cioè sotto forma di un pacchetto Soap. La gestione della richiesta arrivata, comporta controlli di autenticazione ed autorizzazione a livello di trasporto.

Step 13. Se la richiesta supera i controlli, il componente si occupa di inserirla nella coda RequestIN_eGov. A questo punto il modulo ha concluso il suo compito.

Viene adesso il turno del **Modulo di Controllo**.

Step 14. Un pacchetto viene prelevato dalla coda RequestIN_eGov**.**

Step 15. Vengono prelevate le varie informazioni e-Gov, associate al servizio richiesto, nel registro dei servizi, utilizzando come chiave di accesso la tripla (Destinatario, Servizio, Azione) prelevata dalla busta e-Gov presente nel pacchetto. Viene effettuata la validazione del pacchetto effettuando un controllo di conformità del messaggio con lo standard busta SPCoop e verificando che il contenuto del messaggio sia ben formato e sintatticamente corretto. Vengono inoltre effettuati controlli di sicurezza, quali ad es. l'autorizzazione del mittente nell'utilizzo del servizio richiamato. Inoltre vengono effettuati dei tracciamenti, attraverso la creazione di Log. Dopodichè viene controllato la presenza eventuale di messaggi di Fault nella lista Eccezioni della busta e-Gov. Inoltre sono gestite le varie funzionalità e-Gov associate al servizio come affidabilità della consegna e ricezione di un riscontro. Infine avviene lo sbustamento della busta e-Gov.

Step 16. Una volta sbustata la busta e-Gov, viene inserita nella coda RequestIN_eGovToSoap*,* insieme alla tripla che identifica il servizio applicativo richiesto. Nel nostro caso d'uso la tripla è composta dai seguenti valori: [SIL2, Anagrafe, ModificaDati].

Viene quindi il turno del modulo di **Identificazione Porta Applicativa.**

Step 17. Il modulo prende il pacchetto dalla coda condivisa con il modulo precedente e ne preleva il messaggio Soap, e l'identificativo del servizio richiesto (ServiceProvider, servizio, azione).

Step 18. La tripla viene utilizzata per accedere alla tabella locale, che elenca le porte applicative presenti nella porta di dominio, e ottiene le informazioni necessarie per effettuare l'invocazione del servizio applicativo, e cioè una url di consegna ed una modalità di consegna

Step 19. Il pacchetto Soap viene imbustato in un pacchetto insieme all'url e alla modalità di consegna, e viene messo nella coda RequestIN_Soap.

Infine è il turno del modulo di **Consegna Buste Soap**.

Step 20. Questo componente prende un pacchetto dalla coda RequestIN_Soap, e ne preleva la busta Soap, l'url e la Modalità di Consegna.

Step 21. Siccome la modalità di consegna è WSDL viene invocato il ServizioWeb, relativo al WSDL prelevato attraverso l'url di consegna, consegnando la busta Soap.

La Figura 2-56 illustra tutto il caso d'uso nel suo insieme.

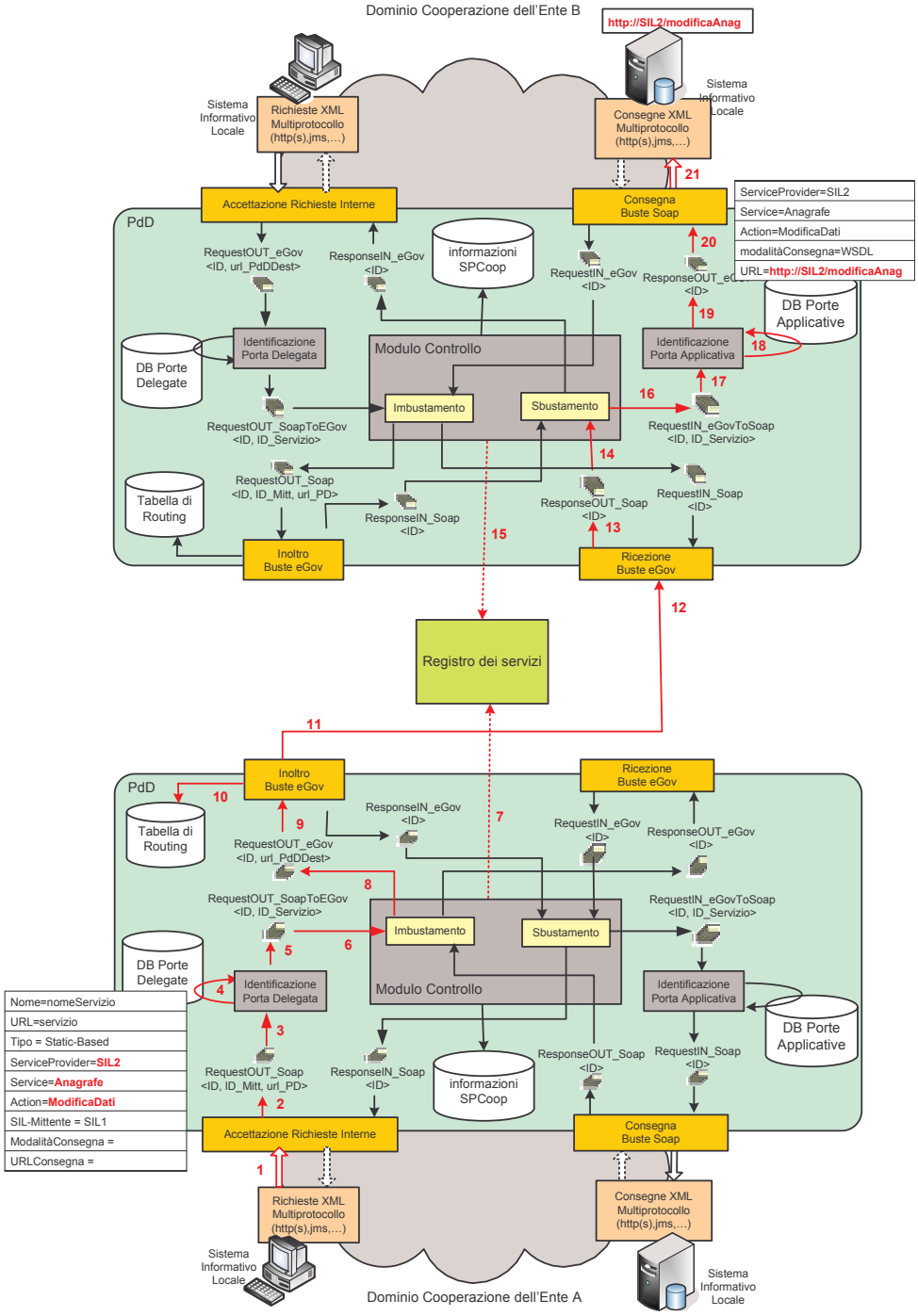


Figura 2-56, Caso d'uso completo di invocazione di un servizio

2.3.1 Pubblicazione / Sottoscrizione di un evento

Questo caso d'uso illustra un esempio di pubblicazione di un evento, da parte di un SIL Pubblicante, e la ricezione di esso da parte di un altro SIL ricevente, che si era precedentemente registrato nel gestore degli eventi. La Figura 2-57 illustra il caso d'uso che verrà descritto in questo paragrafo. Si suppone che un servizio di pubblicazione di uno specifico evento sia stato registrato nella porta di dominio del SIL pubblicante, come porta delegata. Si suppone anche, che il SIL che desidera ricevere tutti gli eventi pubblicati di una specifica categoria, si sia precedentemente registrato nella sua porta di dominio attraverso una porta delegata apposita. (vedi paragrafo 2.2.3).

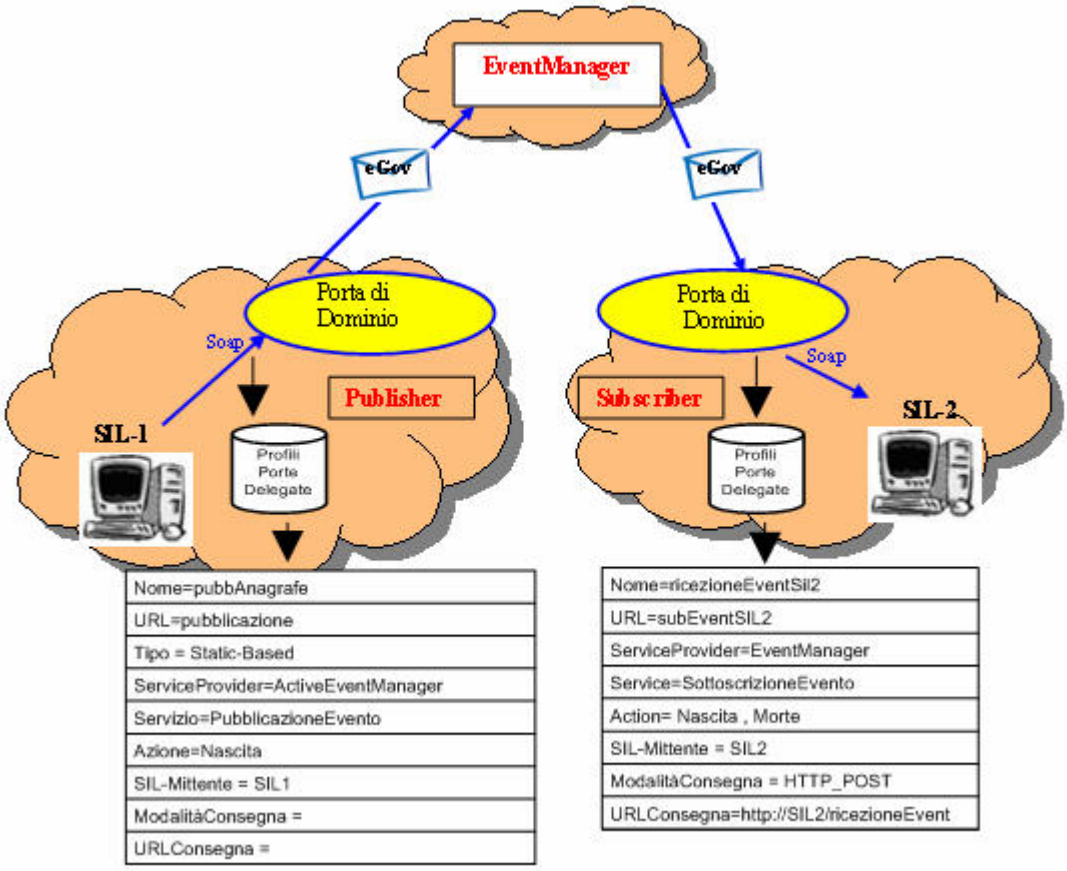


Figura 2-57, Scenario esemplificativo di una pubblicazione e di una sottoscrizione ad un evento

La Figura 2-58 illustra la **fase di pubblicazione di un evento,** descrivendo i passi effettuati nell'architettura OpenSPCoop di una porta di dominio dove è stata registrata una porta delegata apposita per la gestione della pubblicazione.

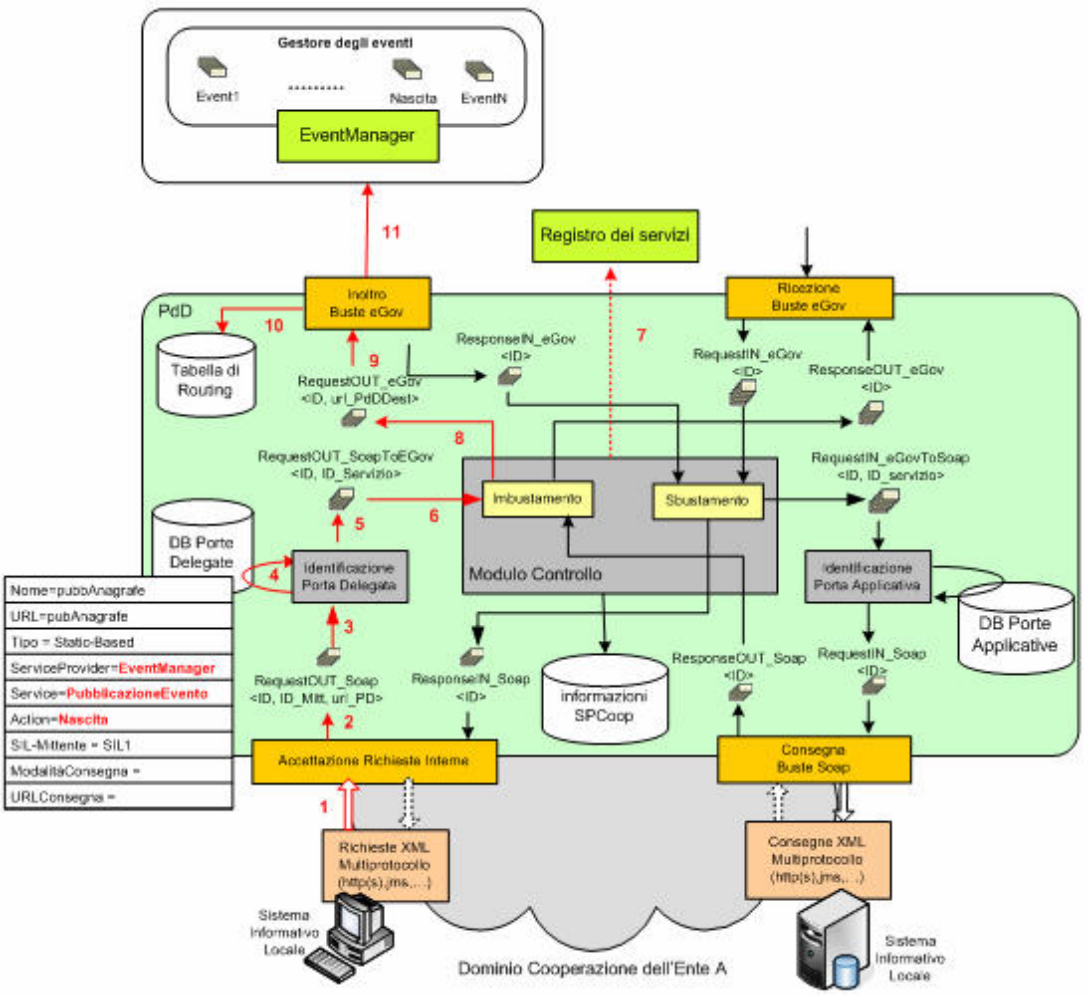


Figura 2-58, Step eseguiti all'interno dell'architettura di una porta di dominio che deve gestire la pubblicazione di un evento

Il SIL1 effettuerà l'invocazione del servizio 'http://openspcoop/**pubAnagrafe**', semplicemente interagendo con il punto di accesso alla sua porta di dominio.

Step 1. La richiesta del servizio, effettuata dal SIL1, arriva al modulo di **Accettazione Richieste Interne,** come fosse una richiesta ad un servizio Web, e cioè sotto forma di un pacchetto Soap. La gestione della richiesta arrivata, comporta controlli di autenticazione ed autorizzazione a livello di trasporto del mittente.

Step 2. Se la richiesta supera i controlli, il componente si occupa di inserirla nella coda RequestOUT_Soap insieme con la parte finale dell'url invocata dal SIL (ServiceExample) e un identificativo del mittente. A questo punto il SIL1 ed il modulo hanno concluso il loro compito.

Viene a questo punto chiamato in causa il modulo di **Identificazione Porta Delegata.**

Step 3. Il modulo prende il pacchetto dalla coda RequestOUT_Soap condivisa con il modulo precedente e ne preleva il messaggio Soap, e l'identificativo del servizio richiesto (pubAnagrafe).

Step 4. Viene utilizzato l'identificativo del servizio richiesto per accedere alla tabella locale, che elenca le porte delegate presenti nella porta di dominio, e ottenere le informazioni necessarie per identificare il servizio richiesto nel registro dei servizi (la chiave di accesso al registro è formata dai valori 'EventManager, PubblicazioneEvento, Nascita').

Step 5. Il pacchetto Soap viene infine posto nella coda RequestOUT_SoapToEGov condivisa con il modulo di controllo, insieme alle informazioni prelevate nella tabella locale (EventManager, PubblicazioneEvento, Nascita) e a informazioni sul mittente.

Viene adesso il turno del **Modulo di Controllo**.

Step 6. Un pacchetto viene prelevato dalla coda condivisa con il modulo 'identificazione porta delegata'.

Step 7. Vengono prelevate le varie informazioni e-Gov, associate al servizio richiesto, nel registro dei servizi, utilizzando come chiave di accesso la tripla (EventManager, PubblicazioneEvento, Nascita) prelevata dalle proprietà presenti nel pacchetto. Non sono però gestite le varie funzionalità e-Gov associate al servizio poiché il modulo si accorge di dover gestire una particolare richiesta verso un gestore degli eventi che richieder una pubblicazione diretta di un evento.

Vengono effettuati controlli di sicurezza, quali ad es. l'autorizzazione del mittente nell'utilizzo della porta delegata richiamata. Inoltre vengono effettuati dei tracciamenti, attraverso la creazione di Log.

Step 8. Una volta creata la busta e-Gov, viene inserita nella coda RequestOUT_eGov.

Una volta creata, la spedizione della busta e-Gov al Gestore degli eventi viene effettuata dal modulo **Inoltro Buste e-Gov**.

Step 9. Questo componente preleva un pacchetto dalla coda RequestOUT_eGov, e ne preleva la busta e-Gov presente.

Step 10. Viene controllato il destinatario della busta, e viene utilizzata la Tabella di Routing per capire a chi consegnare la Busta. In questo caso, la busta è destinata all'EventManager e dalla tabella di routing viene ottenuto l'indirizzo di accesso al Gestore degli Eventi.

Step 11. Viene pubblicata la busta e-Gov nella coda relativa all'evento 'nascita' all'interno dell'EventManager.

La Figura 2-59 illustra la **fase di ricezione di un evento da parte di un sottoscritto,** illustrandone gli step effettuati all'interno dell'architettura OpenSPCoop della porta di dominio del sottoscrittore.

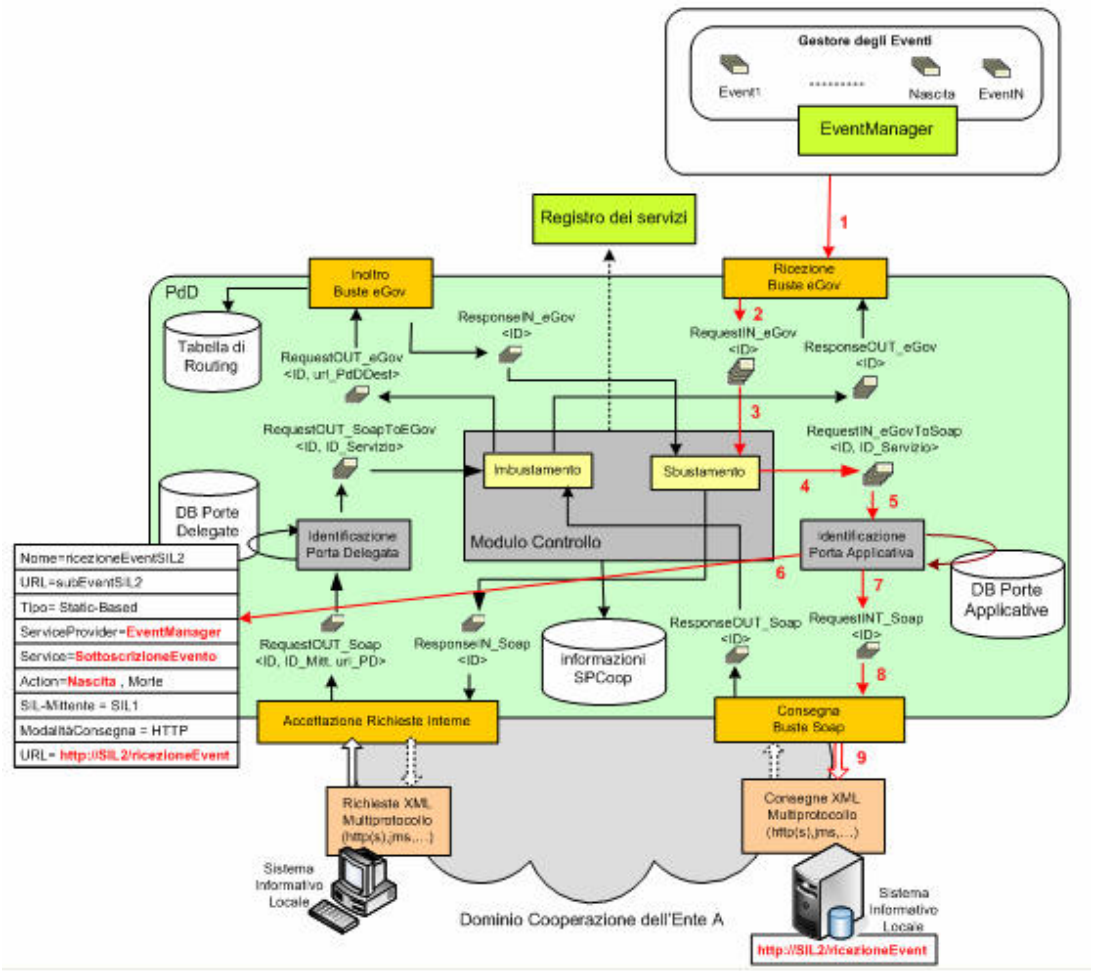


Figura 2-59, Step eseguiti all'interno dell'architettura di una porta di dominio che deve gestire la ricezione di un evento in seguito ad una precedente sottoscrizione

Step 1. La ricezione di eventi, attraverso OpenSPCoop, comporta la creazione di un Listener che si occupa di ricevere tutti i possibili eventi che interessano ai sottoscrittori che si sono registrati all'interno della porta di dominio (vedi paragrafo 2.2.3). In questo esempio, poiché esiste una porta delegata con servizio SottoscrizioneEvento ed azioni Nascita e Morte, assumiamo che OpenSPCoop apra un Listener (gestito dal modulo **Ricezione Buste e-Gov**) sugli eventi Nascita e Morte nel Gestore degli Eventi. Inoltre assumiamo che il SIL2 a cui è associata la porta delegata in questione possa ricevere l'evento attraverso una modalità di interazione PdD-to-SIL passiva, dove OpenSPCoop utilizza l'url di consegna fornita al momento della creazione della porta delegata Infine assumiamo che sia stato pubblicato un evento 'Nascita' e che quindi il Listener lo riceva.
Step 2. Il Modulo **Ricezione Buste e-Gov**, contiene i vari Listener collegati agli eventi nell'EventManager. Al momento della ricezione dell'evento, si occuperà di inserirlo nella coda RequestIN_eGov.

A questo punto entra in gioco **il modulo di controllo.**
Step 3. Il pacchetto viene prelevato dalla coda RequestIN_eGov. Dalle informazioni presenti nella busta e-Gov, il modulo capisce che si tratta di un evento. Riesce a dedurre questo, poiché vede che proviene dall'EventManager. A questo punto viene sbustata la busta e-Gov, e viene inserito l'evento Soap in un pacchetto, insieme alle informazioni prelevate dalla busta e-Gov, quali il ServiceProvider (EventManager) e il tipo di evento (azione).
Step 4. Il pacchetto viene inserito nella coda RequestIN_eGovToSoap condivisa con il modulo 'Identificazione Porta Applicativa'.

Viene a questo punto chiamato in causa il modulo di **Identificazione Porta Applicativa.**
Step 5. Il modulo prende il pacchetto dalla coda condivisa con il modulo precedente e ne preleva il messaggio Soap, e l'identificativo del servizio richiesto (ServiceProvider, servizio, azione).

Step 6. Poiché il ServiceProvider è l'EventManager, capisce che si tratta di un caso di gestione speciale, e invece di accedere come normalmente farebbe alla tabella dei profili delle porte applicative, accede a quella delle porte delegate, cercando l'entry con i seguenti valori:

'ServiceProvider = EventManager'

'Servizio=SottoscrizioneEvento'

'Azione=Nascita'

Visto che una entry esiste, e che sono state specificate una url ed una modalità di consegna, vengono prelevate queste informazioni.

Step 7. Il Soap message viene imbustato in un pacchetto insieme all'url e alla modalità di consegna, e viene messo nella coda RequestIN_Soap.

Infine è il turno del modulo di **Consegna Buste Soap**.

Step 8. Questo componente prende un pacchetto dalla coda RequestIN_Soap, e ne preleva la busta Soap presente, l'url e la Modalità di Consegna.

Step 9. Siccome la modalità di consegna è HTTP viene creata una connessione HTTP all'url, e consegnata la busta Soap (l'evento).

3. Implementazione del prodotto OpenSPCoop

OpenSPCoop è un progetto nato dalla collaborazione tra la società di informatica 'Link.it' di Pisa e il Dipartimento di Informatica dell'Università di Pisa. Gli obiettivi del progetto sono:

1) **Implementazione Open Source** di tutti i componenti, sia periferici che centrali, della specifica SPCoop, con il linguaggio di programmazione Java.

2) Costruzione attorno al progetto di una **comunità** di utenti e sviluppatori esperti della specifica.

In questo capitolo sarà ampiamente discussa l'implementazione di OpenSPCoop realizzata durante il periodo di tesi, tralasciando aspetti infrastrutturali del progetto (necessari a conseguire l'obiettivo 2), discussi nel capitolo 4.

3.1 Architettura Software

La figura seguente illustra l'architettura software del progetto OpenSPCoop.

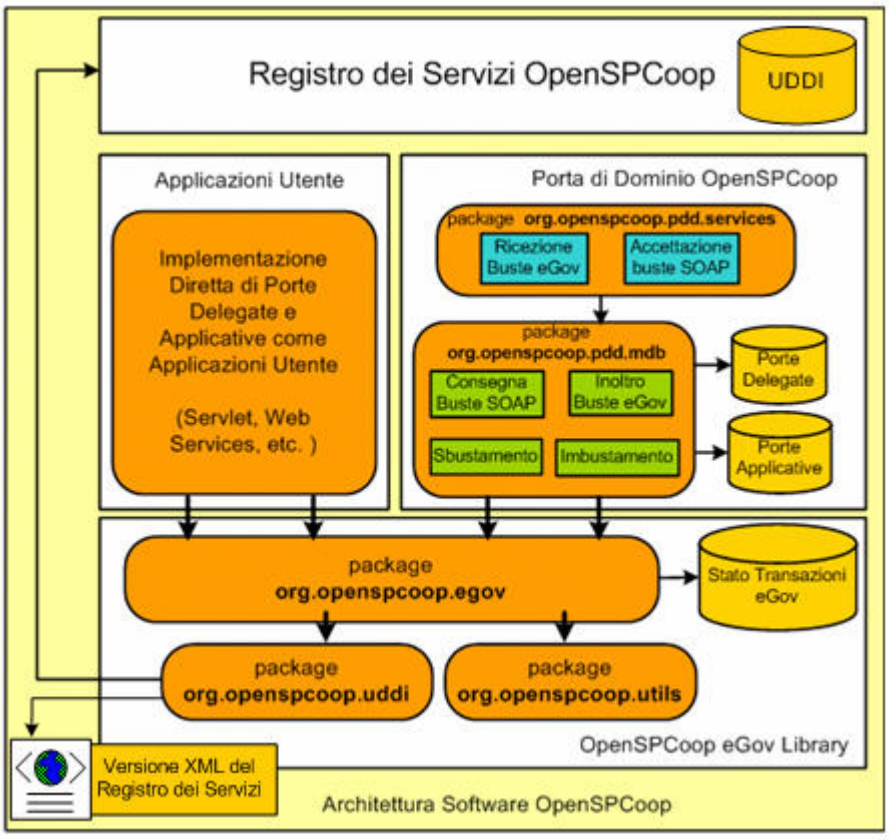


Figura 3-1, Architettura Software del progetto OpenSPCoop

Attualmente il progetto OpenSPCoop è suddiviso in tre sottoprogetti attivi:

1) Libreria e-Gov. Questa libreria si occupa di gestire ogni aspetto della specifica della busta SPCoop. Il progetto in questione permette la realizzazione di buste SPCoop come definite dalla specifica CNIPA. L'implementazione attuale della libreria è ad uno stato di avanzamento di buon livello. Permette di costruire una busta e-Gov in ogni aspetto specificato dal CNIPA. Inoltre permette anche di costruire i messaggi di supporto alla porta di dominio, specificati dal CNIPA, come Messaggi Diagnostici, Messaggi di Tracciamento delle buste SPCoop e Messaggi di Errore Applicativo. Fornisce classi per la gestione di alcuni profili di collaborazione, per la gestione di un History di buste inviate e ricevute e per la gestione della consegna affidabile e senza duplicati.

All'interno della libreria è presente, oltre al package 'org.openspcoop.egov' appena descritto, un package 'org.openspcoop.utils' che permette di manipolare buste soap con e senza attachments. Inoltre è presente anche un package 'org.openspcoop.uddi' che contiene interfacce di supporto all'utilizzo di un registro dei servizi, che è stato implementato nel sottoprogetto apposito.

2) Porta di Dominio OpenSPCoop. La libreria e-Gov, permette di costruire buste SPCoop, ma non include la logica della porta di dominio descritta dalla specifica CNIPA. Un utente che vuole utilizzare l'architettura SPCoop può usufruire della libreria e-Gov per implementarsi 'direttamente' la sua specifica applicazione (porta di dominio 'personale') attraverso ad esempio una servlet. Una implementazione 'personale' della porta di dominio può però non essere vantaggiosa, poiché l'architettura SPCoop, che è stata illustrata nel capitolo 2, non è un contesto semplice e facile da realizzare. Per questo motivo la scelta dell'utente potrebbe indirizzarsi, in alternativa all'uso diretto della libreria e-Gov, ad un utilizzo del sottoprogetto 'Porta di Dominio', che implementa la logica SPCoop. La porta di Dominio OpenSPCoop nasconde la complessità e la difficoltà di implementazione all'utente, che potrà configurare la porta di dominio per le sue necessità semplicemente configurando porte delegate e porte applicative. Naturalmente la porta di dominio, al suo interno, si appoggia alla libreria e-Gov.

3) Il Registro dei Servizi OpenSPCoop. Il sottoprogetto si basa sulla creazione di un registro dei servizi UDDI, che non è parte di questa tesi.

Con l'utilizzo della porta di dominio, si è evidenziata la necessità di un alternativa più semplice al registro UDDI. Questa alternativa è stata implementata in questa tesi (oltre alla libreria e-Gov ed alla porta di dominio) attraverso la realizzazione di un registro dei servizi realizzato come file XML. E' stato definito uno schema del registro ed una libreria che permettesse alla porta di dominio di utilizzarlo (la libreria è stata inglobata all'interno del package 'org.openspcoop.uddi').

3.2 Tecnologie utilizzate

Quanto illustrato nella fase di progettazione del capitolo 2, richiede una tecnologia architetturale che permetta la realizzazione sia di un approccio 'orientato ai servizi', che di un approccio 'basato su eventi'. La tecnologia da utilizzare deve quindi supportare entrambi gli approcci, oltre che fornire altri strumenti di gestione e visionamento dei componenti del progetto OpenSPCoop. A tale scopo, l'utilizzo di un application server Java, basato su tecnologia J2EE, è sembrata la soluzione architetturale migliore in quanto capace di incorporare e gestire:

- Un ambiente di Message Oriented Middleware (MOM), in grado di gestire scambi di messaggi su code persistenti, nel rispetto dello standard Java Message Service (JMS).
- Un ambiente che permetta di implementare i moduli dell'architettura OpenSPCoop come multi-thread, definendoli come particolari Enterprise Java Bean: MDB (Message Driven Bean) che supportino caratteristiche di transazionalità.
- Un ambiente per l'esecuzione di Web Services, conforme allo standard SOAP.
- Un supporto per la comunicazione con un database relazionale.
- Un supporto per la gestione ed il monitoraggio dell'intero sistema.

Per la implementazione del progetto OpenSPCoop, è stato scelto come application server J2EE il prodotto JBoss (descritto nel paragrafo 3.2.1).

All'interno dell'application server sono state utilizzate le seguenti tecnologie, illustrate nel dettaglio nei paragrafi di questa sezione:

- Java Message Service (JMS). Tecnologia utilizzata per lo scambio di messaggi tra le code persistenti create nell'application server.
- Message Driven Bean (MDB). Ha permesso di implementare dei particolari listener su code, multi-thread che supportano caratteristiche di transazionalità.
- Axis 1.3. Rappresenta il motore utilizzato da OpenSPCoop per la manipolazione di messaggi SOAP, e per la creazione di Servizi Web.
- JiBX. Tecnologia utilizzate per la gestione di file e schemi XML, quali il registro dei servizi (versione XML) e il file di configurazione del prodotto OpenSPCoop PdD.
- Ant. Rappresenta lo strumento per la definizione di un ambiente di compilazione del progetto.

3.2.1 JBoss Application Server

L'Application Server JBoss [A21-A23] (attuale versione 4.0.3SP1) è un prodotto già pronto per Java 2 Enterprise Edition (J2EE) [A24] application server. E' il risultato di miglioramenti di versioni precedenti che hanno avuto un grande successo nel mondo dell'open source di Java. Possiede le seguenti caratteristiche:

- Certificato ufficiale rilasciato dagli autori della specifica J2EE 1.4, che dichiara che JBoss è un application server J2EE completo ed utilizzabile.
- Pieno supporto per web services J2EE e Service Oriented Architecture (SOA) (attraverso il sottoprogetto JBoss.Net / JBossWS).
- Supporto per Aspect-Oriented Programming (AOP), che descrive un'architettura necessaria allo sviluppo di soluzioni middleware.
- Integrazione con un Database, Hibernate, che è uno dei framework per la persistenza di oggetti più famoso al mondo. JBoss lo integra nel suo container.
- Clustering e Caching distribuito con una architettura di caching interna al container.

- Implementazione di JMS 1.1 (Java Message Service), attraverso il sottoprogetto **JBoss MQ.**
- Implementazione di JCA (Java Connector Architecture) 1.5, necessaria per definire un supporto transazionale alle applicazioni sviluppate dentro l'application server.
- Implementazione della specifica EJB 2.1 che permette di definire Enterprise Java Bean, in particolare i Message Driven Bean (MDB) utilizzati come vedremo dalla porta di dominio OpenSPCoop.
- Integrazione di un servlet container (per servlet e JSP), quale è Apache Tomcat (**JBoss Web**).

L'architettura dell'application server JBoss [A22] può essere divisa in quattro livelli primari, come viene evidenziato dalla Figura 3-2:

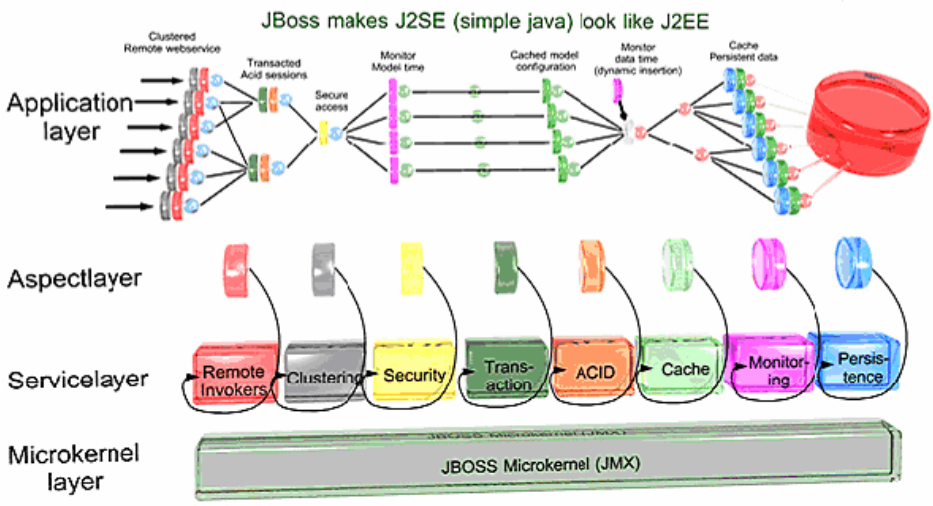


Figura 3-2, Architettura dell'application server JBoss

Microkerner Layer.

Questo livello è il nucleo dell'application server. Si tratta di un modulo realizzato attraverso un'occupazione di risorse ("footprint") estremamente ridotta rispetto al resto dell'architettura. Attraverso l'utilizzo della tecnologia Java JMX (Java Management Extension), il microkernel è una base per gli altri livelli, che offre 'hot deployement' (il deploy di una classe non è effettuato solo staticamente, all'avvio di JBoss, ma anche dinamicamente, mentre JBoss è in esecuzione), loading delle classi avanzato e piena gestione del ciclo di vita delle applicazioni installate nell'application server.

Services Layer.

Sopra al microkernel risiede una architettura orientata ai servizi (SOA). Essa consiste di una serie di servizi, ciascuno dei quali è implementato da un proprio package di classi, ed è pienamente deployable anche a JBoss avviato. I servizi forniti in questo livello possono variare da servizi che gestiscono le transazioni a servizi di messaggistica: ne esiste uno per la gestione della posta, uno per la sicurezza, uno per la gestione di pool di connessioni ecc…. È estremamente facile aggiungere nuovi servizi a questo strato ed allo stesso tempo è possibile eliminare quelli non utilizzati in una particolare implementazione, in modo da ridurre l'occupazione complessiva di risorse.

Aspect Layer.

Questo livello si basa sul modello dell'Aspect-Oriented Programming (AOP) che rende possibile estendere le funzionalità di classi Java in modo semplice e trasparente per il programmatore.

L'Aspect Oriented Programming si basa sul concetto della "separazione delle responsabilità" (ovvero delle funzionalità) fra varie classi Java, che vengono poi estese combinando le funzionalità di ciascuna classe (intercettandone il funzionamento attraversi specifici intercettatori utilizzati da JBoss) e senza necessità di ulteriori modifiche al codice, in modo da ottenere l'applicazione finita richiesta. Questo è particolarmente utile per riutilizzare codice Java già esistente che inizialmente non era stato progettato e sviluppato per essere utilizzato in un application server J2EE.

Application Layer.

Questo livello contiene le applicazioni sviluppate dagli utenti di JBoss (e anche applicazioni pre-installate in JBoss). Queste applicazioni possono sfruttare le capacità offerte dalla vasta struttura dell'application server utilizzando i servizi offerti dal container direttamente (attraverso la programmazione di un codice apposito) o usando il livello superiore (AOP) in aggiunta con specifici tag-driver che addizionano delle funzionalità (comunque implementate dai livelli soprastanti) agli oggetti sviluppati/utilizzati dagli utenti.

3.2.2 Java Message Service (JMS)

JMS è l'acronimo di Java Message Service [A27-A30] ed indica un'insieme di API che permettono lo scambio di messaggi tra applicazioni Java. Prima di descrivere in dettaglio JMS è necessario inquadrare il contesto del suo utilizzo. Con il termine Messaging si definisce un meccanismo che permette la comunicazione asincrona tra client mediante scambio di messaggi. Data questa definizione si può considerare Messaging System un qualsiasi sistema che scambia pacchetti TCP/IP tra programmi client utilizzando un proprio protocollo, ma questo non è lo scopo dei creatori di JMS, che si sono prefissati lo scopo ulteriore di creare un meccanismo di messaging generico, valido per ogni situazione che richieda lo scambio di dati, anche tra applicazioni diverse.

Il sistema che sta alla base di JMS è nominato Message Oriented Middleware (MOM). Mediante l'infrastruttura middleware del MOM processi residenti su macchine diverse possono interagire tra loro mediante l'invio e la ricezione di messaggi tipicamente asincroni. I client interagiscono tra loro, però, non in modo diretto ma tramite la mediazione di un messaging server. Il messaging server riceve i messaggi dai produttori e li recapita ai relativi destinatari (consumatori). Grazie alla mediazione del messaging server i client risultano essere disaccoppiati, cioè non devono sapere nulla l'uno dell'altro. Il **disaccoppiamento** tra il produttore di messaggi e il consumatore di messaggi è una caratteristica molto importante. Nel caso di Remote Procedure Call (come ad esempio RMI in Java) un'applicazione per poter comunicare con un oggetto remoto deve essere a conoscenza dei suoi metodi per poterli invocare. Invece nei sistemi MOM i client non devono interagire direttamente tra di loro: il produttore ed il ricevitore non devono essere a conoscenza l'uno dell'altro ma l'unica cosa che entrambi devono sapere è il formato del messaggio e la "casella di posta" (destination) da usare. La Figura 3-3 evidenzia quanto detto.
Altra importante caratteristica offerta dai sistemi MOM è il concetto di Guaranteed Message Delivery (GMD) che garantisce la consegna del messaggio.

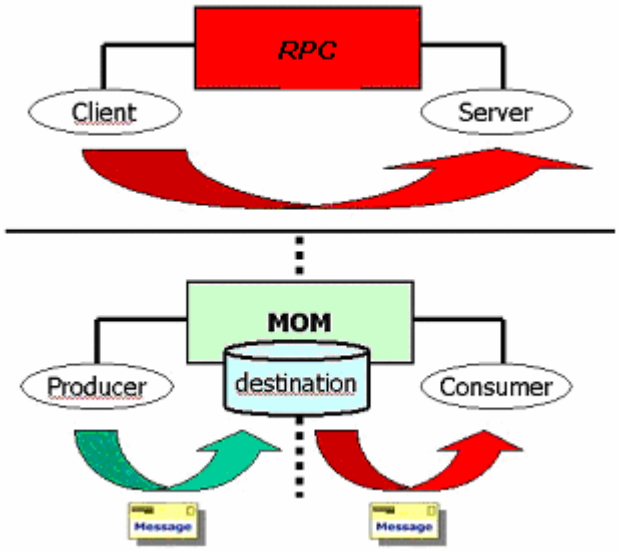


Figura 3-3, Differenza tra un sistema RPC ed un sistema MOM

Nell'architettura MOM sono possibili due tipi di scambio di messaggi [A27]:

- **Point-To-Point.** Un client può mandare uno o più messaggi ad un altro client attraverso una destinazione che prende il nome di Queue (coda). Più produttori e più consumatori possono condividere la stessa coda, ma aspetto importante è che ogni messaggio ha un solo consumatore; questo permette di fatto una comunicazione uno a uno. Non ci sono dipendenze temporali tra sender e receiver del messaggio e quindi il receiver può ricevere il messaggio anche se non era in ascolto sulla coda al momento dell'invio.
- **Publish/Subscribe.** Nel modello Publish/Subscribe (detto anche sistema di messaging event-driven), i client inviano messaggi ad un topic, che si tratta di una coda "speciale" in grado di inoltrare i messaggi a più destinatari. Il sistema si prende carico di distribuire un eventuale messaggio pubblicato da un publisher ai sottoscrittori che si sono dichiarati interessati alla ricezione, avendo attivato una connessione sul topic. Esiste una dipendenza temporale tra publisher e subscriber visto che il receiver può consumare il messaggio solo dopo essersi "sottoscritto" (subscription) al topic d'interesse e dopo che il client ha effettivamente inviato il messaggio. Inoltre il receiver deve mantenere la connessione al topic attiva se vuole continuare a ricevere (a meno di utilizzare Durable Subscriptions). Il messaggio una volta consumato da tutti i subscriber interessati viene tolto dal Topic.

Java Message Service (JMS) è un'insieme di API Java che permette di creare, spedire, ricevere e leggere messaggi. JMS è stato sviluppato da SUN insieme ai maggiori produttori di sistemi MOM per definire un'interfaccia comune indipendente dalla specifica implementazione del sistema di messaging in modo analogo a quanto avviene con JDBC e JNDI. L'applicazione Java JMS risulta così essere indipendente, oltre che dal sistema operativo, anche dalla specifica implementazione del sistema di messaging. In un'applicazione JMS gli attori coinvolti sono riassunti nella Figura 3-4.

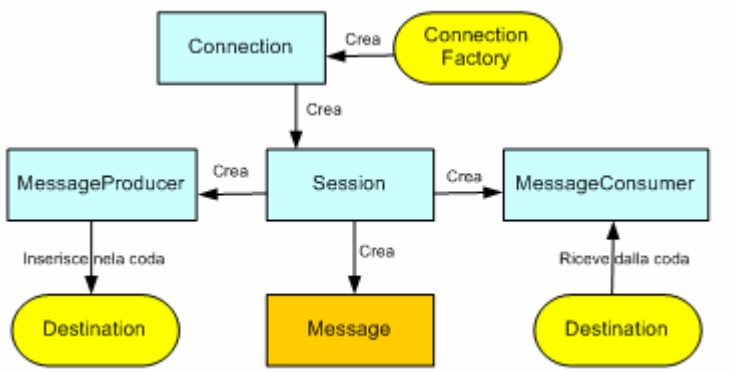


Figura 3-4, Attori coinvolti in un contesto JMS

ConnectionFactory e Destination. La Connection Factory e le destinazioni su cui inserire e ricevere messaggi sono detti Administered Objects poiché sono realizzati attraverso specifiche implementazioni dei Provider JMS. Gli Administered objects sono realizzati ed implementati da diverse aziende e per questo motivo non sono gestiti all'interno dell'interfaccia Java. Le interfacce JMS permettono allo sviluppatore Java di astrarsi dai dettagli implementativi della specifica versione di MOM. La creazione del connection factory e delle destination è compito dell'amministratore JMS che deve inserirli all'interno di un contesto JNDI (associandogli un identificativo) in modo da poterli recuperare da qualsiasi client mediante le normali API JNDI che permettono la ricerca e l'estrazione di oggetti registrati in un server applicativo, a cui è stato associato un identificativo. E' simile ad un server DNS, con l'unica differenza che invece di ritornare un indirizzo IP (frutto di una traduzione dell'identificativo nominale di un host) ritorna un oggetto. Un'applicazione JMS, per inviare messaggi (Sender/Publisher) o per riceverli (Receiver/Subscriber), utilizza i servizi JNDI per ottenere un oggetto di tipo ConnectionFactory e uno o più oggetti destination (Queue o Topic).

Connection. Rappresenta l'astrazione di una connessione attiva su di un particolare JMS provider e si ottiene dall'oggetto Connection Factory.

Session. L'interfaccia Session si ottiene dall'oggetto Connection e rappresenta il canale di comunicazione con una destinazione e permette la creazione sia di produttori che di consumatori di messaggi.

MessageProducer

Il MessageProducer è l'oggetto che permette l'invio dei messaggi verso una particolare destinazione.

MessageConsumer

Il MessageConsumer rappresenta un oggetto in grado di ricevere messaggi.

Messages

I messaggi JMS sono costituiti da tre parti: un'intestazione (message header), una serie di campi contenti proprietà (property fields) ed il corpo (message body). Il **message header** è costituito da campi che sono usati sia dai client che dai provider al fine di identificare ed inoltrare il messaggio. I **property fields** sono coppie di nome e valore e sono utili per costruire "filtri" a livello applicativo utilizzati da client per la ricezione filtrata di messaggi che hanno una determinata caratteristica. Infine nel messaggio è presente un **body** che è specifico a seconda del tipo di messaggio JMS utilizzato**.** I messaggi JMS possono essere di varia natura come evidenzia la Tabella 3-1.

Tipo di Messaggio	Descrizione
TextMessage	Contiene messaggi di tipo String.
ObjectMessage	Contiene oggetti serializzabili.
MapMessage	Contiene coppie nome/valore.
BytesMessage	Contiene uno stream di byte
StreamMessage	Contiene uno stream di valori di tipo primitivo di Java

Tabella 3-1, Tipi di Messaggi JMS

3.2.3 Message Driven Bean (MDB)

Un MDB [A31-A36] è un meccanismo di interazione basato sulla ricezione di messaggi JMS. Si tratta quindi di componenti senza stato, il cui unico compito è quello di processare messaggi JMS provenienti da topic o code. In pratica non è altro che un ascoltatore di una particolare coda o topic, in modo del tutto simile a quello che potrebbe fare un listener JMS scritto ad hoc tramite l'API JMS. Il valore aggiunto che si ha dall'usare un MDB al posto di una normale applicazione client JMS è che in questo caso l'ascoltatore, è nello stesso tempo anche un Enterprise Java Bean [A25-A26]. Gli Enterprise Java Bean sono entità che vivono completamente all'interno di un contesto sicuro, transazionale e tollerante agli errori all'interno di un application server. Questo comporta che l'utilizzatore del MDB può 'spendere' il suo tempo solamente sull'implementazione della logica derivante dalla ricezione di un messaggio (o spedizione), senza pensare ad aspetti quali la ricezione del messaggio stesso (tramite JMS), la gestione di aspetti transazionali ecc… Un altro grosso vantaggio che un utente si trova gratis, senza il minimo sforzo programmativo, è il multi-threading. Un MDB è in grado di processare i messaggi in modo concorrente, ricevendo anche centinaia di messaggi insieme e processandoli tutti in parallelo, dato che il container eseguirà più istanze dello stesso MDB gestite attraverso uno specifico pool.

Il ciclo di vita di un MDB comprende due stadi possibili: 'Non Esistente' e 'Pronto In Pool'. Il significato di entrambi gli stadi è piuttosto intuitivo: nel primo caso il bean non esiste in memoria, mentre nel secondo è stato creato ed è pronto per essere eseguito all'interno di un pool di oggetti, la cui dimensione, determinata dal container, varia in funzione del carico di lavoro.

Nel momento in cui un bean è nel pool, è in grado di processare tutti i messaggi a lui destinati. Il processamento simultaneo avviene grazie alla presenza di più istanze nel pool, dato che ogni istanza può processare un solo messaggio per volta.

Un programmatore, per definire un Message Driven Bean, deve fare implementare ad una sua classe l'interfaccia MessageDrivenBean e l'interfaccia MessageListener. Nei metodi richiesti da queste interfacce, ve ne è uno, 'onMessage()', che sarà richiamato quando un messaggio è ricevuto dall'MDB.

Il programmatore potrà inserire la logica applicativa di gestione del messaggio in questo metodo. Inoltre per la definizione del MDB è necessario associargli dei tag XML, che servono a definire alcune informazioni di configurazione quali il nome del bean, il nome del topic/coda cui deve stare in ascolto, un filtro di selezione dei messaggi JMS, ed altro ancora.

3.2.4 Axis Apache (Soap Engine)

I Web Services [A1, A38, A41, A42] rappresentano una tecnologia consolidata e si trovano in circolazione diverse implementazioni per eseguire deploy delle proprie applicazioni come web service utilizzando i tre mattoni fondamentali, ovvero SOAP, WSDL e UDDI. Una tra le implementazioni più famose, in circolazione, è un prodotto del gruppo Apache: Axis [A37, A39, A40]. E' il motore opensource più famoso per la creazione di WebServices in Java. Axis è un progetto dell'Apache Software Fondation e deriva da SOAP4J, un progetto passato dall'IBM ad Apache. Non è un semplice engine SOAP ma un framework per realizzare sistemi di integrazione basati su SOAP. Il progetto nasce dalla necessità di disporre di una migliore architettura basata su criteri di:

- **flessibilità:** è possibile estendere l'engine per eseguire elaborazioni custom degli header dei messaggi;
- **componibilità**: i gestori di richieste, chiamati Handler in terminologia Axis, possono essere riutilizzati;
- **indipendenza dal meccanismo di trasporto dei messaggi**: in questo modo è possibile utilizzare vari protocolli oltre ad HTTP come SMTP, FTP e messaging.
- **Soap Engine**. Fornisce pieno supporto alle tecnologie dei Web Services, implementando il protocollo Soap, Soap con Attachments, WSDL e api per la gestione di Web Services.

L'engine di Axis permette di implementare servizi web sia lato client (utilizzatori di altri servizi web) sia lato server (fornendo alcuni servizi come web services).

Nel lato server il servizio consiste di un'applicazione Java J2EE, in esecuzione in un "servlet engine" (come ad esempio Apache Tomcat), che viene configurata per ricevere richieste SOAP, processarne i messaggi effettuando un mapping dei dati dal formato pervenuto (XML) ad opportuni oggetti Java e inoltrare l'oggetto ad una specifica applicazione che implementa una interfaccia definita da Axis. Questa applicazione conterrà la logica applicativa del servizio web, che opererà sul messaggio passatogli, e alla fine del proprio lavoro, produrrà un messaggio di ritorno. Il messaggio compie il cammino inverso nell'engine Axis, essendo mappato in dati XML e poi impacchetto in una risposta SOAP che viene restituita al chiamante. Come si può intuire da quanto descritto, l'engine di Axis è basato su di un'architettura 'pipeline' [A39] che permette di processare i messaggi tramite una catena di moduli software come viene illustrato nella Figura 3-5.

Un messaggio arriva (con un protocollo di trasporto specifico) al **listener di trasporto**. Ad esempio potrebbe essere una HTTP servlet. Il lavoro del Listener consisterà nel mappare i dati specifici del protocollo utilizzato in un oggetto Message di axis (org.apache.axis.Message), e di mettere il messaggio in un MessageContext. Il contesto ingloberà anche altre proprietà (es. SoapAction) ed inoltre inserirà nel contesto il nome del protocollo di trasporto (es. http). Una volta che il contesto del messaggio è pronto, il listener redirige il messaggio da gestire all'engine Axis.

L'**engine di Axis** esaminerà innanzitutto il nome del protocollo di trasporto per identificare un oggetto che conterrà una catena di richieste e/o una catena di risposte. Una catena consiste di una sequenza di handlers i quali sono invocati a turno. Se una catena di richieste esiste, essa sarà invocata, passandogli il contesto del messaggio. Questa invocazione provocherà la chiamata di tutti gli handlers specificati nella catena. Dopodichè il motore di Axis cerca una catena di richieste globale (opzionalmente esistente) e se questa catena esiste, sarà utilizzata anch'essa per l'invocazione di ogni suo handler. A questo punto il processo è arrivato all'ultimo stadio, visto che verrà invocato il serviceHandler che contiene un'altra possibile catena e poi il provider con cui è implementata la logica applicativa del servizio.

La logica applicativa del servizio, all'interno del provider produrrà poi una risposta che compierà il cammino inverso.

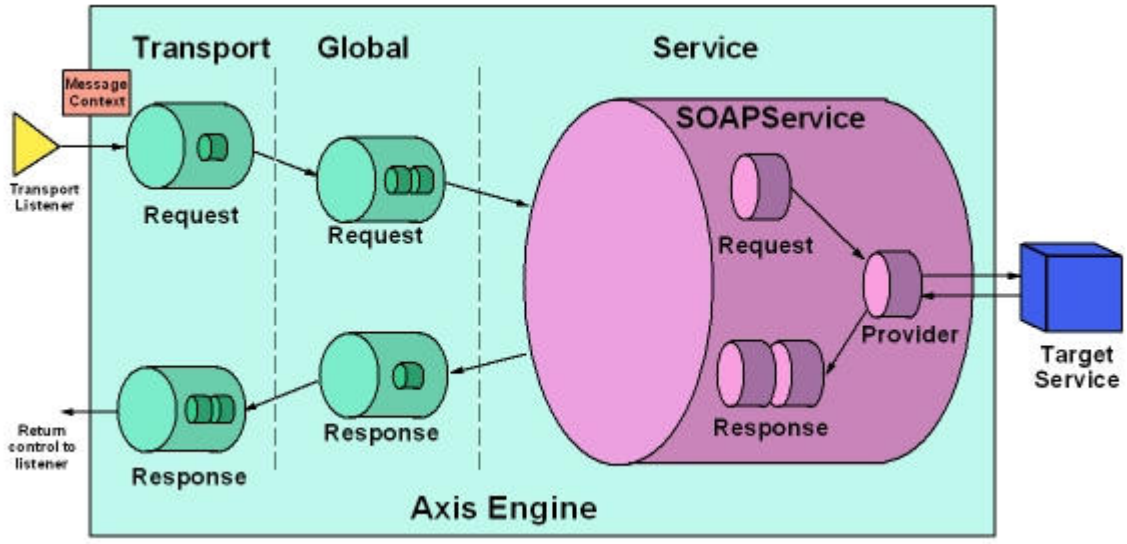


Figura 3-5, Architettura a pipeline del prodotto Apache Axis

I punti chiave di una architettura così definita sono i seguenti:

- **Indipendenza dal trasporto.** Il motore di Axis è indipendente dal trasporto utilizzato. Questo permette di gestire scambio di messaggi SOAP su più protocolli di comunicazione come HTTP(S), JMS, socket TCP/IP, tutti supportati nativamente, ed è aperto per realizzare listener per gestire altri protocolli come SMTP, FTP,
- **Facilità di espansione/modifica del pipeline**. E' possibile definire dei plugin configurabili attraverso dei file di configurazione per realizzare elaborazioni adhoc del messaggio e modificare la catena di elaborazione del messaggio stesso.

3.2.5 JiBX (Binding XML to Java Code)

JiBX [A50] è un framework che permette di legare del codice XML a oggetti Java che sono definiti attraverso classi Java personalizzate e create dall'utente. Gestisce tutti i dettagli per la conversione di dati in e da XML seguendo istruzioni fornite dall'utente. E' stato progettato per effettuare una traduzione tra strutture dati e XML con alta efficienza, e un alto grado di controllo del processo di traduzione.

JiBX utilizza delle definizioni chiamate 'binding definition', i quali sono documenti che definiscono le regole per la conversione da oggetti Java a dati XML e viceversa. Dopo aver compilato le classi Java in file 'class', è possibile eseguire la prima parte del framework JiBX, il **compilatore di binding**. Il compilatore elabora i file binari 'class' prodotti dal compilatore Java, aggiungendoci del codice necessario per l'elaborazione della conversione in istanze di classi di dati xml (e per il processo inverso). Dopo aver eseguito il compilatore di binding, è possibile utilizzare le classi utilizzando il **binding runtime**. Esso effettua il binding di dati XML (che formano l'input) in istanze delle classi generate dal compilatore (processo di unmarshalling) oppure trasforma oggetti Java (che formano stavolta l'input) in documenti di dati XML (processo di marshalling). Questo approccio fornisce i seguenti vantaggi:

- **Flessibilità.** E' possibile utilizzare qualsiasi struttura dati si voglia per la lettura di un file XML, poiché è possibile indicare a JiBX come tradurre essa in e da XML.
- **Performance.** JiBX effettua parsing dei dati XML con alta performance sia nel processo di marshalling che in quello di unmarshalling.
- **Codice chiaro e semplice.** E' possibile scrivere il nostro codice Java, e JiBX lavora con esso.

Inoltre JiBX permette di costruire le classi Java automaticamente, attraverso un generatore XSD, a cui deve essere fornito uno schema XSD che definisce come i file XML da leggere devono essere formati. Questo ulteriore strumento permette di avere in pochi secondi le strutture dati necessarie. Questo aspetto è utile nel caso in cui venga modificato lo schema XSD, poiché in pochi secondi è possibile avere nuove classi compatibili con il nuovo schema. A questo punto l'utilizzo del binding compiler e poi del binding runtime permette di effettuare marshalling e unmarshalling con poco sforzo sia di tempo che di programmazione.

3.2.6 Log4j

La registrazione di eventi che avvengono durante il ciclo di vita di un programma software è un compito tanto comune quanto di notevole importanza, per riuscire a comprendere eventuali malfunzionamenti, operazioni effettuate ecc… in un arco di tempo variabile. Inoltre la registrazione può essere comoda effettuarla su diverse fonti di memorizzazione come file system, database, ecc…
La tecnologia Log4j [A46-A49] permette di effettuare logging di eventi attraverso la chiamata di metodi semplicemente come effettuare una stampa a console video tipo 'System.out.println' o 'printf', offrendo in aggiunta un modo gerarchico per inserire dichiarazioni di logging all'interno di programmi Java. Inoltre la chiamata di un metodo di logging, può provocare la registrazione dell'evento su svariate forme di memorizzazione configurabili attraverso un file di script. Log4J permette inoltre di associare un livello di importanza alla registrazione che si vuole effettuare.

Il vantaggio di utilizzare questo strumento viene evidenziato pensando al seguente scenario. Immaginiamo di avere scritto centinaia di pagine, utilizzando come strumento di debug un qualsiasi metodo primitivo come 'System.out.println'. A questo punto supponiamo che nasca la necessità di salvare ogni informazione stampata a video anche su di un file e su un database. L'unica possibilità che abbiamo è quella di ri-editare tutti i sorgenti Java ed aggiungere il codice necessario. Se invece venisse utilizzato Log4j per la stampa a video, nel nuovo contesto basterebbe modificare il file di configurazione di Log4j aggiungendo degli altri tipi di output oltre alla console video, come ad es. il cammino di un file all'interno del file system e i parametri per la connessione ad una tabella di un database.

L'utilizzo di Log4j coinvolge tre principali entità in gioco:

- Il Logger, che è responsabile per la gestione della maggior parte delle operazioni di log.
- L'Appender, che è responsabile dell'output di una operazione di log.
- Il Layout, che è responsabile della formattazione dell'output di un appender.

Vediamo le tre entità in gioco nel dettaglio.

Logger. Il logger è il cuore del processo di logging. In Log4j ci sono cinque livelli di logging disponibili:

- DEBUG, destinato a informazioni su eventi di grana molto fine destinata al debug di una applicazione.
- INFO, destinato a messaggi informativi che evidenziano il progresso dell'applicazione.
- WARN, destinato a potenziali situazioni pericolose.
- ERROR, destinato a eventi che segnalano errori.
- FATAL, destinato a eventi che causano la morte dell'applicazione (crash).

In aggiunta ci sono altri due speciali livelli di logging disponibili:

- ALL, che non permette mascheramenti dell'evento in caso di filtraggio per importanza. Questo è il livello di importanza maggiore.
- OFF, che permette un mascheramento sicuro dell'evento in caso di filtraggio per importanza. Questo è il livello di importanza minore.

Il comportamento dei logger è gerarchico. La seguente tabella illustra questo fatto, evidenziando nella parte orizzontale il livello di filtraggio dell'output desiderato, associato al Logger utilizzato, e nella parte verticale il livello di logging degli eventi effettuato (associazione di un tipo all'evento registrato):

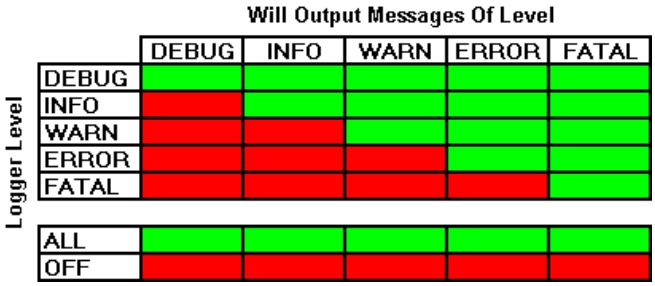


Figura 3-6, Scenario esemplificativo dei filtri su log effettuati

Quindi, una registrazione di un evento produrrà un output solo se il suo tipo è di livello uguale o maggiore a quello associato al logger utilizzato.

Appender. L'appender permette di controllare l'output del logging effettuato da un particolare logger. Gli appender disponibili sotto Log4j sono i seguenti:

- **FileAppender**. Appende eventi ad un file di log

- **ConsoleAppender.** Appende eventi di log al System.out od al System.err usando un layout specificato dall'utente.
- **DailyRollingFileAppender.** Estende il FileAppender così che il file utilizzato cambia, attraverso parametri definito dall'utente, con il passare del tempo (può essere creato un nuovo file ogni giorno, ogni ora ….).
- **RollingFileAppender.** Estende il File appender in modo da cambiare file utilizzato in seguito al raggiungimento di una determinata dimensione del file.
- **WriterAppender.** Appende eventi di log su di un Writer o un OutputStream a seconda della scelta dell'utente.
- **SMTPAppender**. Appende eventi su di una e-mail quando uno specifico evento di logging occorre (impostabile dall'utente).
- **SocketAppender**. Spedisce eventi di log su un remoto server di log.
- **TelnetAppender**. E' un Log4j appender specializzazione nella scrittura su di un read-only socket.
- **JDBCAppender**. E' un appender specializzato nella scrittura dell'evento su di una tabella in un database specificato dall'utente

Un utente può anche implementare il proprio Appender, attraverso la definizione di una classe che implementa l'interfaccia Appender di Log4j. Con questa modalità è possibile fornire qualsiasi modalità di output.

Layout.

Ogni appender deve avere associato un Layout così che possa conoscere come formattare l'output che deve produrre. Ci sono diversi tipi di layout disponibili:

- **HTML Layout**. Formatta l'output come una tabella HTML
- **Pattern Layout**. Formatta l'output basandosi su di un pattern di conversione specificato dall'utente.
- **Simple Layout.** Formatta l'output in una maniera semplice, stampando prima il suo tipo di livello di log, poi una linetta ('-') e poi il messaggio di log.

3.2.7 Ant Apache Compile

Un progetto medio / grosso, come OpenSPCoop, è composto da diverse classi Java che sono dipendenti da altre classi e classi che formano librerie o driver, che sono situate in directory multiple. Per effettuare la compilazione del progetto vi sono due strade alternative. La prima consiste nel compilare tutte le classi a mano, individuando ogni directory dove sono situate librerie e driver, utilizzando tools di compilazione specifici per ogni libreria. Questo processo di compilazione può richiedere diverso tempo, quindi diventa un vero e proprio costo ogni volta che si vuole procedere alla compilazione. Un secondo approccio consiste nell'utilizzare un tool di compilazione unico, che permetta di effettuare tutti i passi necessari per la compilazione del progetto con un unico comando di avviamento, dopo aver configurato un file di configurazione apposito. Questo secondo approccio è permesso dal tool ANT [A43-A45].
Le istruzioni di compilazione, vengono fornite al tool ANT sotto forma di un documento XML. Un esempio semplicissimo di istruzioni che specificano la compilazione di classi Java situate nella directory stessa dove è situato il file 'build.xml' che le contiene, è dato dal seguente codice:

```
<?xml version="1.0"?> ❶
<project name="test" default="compile" basedir="."> ❷

     <property name="src" value="."/> ❸
     <property name="build" value="build"/>

     <target name="init">  ❹
          <mkdir dir="${build}"/>
     </target>

     <target name="compile" depends="init"> ❺
          <!-- Compile the java code -->

          <javac srcdir="${src}" destdir="${build}"/> ❻
     </target>
</project>
```

Vediamo cosa descrive ogni sezione numerata:

1) Dal momento che i files di build sono XML, il documento inizia con una dichiarazione XML che specifica quale versione dell'XML è in uso.

2) L'elemento radice di un build file di ANT è il Project element, che ha tre attributi:
    1. Name. Rappresenta il nome del progetto.
    2. Default. Il target di default da utilizzare quando un target non viene specificato (vedi il resto del paragrafo per ulteriori chiarimenti).
    3. Basedir. La directory base da cui qualsiasi directory relativa utilizzata con ANT viene referenziata.
3) Gli elementi property permettono la dichiarazione di proprietà che sono definibili dall'utente ed utilizzabili all'interno del file di build di ANT. Per referenziare una proprietà chi edita il file di build deve inserire il nome della proprietà all'interno di parentesi graffe (es. ${build}).
4) L'elemento target viene utilizzato come wrapper per una sequenza di azioni. Un target ha un nome, così che esso può essere referenziato da altri, sia esternamente, da linea di comando, sia internamente attraverso la parola chiave 'depends' presente in altri target. Il target nell'esempio, chiamato 'init', ha il compito di creare le cartelle dove dovrà essere posto il risultato della compilazione.
5) Come illustrato nel punto 4, il target 'compile' ha bisogno dell'esecuzione, antecedente al suo utilizzo, del target 'init' (dipendenza da un altro target) che creerà l'ambiente (directory) dove porre il risultato della compilazione (file.class).
6) L'elemento 'javac' è un *task (*così come lo è l'elemento *mkdir*) effettuato nel body di un target. In questo esempio i file sorgenti da compilare sono specificati indicando la directory di appartenenza, attraverso la referenziazione della proprietà 'src'. Discorso analogo vale per la directory dove porre i file.class prodotti, che viene indicata dalla referenziazione della proprietà 'build'.

Il tool ANT, una volta definito il file build di compilazione sarà utilizzabile semplicemente scrivendo a linea di comando: 'ant'. Se inoltre l'utente vuole utilizzare uno specifico target di partenza, potrà utilizzare il comando 'ant nomeTarget'.

Nel sito http://jakarta.apache.org/ant/ è possibile reperire ulteriori informazioni sull'utilizzo di questa tecnologia. ANT fornisce centinaia di diversi task operativi, con svariate funzionalità che vanno dalla gestione del file system all'esecuzione di strumenti di compressione (zip, tar, jar…), dall'esecuzione di comandi Java (java, javac, rmic …) all'esecuzione di comandi esterni. ANT è quindi uno strumento fondamentale per la creazione di un ambiente di sviluppo di un progetto di media / alta dimensione.

3.3 Libreria e-Gov

L'implementazione di una libreria che permettesse la creazione di buste SPCoop compatibili con la specifica CNIPA, è stato il primo step di implementazione del prodotto OpenSPCoop. L'implementazione attuale della libreria è ad uno stato di avanzamento di buon livello. Permette di costruire una busta e-Gov in ogni aspetto specificato dal CNIPA. Inoltre permettere anche di costruire i messaggi di supporto alla porta di dominio, specificati dal CNIPA, come Messaggi Diagnostici, Messaggi di Tracciamento delle buste SPCoop e Messaggi di Errore Applicativo. Fornisce classi per la gestione di alcuni profili di collaborazione, per la gestione di un History di buste inviate e ricevute e per la gestione della consegna affidabile e senza duplicati. Non viene ancora gestita la funzionalità di consegna in ordine.

All'interno della libreria è presente, oltre al package 'org.openspcoop.egov' appena descritto, un package 'org.openspcoop.utils' che permette di manipolare buste soap con e senza attachments. Inoltre è presente anche un package 'org.openspcoop.uddi' che contiene interfacce di supporto all'utilizzo di un registro dei servizi, che è stato implementato nel sottoprogetto apposito.

La Figura 3-7 evidenzia l'architettura software della libreria, illustrando le dipendenze esistenti tra i vari componenti che la formano:

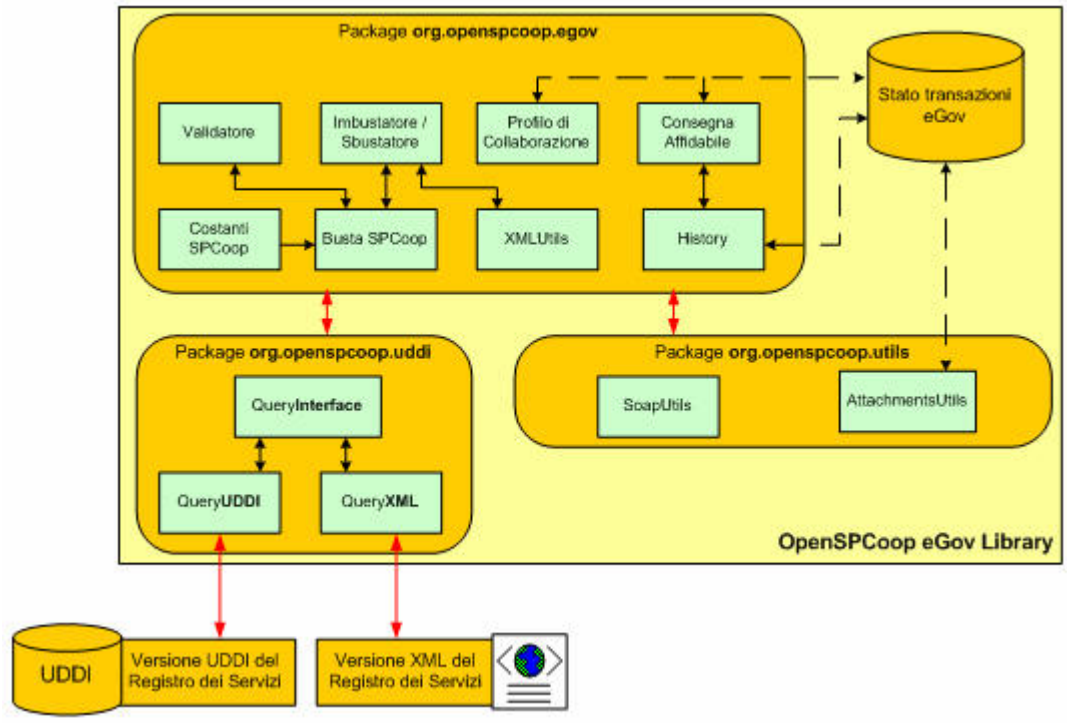


Figura 3-7, Architettura Software della libreria e-Gov

Il package '**org.openspcoop.egov'** è formato dai seguenti componenti di base:

- **CostantiSPCoop.** Questa classe contiene tutti i dati da utilizzare nel contesto della busta SPCoop, specificati dal CNIPA (namaspace, nomi di campi della busta, nomi di errori ecc…)
- **BustaSPCoop.** Si tratta di un insieme di classi che definiscono un oggetto 'Busta' che comprende tutte le informazioni necessarie per la costruzione di un eventuale busta SPCoop. Questo oggetto sarà utilizzato sia dal componente imbustatore, per creare una apposita busta, sia dal componente Validatore, per effettuare una validazione della busta.
- **XMLUtils.** Classe contenente metodi che permettono di definire i messaggi di supporto all'architettura SPCoop, come specificato dal CNIPA. Questi messaggi includono messaggi diagnostici (che dovranno essere interpretati da una console), tracciamenti di buste, e messaggi di errore applicativo da fare recapitare ad una applicazione che invoca un servizio attraverso la propria porta di dominio, nel caso in cui il servizio richiesto non sia effettuabile per un qualsiasi motivo.
- **History.** Classe che permette il salvataggio di buste inviate o ricevute da porte di dominio, attraverso l'interazione con un database apposito. L'interazione avverrà utilizzando l'interfaccia comune di Java, JDBC.

I componenti di base sono poi utilizzati da altri componenti:

- **Validatore.** Componente che effettua la validazione sintattica e semantica di una busta SPCoop.
- **Imbustatore / Sbustatore.** Componente formato da un insieme di classi, il cui compito è l'eventuale costruzione o sbustamento di una busta SPCoop passata come parametro di operazione. Inoltre il componente permette la creazione di messaggi diagnostici, di errore applicativo e di tracciamento.
- **Profilo di Collaborazione.** Componente che si occupa di gestire i profili di collaborazione 'MessaggioSingoloOneWay' e 'Sincrono'.
- **Consegna Affidabile.** Componente che si occupa di implementare i metodi necessari per una gestione dei riscontri, nel contesto in cui le buste SPCoop scambiate tra porte di dominio presentino l'attributo 'confermaRicezione' dell'elemento ProfiloTrasmissione impostato al valore 'true'.

Oltre al package specifico per la gestione della busta SPCoop, nella libreria e-Gov sono forniti altri due package in supporto a 'org.openspcoop.egov'. Un package, '**org.openspcoop.uddi'** include una interfaccia (QueryInterface) che raccoglie i metodi necessari ai vari componenti della libreria per interagire con il registro dei servizi. In particolar modo, il componente 'Imbustatore' avrà necessità di interagire con il registro dei servizi per conoscere le funzionalità SPCoop da associare alla busta (descritte nell'accordo di servizio). Inoltre anche il componente 'Validatore' dovrà interagire con il registro dei servizi per effettuare la validazione dei soggetti e di altre proprietà incluse nella busta.
Sempre all'interno del package 'org.openspcoop.uddi' vengono fornite due classi che implementano l'interfaccia 'QueryInterface':

- **QueryUDDI**. Si occupa di interagire con un registro dei servizi UDDI.
- **QueryXML**. Permetta l'interazione con un registro dei servizi realizzato sotto forma di file XML. Il file che definisce il registro dei servizi, contenente i vari accordi di servizio, e i soggetti che possono effettuare cooperazione applicativa, può essere posto su di una macchina remota o locale.

Infine, il package 'org.openspcoop.utils' contiene due componenti che permettono la manipolazione di SoapMessage:

- **SoapUtils.** Permette la costruzione di messaggi Soap specifici per la cooperazione applicativa SPCoop, in particolare, inglobando nei messaggi specifici SOAP Fault, contenenti descrizioni di possibili eccezioni, come specificato dal CNIPA.
- **AttachmentsUtils.** Modulo che permetta la lettura, gestione, salvataggio e costruzione di messaggi SOAP con Attachments.

Nei prossimi paragrafi sarà illustrato un **esempio di utilizzo della libreria e-Gov.** Dopo aver illustrato alcuni pezzi di codice utili alla creazione/validazione/sbustamento di buste e-Gov conformi alla specifica SPCoop, viene fornito un esempio d'utilizzo della libreria dove viene effettuata una consegna di un servizio e ricezione di una risposta (servizio **Get**).

3.3.1 Imbustamento e Sbustamento di una busta SPCoop

L'utilizzo della libreria verrà illustrato in un contesto in cui non esistono porte di dominio. Nel nostro esempio il SIL1 ha bisogno di un servizio erogato dal SIL2. Gli amministratori dei due SIL possono utilizzare la libreria e-Gov per l'implementazione di due proxy applicativi che contengano la logica di cooperazione applicativa necessaria e mirata solo per i rispettivi Sistemi Informativi Locali. La Figura 3-8 illustra lo scenario citato:

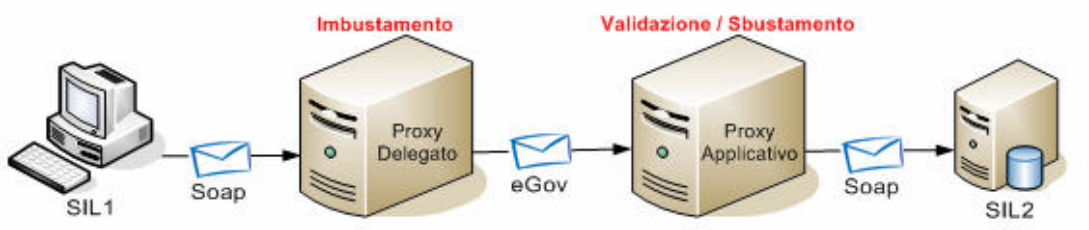


Figura 3-8, Scenario esemplificativo per l'utilizzo della libreria e-Gov

Creazione di una busta e-Gov

La creazione di una busta e-Gov comporta l'aggiunta di un SOAPHeaderElement, nel SOAPHeader di un SOAPEnvelope utilizzato per lo scambio di dati tra due applicazioni (Client e WebService). Ad esempio, supponiamo di avere il seguente messaggio Soap, memorizzato in un file 'Request.xml', che rappresenta la richiesta di un servizio:

```
<soapenv:Envelope
            xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body>
      <ns1:getQuote xmlns:ns1="urn:xmethods-delayed-quotes"
       xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
       xmlns:se="http://schemas.xmlsoap.org/soap/envelope/"
       se:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <symbol xsi:type="xsd:string">IBM</symbol>
      </ns1:getQuote>
</soapenv:Body>
</soapenv:Envelope>
```

Il codice seguente fornisce un esempio di utilizzo della libreria org.openspcoop.egov, dove viene effettuata la creazione di una busta e-Gov.

Dovrà essere importato innanzitutto il package, per un corretto funzionamento del codice:

```
import org.openspcoop.egov.*;
import org.openspcoop.utils.*;
import org.openspcoop.uddi.*;
```

Per effettuare un imbustamento, deve innanzitutto essere creato un oggetto Busta, dove saranno memorizzate le varie informazioni sulle funzionalità e-Gov da associare ad una richiesta SOAP.

```
Busta busta_eGov = new Busta();
```

Dovranno poi essere impostati i campi Mittente, Destinatario, Servizio e Azione.
In questo esempio il SIL1 vuole usufruire del servizio 'Quote', con la specifica azione 'Get' erogato dal SIL2. Queste informazioni devono essere definite precedentemente (al momento della definizione di un servizio, in un registro dei servizi, come ad es. un registro UDDI).

```
// Definizione Porta Delegata:
private static final String tipoServiceProvider = "AOO";
private static final String serviceProvider = "SIL2";
private static final String tipoMittente = "AOO";
private static final String mittente = "SIL1";
private static final String servizio = "Quote";
private static final String azione = "Get";

busta_eGov.setTipoServiceProvider(tipoServiceProvider);
busta_eGov.setServiceProvider(serviceProvider);

busta_eGov.setServizio(servizio);
busta_eGov.setTipoServizio("test");
busta_eGov.setAzione(azione);

busta_eGov.setTipoMittente(tipoMittente);
busta_eGov.setMittente(mittente);
```

Dopodichè deve essere creato un Identificativo e-Gov da associare alla busta. Come da specifica, tra i vari campi posseduti da un identificativo, vi è anche il codice della porta di dominio che si occupa di gestire la richiesta di un servizio. Questo codice deve quindi essere fornito al metodo che permette di costruire l'identificativo, insieme al codice del mittente della richiesta di servizio.

```
private static final String ID_PDD = "PDP-Example";
busta_eGov.setID(Imbustamento.buildID_eGov(ID_PDD,mittente));
```

Deve essere creata anche una data per la registrazione del messaggio.

```
String oraRegistrazione = Imbustamento.getDate_eGovFormat();
busta_eGov.setOraRegistrazione(oraRegistrazione);
```

Rimane da impostare le varie informazioni sulle funzionalità e-Gov da associare alla busta (es. ProfiloCollaborazione, ProfiloTrasmissione, Riscontri ecc…). Queste informazioni possono essere cablate direttamente all'interno di una porta di dominio (proxy Delegato) che si occupa di gestire una particolare richiesta di servizio, oppure possono essere ottenute interagendo con un registro dei servizi. Il codice seguente assume che le informazioni siano cablate all'interno di un proxy delegato.

```
busta_eGov.setProfiloDiCollaborazione(
      Costanti.Profilo_MessaggioSingoloOneWay);
busta_eGov.setInoltro(Costanti.Trasmissione_PiuVolte);
busta_eGov.setConfermaRicezione("False");
```

Viene anche aggiunto un elemento Trasmissione, nella listaTrasmissioni, per registrare nella busta il proxy Delegato che si è occupato di crearla.

```
Trasmissione tras = new Trasmissione();
tras.setOrigine(mittente);
tras.setTipoOrigine(tipoMittente);
tras.setDestinazione(serviceProvider);
tras.setTipoDestinazione(tipoServiceProvider);
tras.setOraRegistrazione(oraRegistrazione);
tras.setTempo(Costanti.LOCAL);
busta_eGov.addTrasmissione(tras);
```

A questo punto l'oggetto Busta è stato correttamente creato ed inizializzato. Bisogna adesso leggere la richiesta SOAP dal file xml 'Request.xml':

```
byte [] soapEnvelope = lettura dal file xml ... ;
```

Procediamo con l'imbustamento.

```
byte [] bytes_busta_eGov =
      Imbustamento.build_eGov(busta_eGov,soapEnvelope);
```

Dopo aver effettuato l'imbustamento, nell'array 'bytes_busta_eGov' si sarà ottenuto la seguente SOAPEnvelope:

```
<soapenv:Envelope
      xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Header>
<eGov_IT:Intestazione
      soapenv:actor="http://www.cnipa.it/eGov_it/portadominio"
      soapenv:mustUnderstand="1"
      xmlns:eGov_IT="http://www.cnipa.it/schemas/2003/eGov_IT/Busta1_0/">
```

```
<eGov_IT:IntestazioneMessaggio>
<eGov_IT:Mittente>
     <eGov_IT:IdentificativoParte  tipo="AOO" >
           SIL1
      </eGov_IT:IdentificativoParte>
</eGov_IT:Mittente>
<eGov_IT:Destinatario>
      <eGov_IT:IdentificativoParte tipo="AOO">
           SIL2
      </eGov_IT:IdentificativoParte>
</eGov_IT:Destinatario>
<eGov_IT:ProfiloCollaborazione>
      EGOV_IT_MessaggioSingoloOneWay
</eGov_IT:ProfiloCollaborazione>
<eGov_IT:Servizio tipo=”test”>Quote</eGov_IT:Servizio>
<eGov_IT:Azione>Ping</eGov_IT:Azione>
<eGov_IT:Messaggio>
      <eGov_IT:Identificatore>
           SIL1_PDP-Example_0000001_2005-10-26_16:48
      </eGov_IT:Identificatore>
      <eGov_IT:OraRegistrazione tempo="EGOV_IT_SPC">
           2005-10-26T16:48:15
      </eGov_IT:OraRegistrazione>
</eGov_IT:Messaggio>
<eGov_IT:ProfiloTrasmissione confermaRicezione="False"
                        inoltro="EGOV_IT_PIUDIUNAVOLTA" />
</eGov_IT:IntestazioneMessaggio>
<eGov_IT:ListaTrasmissioni>
  <eGov_IT:Trasmissione>


      <eGov_IT:Origine>
           <eGov_IT:IdentificativoParte tipo="AOO">
                SIL1
           </eGov_IT:IdentificativoParte>
      </eGov_IT:Origine>
      <eGov_IT:Destinazione>
           <eGov_IT:IdentificativoParte tipo="AOO">
                SIL2
           </eGov_IT:IdentificativoParte>
      </eGov_IT:Destinazione>
      <eGov_IT:OraRegistrazione tempo="EGOV_IT_SPC" >
           2005-10-26T16:48:15
      </eGov_IT:OraRegistrazione>
</eGov_IT:Trasmissione>
</eGov_IT:ListaTrasmissioni>
</eGov_IT:Intestazione>
</soapenv:Header>
<soapenv:Body>
<ns1:getQuote
   soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:ns1="urn:xmethods-delayed-quotes">
   <symbol xmlns=""
           xmlns:se="http://schemas.xmlsoap.org/soap/envelope/"
           xmlns:xsi=http://www.w3.org/1999/XMLSchema-instance
           xsi:type="xsd:string">IBM</symbol>
</ns1:getQuote>
</soapenv:Body>
</soapenv:Envelope>
```

Validazione / Sbustamento di una busta e-Gov

La validazione di una busta e-Gov comporta la validazione del SOAPHeaderElement e-Gov, secondo requisiti definiti nella specifica CNIPA. Lo sbustamento comporterà invece la sua eliminazione dal SOAPHeader, riportando il SOAPEnvelope allo stato primitivo utilizzato per lo scambio di dati tra due applicazioni (Client e WebService).

Supponiamo che una busta e-Gov sia disponibile sotto forma di byte:

```
byte [] busta_eGov = ... ;
```

Bisognerà innanzitutto controllare, che esista realmente una busta e-Gov (header e-Gov nel SOAPHeader).

```
byte [] soapFault =
      Validazione.controlloPresenza_Busta_eGov( busta_eGov );
if (soapFault != null ){
    log.info("Error: il soap message non contiene una busta eGov“);
    return;
}
```

Dopodichè sarà possibile procedere con la validazione. Durante la validazione, sarà anche letto il contenuto della busta, che sarà incluso nell'oggetto, di tipo 'Busta', passato come parametro al metodo di validazione (gli altri parametri del metodo racchiudono aspetti di validazione avanzata non illustrati in questo esempio). Il metodo di validazione, ritorna un 'Vector', contenente eventuali errori di validazione, in caso di validazione senza successo.

```
Busta busta = new Busta();
java.util.Vector errors =
      Validazione.validazioneBusta( busta_eGov, busta,
                                    true, null, “idSessione”,
                                    null, false );
if(errors.size() != 0){
   log.info("Error: SoapMessage con busta e-Gov non valida.");
}
```

Nel caso siano ritornati degli errori di validazione è possibile esaminare gli errori attraverso il seguente codice:

```
for(int i=0; i< errors.size(); i++){
      Eccezione ecc = busta.getEccezione( i );
      log.info( ecc.getContestoCodifica() );
      log.info( ecc.getCodiceEccezione() );
      log.info( ecc.getRilevanza() );
}
```

Mentre con il seguente codice è possibile creare una busta e-Gov Fault, contenente gli errori riscontrati durante la validazione effettuata nel codice precedente:

```
if(errors.size() != 0){
         // Aggiungo problemi riscontrati nella busta.
         while(errors.size() != 0){
              busta_eGov.addEccezione((Eccezione)errors.remove(0));
         }
         // Creo e-GovFault
         byte [] bytes_busta_eGovFault =
            Imbustamento.build_eGov_Fault(busta_eGov, busta );

         // Spedizione al mittente della busta originale
         // la busta contenente le eccezioni riscontrate
         // durante la validazione
         SEND_TO_MITT(bytes_busta_eGovFault);
}
```

Se la validazione ha successo è possibile effettuare l'estrazione dell'header e-Gov con il seguente metodo:

```
byte [] soap = Sbustamento.remove_eGov(busta_eGov);
```

Interazione con il registro dei servizi

Precedentemente, è stato illustrato un esempio riguardante la creazione di una busta. Nell'esempio è stato supposto che le informazioni e-Gov associate ad una richiesta SOAP siano cablate direttamente all'interno del proxy che si occupa di gestire la richiesta di servizio. Se invece le varie funzionalità e-Gov (es. ProfiloCollaborazione, ProfiloTrasmissione, Riscontri ecc….). sono state registrate in un registro dei servizi è possibile utilizzare un oggetto che implementi l'interfaccia **org.openspcoop.uddi.QueryEGov** per prelevarle da esso.

Le informazioni che identificano un servizio univocamente, all'interno del registro UDDI, sono:

[tipoServiceProvider, serviceProvider, servizio, tipoServizio, azione].

Queste devono essere quindi precedentemente impostate in un oggetto di tipo 'ServizioEGov', per poter effettuare la ricerca nel registro.

```
ServizioEGov infoServizio = new ServizioE-ov();
infoServizio.setTipoServiceProvider(tipoServiceProvider);
infoServizio.setServiceProvider(serviceProvider);
infoServizio.setServizio(servizio);
infoServizio.setTipoServizio(tipoServizio);
infoServizio.setAzione(azione);
```

Dopodichè deve essere creato un oggetto specifico per il particolare tipo di registro con cui si intende interagire. L'oggetto può essere implementato utilizzando la classe org.openspcoop.uddi.QueryEGovUDDI, se si intende effettuare ricerche su di un registro dei servizi UDDI, o la classe org.openspcoop.uddi.QueryEGovXML se si intende utilizzare un registro implementato come file XML. Supponiamo di utilizzare il registro in formato XML, che è posto localmente, sui due proxy, nella posizione '/etc/registroServizi.xml'.

```
org.openspcoop.uddi.QueryEGov query =
   new org.openspcoop.uddi.QueryEGovXML("/etc/registroServizi.xml");
```

Adesso è possibile effettuare la ricerca nel registro dei servizi. Se la ricerca ha successo, l'oggetto di tipo 'ServizioEGov', sarà popolato con le varie informazioni e-Gov associate al servizio.

```
boolean serviceExist = query.getInfoServizio(infoServizio);
if(serviceExist != true) {
      log.info("Servizio richiesto non esistente");
      return;
}
busta.setProfiloDiCollaborazione(
      infoServizio.getProfiloDiCollaborazione()[0]);
busta.setInoltro(infoServizio.getInoltro());
if(infoServizio.getConfermaRicezione())
      busta.setConfermaRicezione("True");
else
      busta.setConfermaRicezione("False");
......
```

3.3.2 Uno scenario di utilizzo

Vediamo uno scenario d'uso completo, dove un client vuole spedire un messaggio Soap (contenente dati applicativi) ad un Server. La spedizione è mediata da un proxyDelegato, con cui il Client è costretto a dialogare, e da un proxyApplicativo che deve essere interpellato per poter comunicare con il server. Il server, ricevuti i dati dal cliente, genera una risposta applicativa che dovrà essere restituita al client sempre dietro la mediazione del proxyApplicativo e del proxyDelegato. La Figura 3-9 illustra lo scenario.

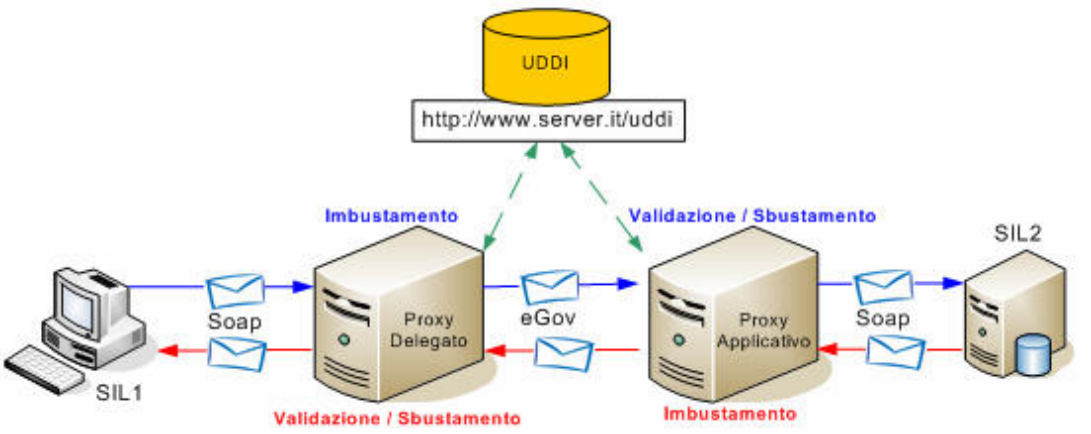


Figura 3-9, Uno scenario completo di utilizzo della libreria e-Gov

L'esempio, di cui è disponibile il codice completo all'indirizzo 'http://www.openspcoop.org/openspcoop/download/esempi-libegov.zip', realizza lo scenario attraverso l'implementazione di due servlet che fungono da proxy delegato e proxy applicativo. Nell'esempio un SIL1 vuole usufruire di un servizio 'Quote' con azione 'Echo' erogato da un SIL2. Il server utilizzato nell'esempio è un webService pubblico della XMethods (http://www.xmethods.net/), disponibile all'indirizzo 'http://64.124.140.30/soap'. Il codice del **servlet proxyDelegato** (che si occupa dell'imbustamento della richiesta, e della validazione/sbustamento della risposta) è il seguente:

```
// Libreria org.openspcoop
import org.openspcoop.egov.*;
import org.openspcoop.utils.*;
import org.openspcoop.uddi.*

public class ProxyDelegatoSincrono extends HttpServlet {

/** Logger utilizzato per riscontro errori di validazione. */
private static Logger log= Logger.getLogger("ProxyDelegato");

// Definizione Proxy Delegato
private static final String IDProxyDelegato = "PD-Sincrono";
private static final String portaApplicativa =
      "http://localhost:8080/PROXY/ProxyApplicativo_Sincrono";

// Definizione Porta Delegata fornita:
private static final String tipoServiceProvider = "AOO";
private static final String serviceProvider = "SIL2";
private static final String tipoMittente = "AOO";
private static final String mittente = "SIL1";
private static final String servizio = "Quote";
private static final String azione = "Echo";
```

```
public void doGet(HttpServletRequest req, HttpServletResponse res)
      throws ServletException, IOException {

      /* ----  Ricezione richiesta del client  ---- */
      ServletInputStream sin = req.getInputStream();
      ByteArrayOutputStream outStr = new ByteArrayOutputStream();
      int read;
      while( (read = sin.read()) != -1)
          outStr.write(read);

      /* ----  Creazione Busta e-Gov ---- */
      Busta busta_eGov = new Busta();
      String oraRegistrazione = Imbustamento.getDate_eGovFormat();
      // Destinatario e Mittente
      busta_eGov.setTipoServiceProvider(tipoServiceProvider);
      busta_eGov.setServiceProvider(serviceProvider);
      busta_eGov.setTipoMittente(tipoMittente);
      busta_eGov.setMittente(mittente);
      // Servizio e Azione
      busta_eGov.setServizio(servizio);
      busta_eGov.setTipoServizio("test");
      busta_eGov.setAzione(azione);
      // Messaggio
      busta_eGov.setID(Imbustamento.buildID_eGov(IDProxyDelegato,
                                                 mittente));
      busta_eGov.setOraRegistrazione(oraRegistrazione);

      // Interazione con il registro UDDI
      ServizioEGov infoServizio = new ServizioEGov();
      infoServizio.setTipoServiceProvider(tipoServiceProvider);
      infoServizio.setServiceProvider(serviceProvider);
      infoServizio.setServizio(servizio);
      infoServizio.setAzione(azione);
      QueryEGov query =
                  new QueryEGovXML("http://www.server.it/uddi");
      boolean serviceExist = query.getInfoServizio(infoServizio);
      if(serviceExist == false){
            // Ritorno operazione non correttamente riuscita
            SoapUtils.build_Soap_Fault(
                  "Servizio non registrato nel Registro UDDI.",null,
                  "Client.ServiceUnknown",null,null,null);
            ServletOutputStream sout = res.getOutputStream();
            sout.write(soapFault);
            return;
      }
      busta_eGov.setProfiloDiCollaborazione(
            infoServizio.getProfiloDiCollaborazione()[0]);
      busta_eGov.setInoltro(infoServizio.getInoltro());
      if(infoServizio.getConfermaRicezione())
            busta_eGov.setConfermaRicezione("True");
        else
            busta_eGov.setConfermaRicezione("False");

      // Tracciamento
      Trasmissione tras = new Trasmissione();
      tras.setOrigine(mittente);
      tras.setTipoOrigine(tipoMittente);
      tras.setDestinazione(serviceProvider);
      tras.setTipoDestinazione(tipoServiceProvider);
      tras.setOraRegistrazione(oraRegistrazione);
      tras.setTempo(Costanti.LOCAL);
      busta_eGov.addTrasmissione(tras);
```

```
      /* ---- Imbustamento ---- */
      byte [] bytes_busta_eGov =
           Imbustamento.build_eGov(busta_eGov,outStr.toByteArray());

      /* -- Inoltro al proxy applicativo e ricezione risposta  -- */
      .......  byte [] risposta =... risposta...

      /* ---- gestione eventuale Risposta con errori ---- */
      if(Validazione.busta_eGov_Presente(risposta) == false){
          // Ritorno operazione non correttamente riuscita
          SoapUtils.build_Soap_Fault(
                              "Trasmissione Fallita",null,
                              "Client.ServiceError",null,null,null);
          ServletOutputStream sout = res.getOutputStream();
          sout.write(soapFault);
          return;
      }

      if(Validazione.containsListaEccezioni(risposta)){
          /* gestione busta e-Gov ritornata contentente eccezioni */
          // Validazione Busta e-Gov
          Busta busta_eGov_Risposta = new Busta();
          java.util.Vector errors =
                  Validazione.validazioneBusta(risposta,
                                  busta_eGov_Risposta,false, null,
                                  "idSessione", null, false );
          if(errors.size() != 0){
               log.error("Busta contenente eccezioni, malformata.");
               return;
          }else{
               // Eventuale Gestione delle eccezioni ritornate
               // TO DO...
               // Ritorno operazione non correttamente riuscita
               SoapUtils.build_Soap_Fault(
                              "Trasmissione Fallita",null,
                              "Client.ServiceError",null,null,null);
               ServletOutputStream sout = res.getOutputStream();
               sout.write(soapFault);
               return;
          }
      }else{
            /* Gestione Risposta Applicativa */
            Busta busta_eGov_Risposta = new Busta();
            java.util.Vector errorsRisposta =
                  Validazione.validazioneBusta(risposta,
                                  busta_eGov_Risposta,true, null,
                                  "idSessione", null, false );
            if(errorsRisposta.size() == 0) {
                  // Ritorno risposta applicativa
                  byte [] soap = Sbustamento.remove_eGov(risposta);
                  ServletOutputStream sout = res.getOutputStream();
                  sout.write(soap);
            }else{
                  // Gestione eventuali errori
            }
      }
}
public void doPost(HttpServletRequest req, HttpServletResponse res)
      throws ServletException, IOException { doGet(req,res); }
}
```

Il codice del **servlet proxyApplicativo** (che si occupa della validazione/sbustamento della richiesta e dell'imbustamento della risposta) è il seguente:

```
// Libreria org.openspcoop
import org.openspcoop.egov.*;
import org.openspcoop.utils.*;
import org.openspcoop.uddi.*;

public class ProxyApplicativoSincrono extends HttpServlet {

/** Logger utilizzato per debug. */
private static Logger log =Logger.getLogger("ProxyApplicativo");

// Definizione Proxy Applicativo
private static final String IDProxyApplicativo = "PA";

// Definizione Porta Applicativa fornita:
private static final String tipoServiceProvider = "AOO";
private static final String serviceProvider = "SIL2";
private static final String servizio = "Quote";
private static final String azione = "Echo";
private static final String urlConsegna =
                      "http://64.124.140.30/soap";

public void doGet(HttpServletRequest req, HttpServletResponse res)
      throws ServletException, IOException {

      /* ----  Ricezione richiesta di un proxy delegato  ---- */
      ServletInputStream sin = req.getInputStream();
      ByteArrayOutputStream outStr = new ByteArrayOutputStream();
      int read;
      while( (read = sin.read()) != -1)
          outStr.write(read);

      /* ---- Controllo presenza di una busta e-Gov ---- */
      byte [] soapFault = Validazione.controlloPresenza_Busta_eGov(
                                          outStr.toByteArray());
      if (soapFault != null){
          // Non è presente una busta e-Gov
          ServletOutputStream sout = res.getOutputStream();
          sout.write(soapFault);
          return;
      }

      /* ---- Validazione Busta e-Gov ---- */
      Busta busta_eGov = new Busta();
      java.util.Vector errors =
            Validazione.validazioneBusta(outStr.toByteArray(),
                                         busta_eGov, true, null,
                                        "idSessione", null, false );
      if(errors.size() != 0){
          // Aggiungo problemi riscontrati nella busta...
          while(errors.size() != 0){
            busta_eGov.addEccezione((Eccezione)errors.remove(0));
          }
```

```
            // Creo e ritorno messaggio SPCoop di errore
            byte [] bytes_busta_eGovFault =
                 Imbustamento.build_eGov_Fault(outStr.toByteArray(),
                                               busta_eGov);
            ServletOutputStream sout = res.getOutputStream();
            sout.write(bytes_busta_eGovFault);
            return;
        }

        /* ---- Sbustamento ---- */
        byte [] soap = Sbustamento.remove_eGov(outStr.toByteArray());

        /* ----  Consegna all'applicazione Destinataria ---- */
        .......  byte [] risposta =... risposta...

        /* ---- Imbustamento risposta applicativa ---- */
        // Inverto Mittente con Destinatario
        String mitt = busta_eGov.getMittente();
        String tipoMitt = busta_eGov.getTipoMittente();
        busta_eGov.setMittente(busta_eGov.getServiceProvider());
        busta_eGov.setTipoMittente(busta_eGov.getTipoServiceProvider());
        busta_eGov.setServiceProvider(mitt);
        busta_eGov.setTipoServiceProvider(tipoMitt);
        // Aggiungo RiferimentoMessaggio
        busta_eGov.setRiferimentoMessaggio(busta_eGov.getID());
        // Creo nuovo identificatore Busta
        busta_eGov.setID(Imbustamento.buildID_eGov(IDProxyApplicativo,
                                         busta_eGov.getMittente()));
        // Creo nuova ora registrazione
        String oraRegistrazione = Imbustamento.getDate_eGovFormat();
        busta_eGov.setOraRegistrazione(oraRegistrazione);
        // Aggiungo Trasmissione alla lista trasmissioni
        Trasmissione tras = new Trasmissione();
        tras.set(...);
        busta_eGov.addTrasmissione(tras);

        // Imbustamento
        byte [] bytes_risposta_eGov =
        Imbustamento.build_eGov(busta_eGov,outResponse.toByteArray());

        /* ---- spedizione risposta applicativa ---- */
        ServletOutputStream sout = res.getOutputStream();
        sout.write(bytes_risposta_eGov);
}

public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        doGet(req,res);
    }
}
```

3.4 Registro dei Servizi

Il sottoprogetto ha come obiettivo la creazione di un registro dei servizi UDDI e la sua implementazione non rientra in questa tesi. Con l'utilizzo della porta di dominio, si è evidenziata anche la necessità di un alternativa più semplice. Questa alternativa è stata implementata in questa tesi (oltre alla libreria e-Gov ed alla porta di dominio) attraverso l'implementazione di un registro dei servizi realizzato come file XML. E' stato definito uno schema del registro [Allegato A] ed una libreria che permettesse alla porta di dominio di utilizzarlo (La libreria è stata inglobata all'interno del progetto 'org.openspcoop.uddi'). Il registro dei servizi è stato implementato attraverso l'utilizzo della tecnologia JiBX, sia per la generazione delle classi che ne formano la struttura, sia per quanto riguarda la lettura dei dati presenti nel file XML effettuata dalla classe 'QueryEGovXML' della libreria e-Gov presente nel package 'org.openspcoop.uddi'.

Vediamo come utilizzare un registro dei servizi 'XML' Il codice seguente illustra un esempio di file xml che definisce un registro dei servizi che include la definizione di due servizi applicativi:

- servizio 'Xmethods' erogato da un sistema informativo locale 'Server', su cui è possibile richiamare le azioni: Rec, Echo e Ping.
- servizio 'ConsegnaJMS' erogato dal solito sistema informativo locale 'Server', su cui è possibile richiamare le azioni: Push e Push_onTopic.

```
<registro_servizi xmlns="http://www.openspcoop.org/uddi/xmlregistry"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation=
   "http://www.openspcoop.org/uddi/xmlregistry registroServizi.xsd">

<porta_applicativa>
      <service_provider tipo="SPC">Server</service_provider>
      <servizio valore="Xmethods">
            <tipo_servizio valore="SPC" />
            </servizio>
      <azione valore="Rec">
            <inoltro_senza_duplicati utilizzo="true" />
            <conferma_ricezione utilizzo="true" />
            <scadenza minuti="5" />
      </azione>
      <azione valore="Echo">
            <profilo_sincrono utilizzo="true" />
            <profilo_oneway utilizzo="false" />
      </azione>
```

```
      <default utilizzo_senza_azione="false">
            <azioni_permesse valore="Ping" />
            <profilo_asincrono_asimmetrico utilizzo="false" />
            <profilo_asincrono_simmetrico utilizzo="false" />
            <profilo_sincrono utilizzo="false" />
            <profilo_oneway utilizzo="true" />
            <collaborazione utilizzo="false" />
            <ordine_consegna utilizzo="false" />
            <inoltro_senza_duplicati utilizzo="false" />
            <conferma_ricezione utilizzo="false" />
      </default>
</porta_applicativa>

<porta_applicativa>
      <service_provider tipo="SPC">Server</service_provider>
      <servizio valore="ConsegnaJMS">
            <tipo_servizio valore="SPC" />
      </servizio>
      <default utilizzo_senza_azione="false">
            <azioni_permesse valore="Push Push_onTopic" />
            <profilo_asincrono_asimmetrico utilizzo="false" />
            <profilo_asincrono_simmetrico utilizzo="false" />
            <profilo_sincrono utilizzo="false" />
            <profilo_oneway utilizzo="true" />
            <collaborazione utilizzo="false" />
            <ordine_consegna utilizzo="false" />
            <inoltro_senza_duplicati utilizzo="false" />
            <conferma_ricezione utilizzo="false" />
      </default>
</porta_applicativa>

<identificativo_parte>
      <service_provider tipo="SPC">
            MittenteDiProva
      </service_provider>
      <id_porta_di_dominio>PisaLinkSPCoopIT</id_porta_di_dominio>
</identificativo_parte>
<identificativo_parte>
      <service_provider tipo="SPC">Server</service_provider>
      <id_porta_di_dominio>PisaLinkSPCoopIT</id_porta_di_dominio>
</identificativo_parte>

<porta_di_dominio>
      <id>PisaLinkSPCoopIT</id>
      <end_point>
            http://localhost:8080/axis/services/OpenSPCoop_PA
      </end_point>
</porta_di_dominio>

</registro_servizi>
```

Esaminiamo in dettaglio gli attori definibili all'interno del registro dei servizi.

Porta Applicativa. Una porta applicativa, all'interno del registro, viene identificata dalla tripla [ServiceProvider, servizio (e tipo), azione], con gli ultimi due parametri opzionali. Ad un servizio è possibile associare una lista di tipi (almeno uno è obbligatorio), e per ogni tipo è possibile specificare anche un end-point diverso (Vedi la spiegazione sull'utilizzo degli 'end-point' seguente, per ulteriori chiarimenti). Ad ogni servizio sono associabili le seguenti funzionalità SPCoop, che rappresenteranno il profilo di Default del servizio (elemento 'Default'):

- ***Azioni permesse*.** Indica le azioni associate ad una porta applicativa.
- ***Profilo di Collaborazione*.** La presenza di un valore 'true'nell'attributo 'utilizzo' dei seguenti elementi, impone l'utilizzo del relativo profilo:
  - <profilo_asincrono_asimmetrico>
  - <profilo_asincrono_simmetrico>
  - <profilo_sincrono>
  - <profilo_oneway>

  Possono essere presenti anche più di un profilo, ma in tal caso l'ordine di utilizzo sarà esattamente quello mostrato. La porta di dominio mittente utilizzerà il profilo con maggiore priorità (asincrono_asimmetrico > asincrono_simmetrico > sincrono > oneway).
- ***Inoltro senza Duplicati***. Se questa funzionalità viene attivata (true), la consegna all'applicazione destinataria sarà effettuata scartando eventuali pacchetti duplicati.
- ***ConfermaRicezione***. Se attivata (true), viene garantita la consegna di una busta da una porta di dominio mittente verso una porta di dominio destinataria.
- ***Collaborazione e Consegna in ordine***. Se entrambe le funzionalità vengono attivate (true), viene instaurata una collaborazione ed una consegna in ordine dei pacchetti inviati da uno stesso mittente (la consegna viene effettuata anche scartando i duplicati e garantendo una consegna affidabile).
- ***Scadenza*.** Parametro opzionale. Indica la validità temporale, in minuti, della busta e-Gov che sarà creata.
- ***EndPoint*.** Parametro opzionale. Vedi la spiegazione sull'utilizzo degli 'end-point' seguente per ulteriori chiarimenti.

Ad una porta applicativa, identificata dalla coppia [ServiceProvider, Servizio(e tipo)], è associabile un profilo e-Gov (insieme di informazioni sulle funzionalità e-Gov) di 'default', utilizzato per richieste di servizio senza azioni (nel caso in cui l'attributo 'utilizzo_senza_azione' dell'elemento default sia impostato a 'true') e per le richieste con le azioni definite nel tag 'azioni_permesse'. Inoltre è possibile associare un profilo diverso da quello di default, per una specifica azione, definendo il profilo specifico, all'interno del tag 'azione'. All'interno del tag possono essere definiti elementi che hanno la medesima semantica e sintassi di quelli definiti all'interno del tag 'default' (tranne che per l'elemento 'azioni_permesse'). Dovranno essere definiti solo gli elementi che descrivono un cambiamento dal profilo di default poiché la presenza di un elemento nell'header 'azione' provocherà il rimpiazzamento del medesimo elemento presente come default, al momento dell'invocazione della specifica azione. Gli elementi che non saranno definiti nel tag 'azione' saranno impostati attraverso la lettura dell'omonimo elemento di default.
L'esempio seguente illustra un esempio di utilizzo di questa sintassi, per una porta applicativa che ha le seguenti caratteristiche:

1. ServiceProvider: SIL1
2. Nome servizio: ExampleService (tipo: 'SPC')
3. Azioni possibili: Registra, Cancella, Aggiorna, Get, GetLimited

Le azioni Registra, Cancella e Aggiorna, necessitano di un profilo 'OneWay'. Mentre l'azione Get necessita di un profilo 'Sincrono', dove però, a differenza dell'azione GetLimited, la richiesta non ha una scadenza temporale. Inoltre tutte le azioni sono effettuate da un servizio disponibile nell'end-point 'http://www.openspcoop.org/Service', tranne che l'azione GetLimited, fornita dal server posto all'indirizzo 'http://www.openspcoop.org/ServiceLimited'.
Supponiamo inoltre che 'ExampleService' sia utilizzabile anche in versione Debug. Nel qual caso qualsiasi richiesta sarà indirizzata all'end-point 'http://www.openspcoop.org/Debug'.

La definizione della porta applicativa sarà la seguente:

```
<porta_applicativa>
      <service_provider tipo="AOO">SIL1</service_provider>
      <servizio valore=" ExampleService ">
            <tipo_servizio valore="SPC" />
            <tipo_servizio valore="Debug">
                  <end_point>
                        http://www.OpenSPCoop.org/Debug
                  </end_point>
                  </tipo_servizio>
            </servizio>
      <azione valore="Get">
            <profilo_sincrono utilizzo="true" />
            <profilo_oneway utilizzo="false" />
      </azione>
      <azione valore="GetLimited ">
            <profilo_sincrono utilizzo="true" />
            <profilo_oneway utilizzo="false" />
            <scadenza minuti="5" />
            <end_point>
                  http://www.OpenSPCoop.org/ServiceLimited
            </end_point>
      </azione>
      <default utilizzo_senza_azione="false">
            <azioni_permesse valore="Registra Cancella Aggiorna" />
            <profilo_asincrono_asimmetrico utilizzo="false" />
            <profilo_asincrono_simmetrico utilizzo="false" />
            <profilo_sincrono utilizzo="false" />
            <profilo_oneway utilizzo="true" />
            <collaborazione utilizzo="false" />
            <ordine_consegna utilizzo="false" />
            <inoltro_senza_duplicati utilizzo="false" />
            <conferma_ricezione utilizzo="false" />
            <end_point>http://www.OpenSPCoop.org/Service</end_point>
      </default>

</porta_applicativa>
```

Sistemi Informativi Locali coinvolti nella cooperazione applicativa.

Nel registro dei servizi OpenSPCoop devono essere registrati tutti gli identificativi degli attori (SIL) che possono essere coinvolti in scambi di buste tra le porte di dominio. Ad ogni identificativo deve essere associato il codice della propria porta di dominio (per permettere ad una porta di dominio mittente di conoscere l'indirizzo di accesso della porta di dominio dell'identificativo destinatario incluso nella busta). Inoltre ad un identificativo, può essere anche associato (opzionalmente) un end-point, vedi la spiegazione sull'utilizzo degli 'end-point' seguente per ulteriori dettagli.

Informazioni sulle porte di dominio coinvolte nella cooperazione applicativa.
Nel registro dei servizi OpenSPCoop devono essere registrate le porte di dominio, associando loro un nominativo, un codice ed un punto di accesso (end-point).

End-Point (punto di accesso di un servizio).
Nei paragrafi precedenti è stata descritta la possibilità di associare un 'end-point' (punto di accesso di un servizio) a cinque entità diverse del registro dei servizi di OpenSPCoop; solo nell'elemento 'porta di dominio' è obbligatorio una sua presenza. Questo permette una maggiore flessibilità ed una compatibilità con porte di dominio diverse da quella di OpenSPCoop.
La scelta dell'end-point da utilizzare prevede una ricerca gerarchica (fino a che la ricerca non ha successo e un end-point viene trovato). Si parte dalla ricerca nel tipo di un servizio richiesto, dopodichè si passa al profilo di una particolare azione specificata in un servizio (profilo diverso da quello di default), passando poi per il profilo di default del servizio stesso. Dopodichè viene cercato nella definizione del service-provider del servizio fino ad arrivare alla definizione della porta di dominio, dove un end-point deve essere obbligatoriamente definito.

Nota. Il registro dei servizi è stato oggetto di molte revisioni, che hanno portato la porta di dominio OpenSPCoop alla interoperabilità con altre porte di dominio esistenti prodotte dalle aziende HP, Oracle e Microsoft. Questa particolarità sarà discussa nel capitolo 4, dove viene descritto il progetto OpenSPCoop discutendo della sua interoperabilità con altre porte di dominio.

3.5 Porta di Dominio

La libreria e-Gov, permette di costruire buste SPCoop, ma non include la logica della porta di dominio descritta dalla specifica CNIPA. Un utente che vuole utilizzare l'architettura SPCoop può usufruire della libreria e-Gov per implementarsi 'direttamente' la sua specifica applicazione attraverso ad esempio una servlet, come viene illustrato nel paragrafo 3.3.2. L'approccio di implementarsi a mano un proxy applicativo (utilizzando la libreria e-Gov) non è un approccio conveniente, poiché, come illustrato nel capitolo 2, la logica SPCoop non è un contesto semplice e facile da realizzare. Per questo motivo la scelta dell'utente potrebbe indirizzarsi, in alternativa all'uso diretto della libreria e-Gov, ad un utilizzo del sottoprogetto 'Porta di Dominio', che implementa la logica SPCoop. La porta di Dominio OpenSPCoop, nasconde la complessità e la difficoltà di implementazione all'utente, che potrà configurare la porta di dominio per le sue necessità semplicemente attivando porte delegate e porte applicative attraverso la configurazione di un file XML. Naturalmente la porta di dominio, al suo interno, si appoggia alla libreria e-Gov. La Figura 3-10 evidenzia l'architettura software della porta di dominio OpenSPCoop.

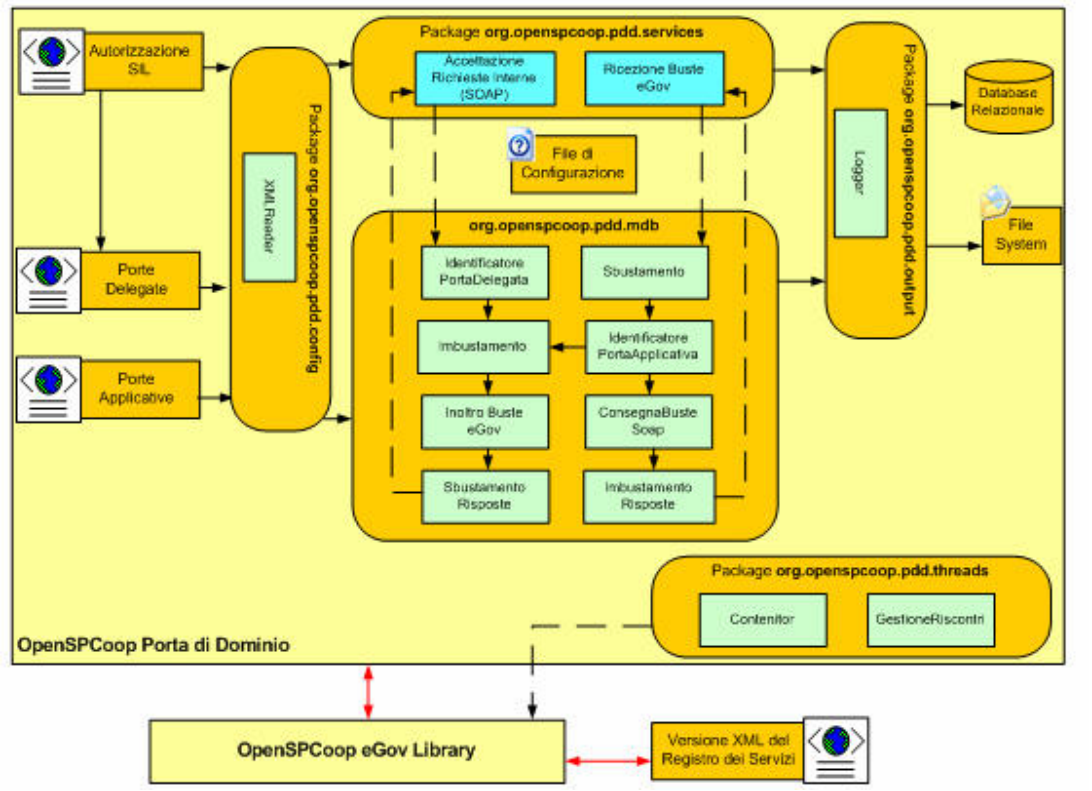


Figura 3-10, Architettura Software della porta di dominio OpenSPCoop

Come si può notare dalla Figura 3-10, la porta di dominio è composta da cinque sotto-package. Vediamo in dettaglio ognuno di essi, iniziando dal package '**org.openspcoop.pdd.services**'. Le classi racchiuse in questo package forniscono l'implementazione di due web services, realizzati tramite tecnologia axis, che forniscono i punti di accesso dall'esterno alla porta di dominio OpenSPCoop. Vediamo le funzionalità offerte dai due web services:

Accettazione Richieste Interne (SOAP). I sistemi informativi locali di un dominio di cooperazione, quando vogliono utilizzare un servizio erogato in un altro dominio, devono accedere al web service implementato da questo componente. Il componente è composto da più classi, una delle quali è un handler axis utilizzato per la raccolta dei parametri impostati nella url di invocazione del web service. In particolar modo, il parametro risiedente dopo la url di invocazione in questione (es. http://localhost:8080/openspcoop/**parametroIndividuazionePortaDelegata**) verrà utilizzato per identificare la porta delegata richiesta dal SIL e quindi per controllare se il SIL ha l'autorizzazione all'utilizzo della porta delegata. I controlli effettuati in questo contesto sono:

- Autenticazione del SIL tramite user e password (autenticazione HTTP-BASED).
- Autorizzazione del SIL ad invocare la porta delegata richiesta.

Le informazioni sui controlli di accesso dei SIL alle porte delegate sono mantenute in un **file XML** definito da OpenSPCoop (vedi 3.5.1), in cui l'amministratore della porta di dominio deve avere configurato appropriatamente i parametri di accesso.
Il componente 'Accettazione Richieste Interne', oltre a verificare l'autorizzazione del SIL ad invocare la porta delegata richiesta, ne prende in carico il contenuto passandolo al componente 'Identificazione Porta Delegata' incluso nel package org.openspcoop.pdd.mdb. Come sarà illustrato nella descrizione del package, questo componente è realizzato tramite tecnologia MDB, e quindi è in attesa di un messaggio su di una coda JMS. Il passaggio è quindi realizzato tramite l'inserimento del contenuto della richiesta (oltre a altre informazioni di gestione quali la porta delegata richiesta dal SIL) in un messaggio JMS posto nella coda di attesa di 'Identificazione Porta Delegata'.

Effettuato lo smistamento del contenuto della SOAP Call pervenuta, il componente in questione, rimane in attesa di una risposta su di una coda JMS, filtrando la ricezione per una proprietà del messaggio JMS contenente un identificatore univoco precedentemente associato al messaggio passato al modulo del package 'mdb'. Come si può intuire, l'identificatore univoco rimarrà associato al messaggio che sarà girato tra i vari nodi dell'architettura OpenSPCoop PdD, fino a ritornare al nodo in questione. Il contenuto ritornato sarà utilizzato dal componente 'Accettazione Richieste Interne' per formare una SOAP Response da ritornare al SIL che ha effettuato la richiesta.

Ricezione Buste e-Gov. Una porta di dominio mittente, quando ha creato una busta SPCoop, la inoltra ad un'altra porta di dominio che deve quindi fornire un punto di accesso dove la spedizione deve essere eseguita. Questo componente implementa proprio questo punto di accesso, accettando solo messaggi Soap (SOAP Call) contenenti al loro interno header e-Gov. Una volta ricevuto il messaggio lo redirige al componente 'Sbustamento' incluso nel package org.openspcoop.pdd.mdb. Così come per ogni altro componente appartenente al package 'org.openspcoop.pdd.mdb', la sua realizzazione è stata effettuata tramite tecnologia MDB, e quindi il componente si trova in uno stato di attesa di un messaggio su di una coda JMS. Il passaggio è quindi realizzato tramite l'inserimento del contenuto della SOAP Call (oltre al altre informazioni di gestione) in un messaggio JMS posto nella coda di attesa di 'Sbustamento'. Effettuato lo smistamento del contenuto della SOAP Call pervenuta, il componente in questione, rimane in attesa di una risposta su di una coda JMS, filtrando la ricezione per una proprietà del messaggio JMS contenente un identificatore univoco precedentemente associato al messaggio passato al modulo del package 'mdb', esattamente come spiegato per il componente 'Accettazione richieste Interne'. Il contenuto ritornato nella risposta sarà utilizzato per formare una SOAP Response da ritornare alla porta di dominio che ha effettuato la richiesta.

Il cuore della porta di dominio, è implementato attraverso il package '**org.openspcoop.pdd.mdb**' che include diversi componenti, ognuno con la propria funzionalità, implementati tramite tecnologia MDB. Quindi ogni componente viene attivato attraverso l'inserimento di un messaggio JMS nella sua coda di attesa.

I componenti implementati nel package sono i seguenti:

Identificazione Porta Delegata. Si occupa di ricevere il contenuto di una richiesta di invocazione di una porta delegata (precedentemente elaborata dal componente 'Accettazione Richieste Interne'), e di identificarne la presenza all'interno del database dei profili delle porte delegate. Nell'attuale implementazione di OpenSPCoop PdD, il database è realizzato tramite un file di configurazione XML (vedi paragrafo 3.5.1), dove sono elencate le varie porte delegate installate nella porta di dominio. L'identificazione con successo di una porta delegata, ne causerà l'estrazione di informazioni SPCoop inerenti al servizio da invocare (erogato su di un altro dominio di cooperazione). Le informazioni includono il ServiceProvider erogante, il servizio e il tipo erogato e l'azione da intraprendere. Un messaggio JMS, contenente le varie informazioni estratte dalla porta delegata, viene costruito ed inserito nella coda di attesa del componente 'Imbustamento'.

Imbustamento. Si occupa di ricevere il contenuto di una richiesta effettuata da un SIL interno al dominio e le associate informazioni SPCoop che identificano il servizio richiesto. Le informazioni sono utilizzate per accedere al registro dei servizi di OpenSPCoop e estrarre l'accordo di servizio inerente al servizio richiesto. Nell'accordo saranno estratte le informazioni utili alla creazione della busta SPCoop quali ad esempio:

- *Profilo di Collaborazione*. Se il profilo di collaborazione non è MessaggioSingoloOneWay saranno gestiti gli stati di avanzamento del profilo come descritto nella fase di progettazione nei paragrafi 2.2.4.7-2.2.4.9.
- *Tracciamento.* Viene effettuato il tracciamento sia su file system e database, attraverso l'utilizzo del package 'org.openspcoop.pdd.output', sia inserendo opportune informazioni nella busta SPCoop in fase di costruzione.
- Consegna affidabile. Se nell'accordo di servizio è richiesto una consegna affidabile, viene inizializzata una gestione dei riscontri. La gestione sarà poi realmente effettuata dal componente 'GestioneRiscontri' appartenente al package 'org.openspcoop.pdd.threads' (per ulteriori spiegazioni, vedi il restante paragrafo).

- *Scadenza.* Se nell'accordo di servizio è presente una scadenza da associare alla busta SPCoop la scadenza viene applicata.

Il componente in questione, interagisce con il registro dei servizi anche per conoscere l'indirizzo fisico di destinazione della porta di dominio a cui dovrà essere spedita la busta. Un messaggio JMS, contenente la busta SPCoop creata e varie altre informazioni (es. indirizzo porta di dominio destinataria), viene costruito ed inserito nella coda di attesa del componente 'Inoltro Buste e-Gov'.
Inoltre se il profilo di collaborazione non prevede una risposta (OneWay), viene generata una risposta di 'richiesta presa in carico', inserita in un messaggio JMS inviato al componente 'Accettazione Richieste Interne'.

Inoltro Buste eGov. Si occupa di ricevere il contenuto di una richiesta effettuata da un SIL e imbustata con informazioni SPCoop dal componente 'Imbustamento' e di spedire la busta alla porta di dominio identificata dall'indirizzo fisico inserito come proprietà nel messaggio JMS ricevuto. Rimane, dopodichè, in attesa di una risposta HTTP che potrà contenere carico applicativo o meno. La gestione del carico differisce a seconda del profilo di collaborazione associato al servizio. Vediamo la gestione dei seguenti profili:

Messaggio Singolo OneWay.

- Carico vuoto (oppure messaggio Soap che non contiene né Fault né busta e-Gov). Transazione del profilo corretta.
- Messaggio SPCoop. Il messaggio viene smistato al componente 'Sbustamento Risposta'.
- Messaggio Soap con Fault. Il fault viene registrato nei log.

Profilo Sincrono.

- Carico vuoto (oppure messaggio Soap che non contiene ne Fault ne busta e-Gov). *Errore, le informazioni sull'invocazione del servizio vengono registrate nei log.*
- Messaggio SPCoop. Il messaggio viene smistato al componente 'Sbustamento Risposta'.
- Messaggio Soap con Fault. Il fault viene registrato nei log.

Sbustamento Risposte. Si occupa di ricevere una busta SPCoop pervenuta come risposta alla porta di dominio, in seguito ad una richiesta precedentemente effettuata. La busta viene validata sia semanticamente che sintatticamente, e se contiene una lista eccezioni, viene registrata nei log, evidenziando le eccezioni presenti.
In caso non sia presente una lista eccezioni viene controllato se sono presenti riscontri a buste precedentemente inviate. In caso affermativo i riscontri presenti vengono utilizzati per cancellare le informazioni utilizzate dal componente del package 'org.openspcoop.pdd.threads' (vedi paragrafo restante) che si occupa di re-inviare le buste non riscontrate dopo un certo intervallo di tempo.
Infine, se il profilo è sincrono, viene consegnato il messaggio pervenuto, dopo averne effettuato lo sbustamento dell'header e-Gov, al modulo 'Accettazione Richieste Interne'.

Sbustamento. Si occupa di ricevere una busta SPCoop pervenuta alla porta di dominio. La busta viene validata sia semanticamente che sintatticamente, e se contiene una lista eccezioni, viene registrata nei log, evidenziando le eccezioni presenti.
In caso non sia presente una lista eccezioni viene controllato se sono presenti riscontri e casomai sono gestiti come descritto precedentemente per il modulo 'SbustamentoRisposte'. Inoltre se viene richiesta la ricezione di buste senza duplicati, viene effettuato un controllo nell'History delle buste ricevute per verificare che la busta non sia già stata processata dalla porta di dominio.
Un messaggio JMS, contenente la richiesta pervenuta, sbustata dall'header e-Gov, con le varie informazioni estratte dalla busta SPCoop e necessarie per l'identificazione della porta applicativa richiesta, viene costruito ed inserito nella coda di attesa del componente 'Identificazione Porta Applicativa'.

Identificazione Porta Applicativa. Si occupa di ricevere il contenuto di una richiesta, verso una porta applicativa, presente all'interno di una busta SPCoop e di identificarne la presenza all'interno del database dei profili delle porte applicative. Nell'attuale implementazione di OpenSPCoop PdD, il database è realizzato tramite un file di configurazione XML (vedi paragrafo 3.5.1), dove sono elencate le varie porte applicative installate nella porta di dominio.

L'identificazione con successo di una porta applicativa (attraverso l'individuazione di una porta che presenta service provider, servizio ed azione uguale a quello pervenuto nella busta SPCoop), ne causerà l'estrazione di informazioni per la consegna al sistema informativo locale che eroga il servizio. Un messaggio JMS, contenente le varie informazioni estratte dalla porta applicativa, viene costruito ed inserito nella coda di attesa del componente 'Sbustamento'.

Consegna Buste Soap. Si occupa di ricevere il contenuto di una richiesta destinata ad un sistema informativo locale del dominio, e di utilizzarlo nell'invocazione della porta applicativa, precedentemente identificata dal modulo omonimo. L'invocazione varia a seconda dei parametri di consegna associati alla porta applicativa. Le consegne possibili nell'attuale implementazione della porta di dominio sono le seguenti:

- Invocazione di un servizio Web.
- Semplice consegna HTTP POST.
- Consegna su coda o topic JMS.

Nel caso in cui il profilo di collaborazione utilizzato sia sincrono, il componente si mette in attesa di una risposta, ed una volta ricevuta la inserisce in un messaggio JMS inviato al componente 'ImbustamentoRisposte'.

Imbustamento Risposte. Il modulo in questione si occupa di imbustare una risposta applicativa ricevuta dal modulo precedente, e di inviare la busta SPCoop al modulo 'Ricezione Buste e-Gov' che si trova in attesa della risposta.

La porta di dominio OpenSPCoop, oltre ai due sotto-package precedentemente descritti, contiene anche altri sotto-package con funzionalità minori.

Il package '**org.openspcoop.pdd.output**' contiene un modulo 'Logger' che si occupa, attraverso la tecnologia Log4j, di registrare su files e su database messaggi diagnostici e di tracciamento effettuati dalla porta di dominio. Le impostazioni di registrazione sono effettuate attraverso un **file di proprietà** apposito descritto nella sezione 3.5.2.

Il package '**org.openspcoop.pdd.thread**' è composto da un 'Contenitore', che è un componente predisposto a contenere diversi Threads di gestione delle funzionalità SPCoop. Inoltre nel contenitore viene inserito il componente 'GestioneRiscontri', che si occupa di re-inviare buste SPCoop (salvate in un history delle buste inviate) le quali non sono state ancora riscontrate dopo che un determinato intervallo di tempo è trascorso dal loro invio.

Infine il package '**org.openspcoop.pdd.config',** in particolar modo il componente 'XMLReader', viene utilizzato per la lettura del file di configurazione dove sono memorizzate porte delegate (con informazioni di accesso) e porte applicative. Il file XML viene dinamicamente mappato in strutture dati, attraverso l'utilizzo della tecnologia JiBX.

Nel paragrafo seguente, viene descritto il file di configurazione di OpenSPCoop contenente la definizione di porte delegate, porte applicative e diritti di accesso. Inoltre viene descritto anche il file di proprietà di OpenSPCoop dove è possibile fornire alcune proprietà di configurazione della porta di dominio. Dopodichè viene illustrato uno scenario d'utilizzo della porta di dominio, lo stesso utilizzato nel paragrafo 3.3.2, evidenziando la facilità con cui OpenSPCoop PdD permette di costruire porte delegate e porte applicative.

3.5.1 Il file di configurazione delle porte delegate e applicative

La creazione di una porta delegata (con associati diritti di invocazione) o di una porta applicativa, viene impostato, nella porta di dominio OpenSPCoop, attraverso un semplice file di configurazione).

Il codice seguente illustra un esempio di file di configurazione:

```
<openspcoop xmlns="http://www.openspcoop.org/pdd/config/xml"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation=
            "http://www.openspcoop.org/pdd/config/xml config.xsd">

<porta_delegata url="OneWay" tipo="static">
      <service_provider tipo="SPC">LINKIT</service_provider>
      <servizio tipo="SPC">Tracer</servizio>
      <azione>Rec</azione>
      <sil_mittente>SIL_NoAuthentication</sil_mittente>
</porta_delegata>

<porta_delegata url="OneWayAuthentication" tipo="static">
      <service_provider tipo="SPC">LINKIT</service_provider>
      <servizio tipo="SPC">Tracer</servizio>
      <azione>Rec</azione>
      <sil_mittente>SIL_Example</sil_mittente>
</porta_delegata>

<porta_delegata url="OneWayJMS" tipo="static">
      <service_provider tipo="SPC">LINKIT</service_provider>
      <servizio tipo="SPC">ConsegnaJMS</servizio>
      <azione>Push</azione>
      <sil_mittente>SIL_NoAuthentication</sil_mittente>
</porta_delegata>

<porta_delegata url="OneWayJMSTopic" tipo="static">
      <service_provider tipo="SPC">LINKIT</service_provider>
      <servizio tipo="SPC">ConsegnaJMS</servizio>
      <azione>Push_onTopic</azione>
      <sil_mittente>SIL_NoAuthentication</sil_mittente>
</porta_delegata>

<porta_applicativa>
      <service_provider tipo="SPC">LINKIT</service_provider>
      <servizio tipo="SPC">Tracer</servizio>
      <azione>Rec</azione>
      <consegna>
          <http url="http://localhost:8080/TRACE/trace"
                                    sbustamento_soap="true" />
      </consegna>
</porta_applicativa>
```

```
<porta_applicativa>
      <service_provider tipo="SPC">LINKIT</service_provider>
      <servizio tipo="SPC">ConsegnaJMS</servizio>
      <azione>Push</azione>
      <consegna>
         <jms name="queue/TEST" tipo="queue"
              send_as="BytesMessage"  sbustamento_soap="true">
            <context initial_context_factory=
                  "org.jnp.interfaces.NamingContextFactory"
                  url_pkg_prefixes="org.jnp.interfaces"
                  provider_url="127.0.0.1"
                  connection_factory="ConnectionFactory" />
            <properties>
                  <service_provider to_property="SP" />
                  <tipo_service_provider to_property="tipoSP" />
                  <servizio to_property="servizio" />
                  <azione to_property="action" />
            </properties>
         </jms>
      </consegna>
</porta_applicativa>

<porta_applicativa>
      <service_provider tipo="SPC">LINKIT</service_provider>
      <servizio tipo="SPC">ConsegnaJMS</servizio>
      <azione>Push_onTopic</azione>
      <consegna>
         <jms name="topic/TEST_TOPIC" tipo="topic"
                send_as="TextMessage" sbustamento_soap="true">
              <context   initial_context_factory=
                  "org.jnp.interfaces.NamingContextFactory"
                  url_pkg_prefixes="org.jnp.interfaces"
                  provider_url="127.0.0.1"
                  connection_factory="ConnectionFactory" />
               <properties>
                  <mittente to_property="mitt" />
                  <tipo_mittente to_property="tipoMitt" />
               </properties>
         </jms>
      </consegna>
</porta_applicativa>


<porta_di_dominio>
      <id>PisaLinkSPCoopIT</id>
</porta_di_dominio>

<sil>
      <principal>SIL_NoAuthentication</principal>
      <authentication tipo="ALLOW-ALL" />
      <provider tipo="SPC">MittenteDiProva</provider>
</sil>

<sil>
      <principal>SIL_Example</principal>
      <authentication tipo="HTTP-BASED" password="123456" />
      <provider tipo="SPC">MittenteDiProva</provider>
</sil>

</openspcoop>
```

I vari attori definibili nel file di configurazione, sono i seguenti:

- Porte Delegate
- Porte Applicative
- Sil mittenti
- Informazioni sulla Porta Di Dominio

Porte Delegate.

Al momento dell'invocazione di un servizio da parte di un SIL (Client), il modulo OpenSPCoop che raccoglie la richiesta di servizio, si appoggia ad una lista dei **Profili delle Porte Delegate** per riuscire ad identificare dinamicamente il servizio (o l'evento) richiesto. Quindi, in un tempo antecedente all'invocazione di un servizio, deve essere fatta una registrazione nel file di configurazione di una porta delegata apposita per il servizio stesso.

Al momento della registrazione, per ogni porta delegata, andranno specificati i seguenti dati:

- ***url.*** Rappresenta la porta delegata richiesta dal Client. Ad esempio, se viene definita una porta delegata con questo parametro uguale al valore 'Test', poi il client dovrà invocare la URL della porta di dominio, associando come parte finale dell'url il valore 'Test'.

  Supponendo che sia attiva una porta di dominio all'indirizzo 'http://portadidominio.org', il client (SIL) dovrà invocare il servizio 'http://portadidominio.org/**Test**' per accedere alla porta delegata in questione.
- ***tipo*.** Indica il tipo di porta delegata tra tre possibili tipi: 'static-based', 'url-based' e 'content-based'. Per ulteriori spiegazioni si rimanda al paragrafo 2.2.2 che discute dell'interfacciamento SIL-to-PdD.
- ***Service Provider*.** Indica un SIL, appartenente ad un'altra porta di dominio, che fornisce l'erogazione del servizio associato a questa porta delegata. Può essere anche l'EventManager se il servizio richiesto è una pubblicazione o sottoscrizione di un evento.
- ***Servizio e tipo.*** Indica l'identificativo del servizio (e del tipo associato) fornito dal Service Provider.
- ***Azione.*** Specifica il tipo di azione richiesta al servizio (può identificare anche un evento o una lista di eventi).

- ***SIL*-Mittente.** Indica il SIL che potrà invocare la porta Delegata. (Sarà utilizzato per vedere che tipo di autenticazione è associata al SIL).

La Figura 3-11 illustra un esempio di invocazione di un servizio, da parte di un SIL. Il SIL1 vuole utilizzare il servizio 'Anagrafe'. In questo esempio, il SIL aveva effettuato la registrazione della porta delegata, fornendo i dati del ServiceProvider, del servizio (e tipo) e l'azione al momento della registrazione. **L'accesso alla porta delegata**, al momento dell'invocazione del servizio da parte del SIL permette di estrarre le informazioni (ServiceProvider, Servizio e tipo, Azione) necessarie per la creazione della busta e-Gov.

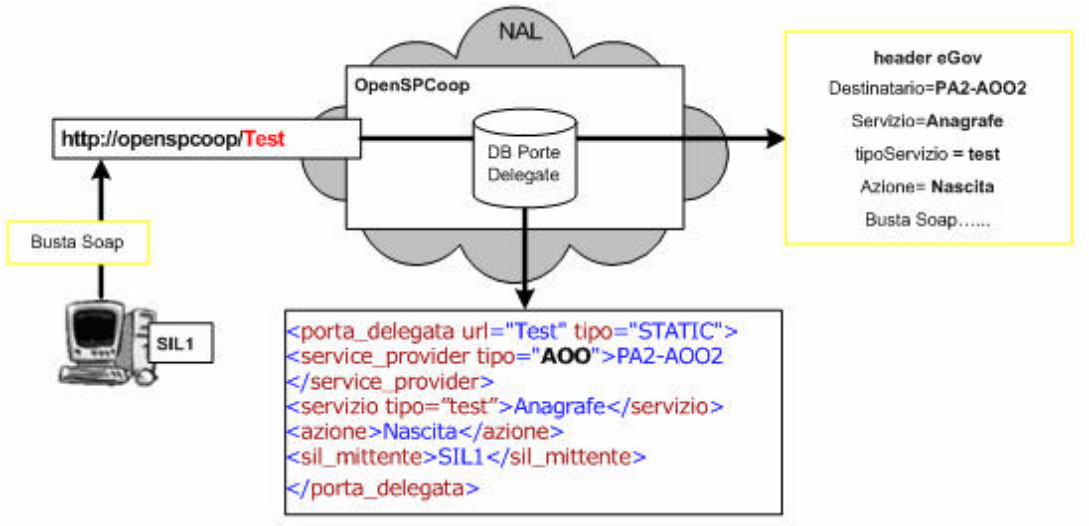


Figura 3-11, Esempio di Porta Delegata

Porte Applicative.

Il servizio applicativo, fornito da un SIL, deve essere registrato come porta applicativa all'interno della porta di dominio. Quindi in una porta di dominio è presente una lista dei **Profili delle Porte Applicative**, in cui sono registrati tutti i servizi applicativi forniti dai SIL che vi appartengono.

Le informazioni che sono necessarie, al momento della creazione di una porta applicativa, sono le seguenti:

- ***Service Provider*.** Indica il SIL che possiede il servizio di erogazione.
- ***Servizio e tipo.*** Indica l'identificativo del servizio (e del tipo associato) fornito dal Service Provider.

- ***Azione.*** Specifica il tipo di azione richiesta al servizio.
- **Consegna.** Specifica il tipo e i parametri per effettuare l'invocazione del servizio. Per ulteriori chiarimenti vedi il resto del paragrafo.

La Figura 3-12 illustra un esempio di invocazione di una porta applicativa, che è collegata ad un servizio web. Una richiesta e-Gov arriva ad OpenSPCoop, che utilizzando le informazioni incluse nella busta, identifica dinamicamente il servizio applicativo da invocare, interagendo con la tabella dei profili delle porte applicative.

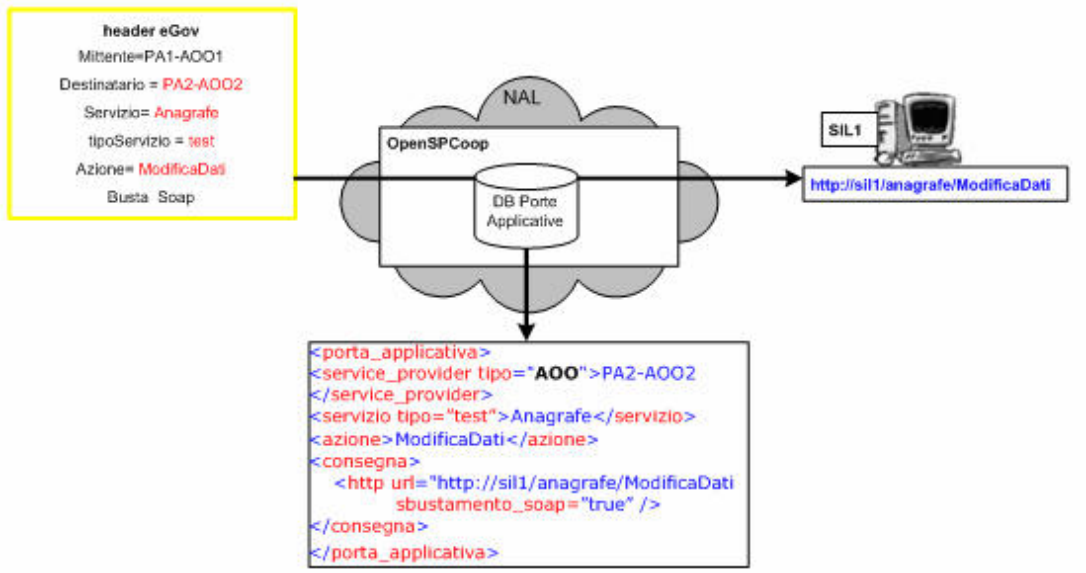


Figura 3-12, Esempio di Porta Applicativa con consegna 'HTTP' ad un Server Web

Il tipo di consegna illustrato nella Figura 3-12, è la **consegna http**. E' ulteriormente classificata in due tipi:

- ***Invocazione di un servizio web.*** (attributo 'sbustamento_soap' impostato al valore 'false'). In questo contesto, la busta arrivata alla porta di dominio, viene sbustata dall'header e-Gov, e dopodichè viene utilizzato il restante SoapEnvelope per l'invocazione del servizio web fornito nell'url indicata nella proprietà 'url' del tag 'http'.
- ***Consegna http.*** (attributo 'sbustamento_soap' impostato al valore 'true'). Il soap envelope, pervenuto alla porta di dominio, oltre ad essere sbustato dell'header e-Gov, viene ulteriormente sbustato delle informazioni soap e viene consegnato all'url indicata nella proprietà 'url' del tag '.

La Figura 3-13 illustra invece, l'altro tipo di consegna fornito da OpenSPCoop. La **consegna** avviene **su una coda (o topic) JMS**.

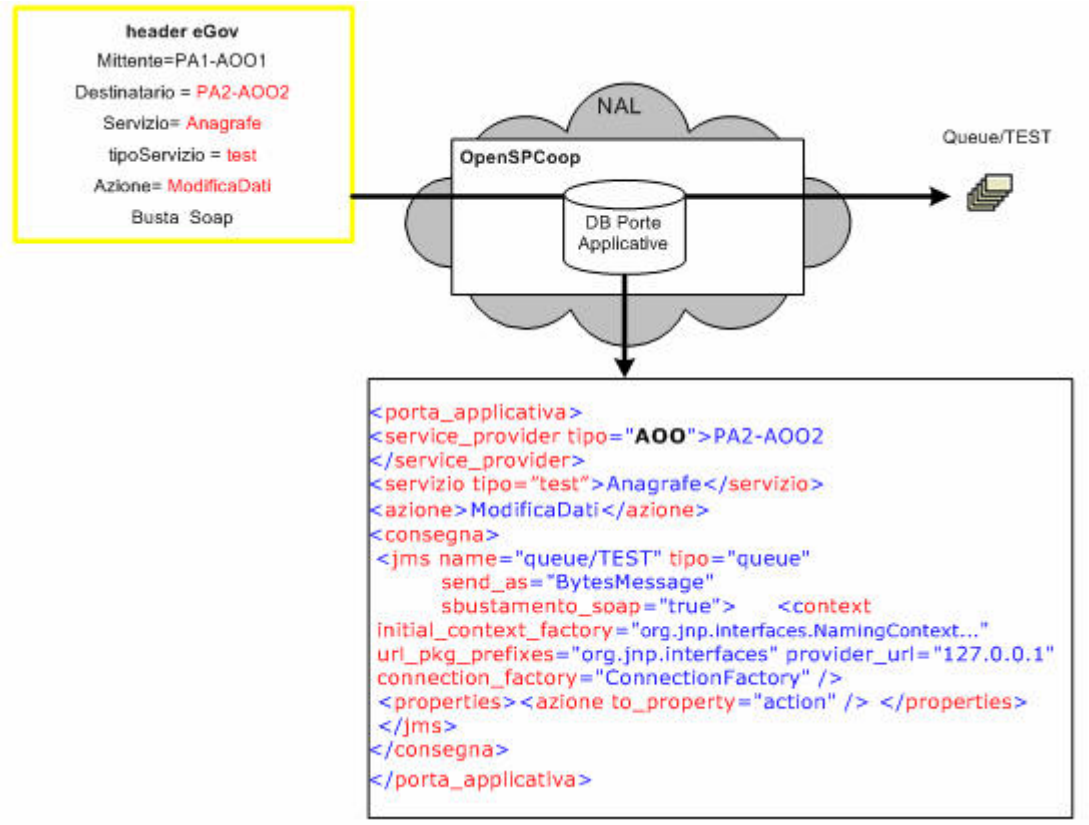


Figura 3-13, Esempio di Porta Applicativa con consegna 'JMS' su di una coda

. I seguenti parametri permettono di personalizzare la consegna JMS:

- **name.** Indica il nome della coda (o del topic) su cui effettuare l'inserimento. (il nome deve essere fornito con l'estensione completa, ad es. queue/ TEST)
- **tipo.** Indica il tipo di coda utilizzata. Valori possibili sono 'queue' per effettuare un inserimento su di una semplice coda, o 'topic' per effettuare una pubblicazione su di un topic.
- **send_as.** Indica il tipo di oggetto utilizzato per la consegna nella coda. I tipi possibili sono 'TextMessage' o 'BytesMessage'.
- **sbustamento_soap**. Indica se la busta arrivata alla porta di dominio, una volta sbustata dell'header e-Gov, deve essere interamente inserita nella coda (valore uguale a 'false') o se nella coda deve essere inserito solo il contenuto del soap body (valore uguale a 'true').

- **Parametri di configurazione (tag context)**. Attraverso il tag 'context' è possibile fornire i parametri per l'accesso alla coda:
  - **initial_context_factory.**
  - **url_pkg_prefixes.** (opzionale).
  - **provider_url.**
  - **connection_factory.**
- **Proprietà JMS (tag properties).** Attraverso il tag 'properties' è possibile indicare la proprietà del messaggio JMS in cui includere specifici valori della busta e-Gov pervenuta, specificando il nome della proprietà (attributo 'toProperty'). I valori della busta e-Gov utilizzabili sono selezionabili attraverso i seguenti tag:
  - <mittente to_property=" **nomeProperty**" />
  - <tipo_mittente to_property="**nomeProperty**" />
  - <service_provider to_property=" **nomeProperty**" />
  - <tipo_service_provider to_property=" **nomeProperty**" />
  - <servizio to_property=" **nomeProperty**" />
  - <tipo_servizio to_property="**nomeProperty"**/>
  - <azione to_property=" **nomeProperty** " />

Autenticazione ed Autorizzazione dei Sistemi Informativi Locali.

Nella porta di dominio devono essere registrati tutti i SIL che possono invocare servizi. Per ogni SIL Mittente deve essere specificato:

- **Principal.** Indica l'username in caso di autenticazione HTTP-BASED.
- **Tipo autenticazione.** Indica il tipo di autenticazione associato ad un SIL, ad esempio 'HTTP-BASED'. Inoltre è possibile permettere che una porta delegata sia invocata da ogni SIL indiscriminatamente, semplicemente associando alla porta delegata un SIL particolare, la cui definizione contenga come tipo di autenticazione 'ALLOW-ALL'.
- **Password.** Indica la password che dovrà essere utilizzata dal SIL Mittente per autenticarsi tramite autenticazione HTTP-BASED.
- **Provider.** Indica il codice e il tipo dell'area organizzativa a cui appartiene il SIL mittente.

Ogni client, per effettuare un'invocazione di una porta delegata, dovrà utilizzare il proprio username e la propria password per effettuare l'autenticazione. Se la porta di dominio riceve una richiesta di servizio senza informazioni di autenticazione (es. header http authentication per l'autenticazione di tipo HTTP-BASED) restituisce un messaggio di errore applicativo 'Authentication Required', a meno che la porta delegata non abbia associata una definizione di SIL che non richieda autenticazione (tipo uguale a 'ALLOW-ALL'). Se le informazioni di autenticazione sono scorrette, la porta di dominio genererà un messaggio di errore applicativo 'Authentication Failed'. Inoltre un SIL potrà essere associato ad una o più porte delegate, e potrà utilizzare solo quelle. Se un SIL prova ad utilizzare una porta delegata non associatagli, la porta di dominio genererà un messaggio di errore applicativo 'Authorization Failed'.

Informazioni sulla Porta di Dominio.

E' possibile associare alla porta di dominio le seguenti informazioni:

- **Nome.** Informazione di carattere generale associato alla porta di dominio.
- **Codice.** Il codice della porta di dominio è importante per l'imbustamento e-Gov.

3.5.2 Configurazione del prodotto OpenSPCoop PdP

La porta di dominio 'OpenSPCoop' è reperibile all'indirizzo 'www.openspcoop.org'. L'ultima versione scaricabile contiene la seguente struttura:

- **Deploy**. In questa cartella sono reperibili i files necessari per una corretta installazione della porta di dominio.
- **Src**. Vengono forniti i sorgenti '.java' che realizzano la porta di dominio, e la libreria 'org.openspcoop.egov' utilizzata per gestire le funzionalità e-Gov.
- **Doc**. Viene fornita la documentazione, comprendente manuali d'uso e d'installazione, oltre alla documentazione API della libreria 'org.openspcoop.egov' e 'org.openspcoop.pdd'.
- **Example.** Vengono forniti Client e Server esemplificativi, che permettono di utilizzare la porta di dominio e testarne le funzionalità offerte.

Attualmente, all'interno della distribuzione del prodotto sono presenti due files 'properties' che contengono la configurazione della porta di dominio. Attraverso uno dei due file, '**openspcoop.properties',** è possibile impostare vari aspetti funzionali della porta di dominio OpenSPCoop, mentre con l'altro '**logger.log4j.properties'** è possibile impostare aspetti di logging degli eventi.
Il file di configurazione 'openspcoop.properties' è inserito all'interno della distribuzione nella cartella 'deploy/pdd/properties/openspcoop.properties'. Gli aspetti configurabili sono i seguenti:

1) File di Configurazione. Deve essere fornito il path (o una url) dove è possibile reperire il file di configurazione che include la definizione di porte delegate e di porte applicative gestite dalla porta di dominio descritto nel paragrafo precedente. Un esempio di file di configurazione 'config.xml' (e di schema associato 'config.xsd'), utilizzato poi dagli esempi della guida utente, lo si può trovare nella cartella '/deploy/pdd/config_file/' della distribuzione.
Se si desidera cambiare il path del file di configurazione (il file può essere anche posto in remoto; in tal caso dovrà essere fornita una url) bisogna modificare la voce 'org.openspcoop.pdd.config.fileConfig', ed impostare un cammino alternativo.
Se si desidera porre lo schema config.xsd in una locazione diversa da dove è posto il file config.xml, bisogna editare il file xml e cambiare la voce 'xsi:schemaLocation="http://www.openspcoop.org/pdd/config config.xsd"' fornendo il cammino completo del file config.xsd.
Vediamo alcuni esempi:

1. 'xsi:schemaLocation="http://www.openspcoop.org/pdd/config /home/user/config.xsd"'
2. 'xsi:schemaLocation="http://www.openspcoop.org/pdd/config http://www.mysite.org/config.xsd"'

2) Registro dei servizi di OpenSPCoop. Attualmente l'utilizzo del registro dei servizi, come è stato illustrato nella sezione 3.4, è disponibile in due versioni:

- **XML**. Il registro è realizzato tramite un semplice file xml. Questa versione è utile a coloro che desiderano provare la porta di dominio OpenSPCoop senza dover installare un registro dei servizi reale.

- **UDDI.** Il registro è realizzato tramite un registro UDDI. Una prima versione (sperimentale) è realizzata attraverso la classe QueryEGovUDDI del package org.openspcoop.uddi.

La scelta del tipo di registro può essere effettuata tramite il file properties 'openspcoop.properties' attraverso la voce 'org.openspcoop.uddi.tipo'.
Un esempio di registro dei servizio (di tipo XML) 'registroServizi.xml' (e di schema associato 'registroServizi.xsd'), utilizzato poi dagli esempi della guida utente, lo si può trovare nella cartella '/deploy/pdd/config_file/' della distribuzione.
Se si desidera cambiare il tipo del registro dei servizi, e l'associata url (il file di un registro di tipo XML può essere anche posto in remoto) bisogna modificare la voce 'org.openspcoop.uddi.url', ed inserire una url che identifica un registro dei servizi UDDI, o una url che identifica un file xml (se il file è in locale dovrà essere fornito il cammino).
Se si desidera porre lo schema registroServizi.xsd in una locazione diversa da dove è posto il file registroServizi.xml, bisogna editare il file xml e cambiare la voce 'xsi:schemaLocation="http://www.openspcoop.org/uddi/xmlregistry registroServizi.xsd"' fornendo il cammino completo del file registroServizi.xsd. Vediamo alcuni esempi:

1. 'xsi:schemaLocation="http://www.openspcoop.org/uddi/xmlregistry /home/user/registroServizi.xsd"'
2. 'xsi:schemaLocation="http://www.openspcoop.org/uddi/xmlregistry http://www.mysite.org/registroServizi.xsd"'

3) Validazione della Busta e-Gov. Attraverso il file openspcoop.properties è possibile impostare, utilizzando la voce 'org.openspcoop.egov.validazione', la modalità di validazione delle buste ricevute dalla porta di dominio ad uno dei tre seguenti valori:

1. **Warning_only**. Se si verificano errori di validazione, questi comporteranno solamente la produzione di un messaggio diagnostico.
2. **Active**. In caso di errori di validazione, sarà prodotto un messaggio diagnostico e la busta e-Gov non sarà ulteriormente processata. Inoltre sarà prodotta un Messaggio SPCoop Errore da inoltrare al mittente della busta.
3. **Not_Active**. La validazione della busta non viene effettuata.

Inoltre OpenSPCoop mette a disposizione due tipi di validazione della busta, intercambiabili attraverso la voce 'org.openspcoop.egov.tipoValidazione':

1. **openspcoop**. Viene effettuata la validazione senza l'utilizzo dello schema xsd. Questo tipo di validazione comporterà una prestazione maggiore rispetto all'altro tipo, ma non garantisce che la validazione sia esente da errori.
2. **conSchema**. Viene effettuata la validazione con l'ausilio dello schema xsd che descrive gli elementi costitutivi di una busta SPCoop.

***4) Connessione al Database*.** Nella **versione attuale** di OpenSPCoop **l'utilizzo di un database è obbligatorio per il funzionamento della porta di dominio**. E' possibile impostare la configurazione di accesso al DB le seguenti voci:

- org.openspcoop.database.url, permette di definire la stringa di connessione al DB utilizzata dal Connettore JDBC.
- org.openspcoop.database.driver.classname, permette di definire il nome della classe del driver JDBC da utilizzare.
- org.openspcoop.database.user e org.openspcoop.database.password. Queste voci permettono di definire l'username e la password che la porta di dominio deve utilizzare per connettersi al DB.

5) Directory per il Salvataggio di informazioni persistenti. La versione attuale di OpenSPCoop richiede la possibilità di utilizzare uno spazio fisico su disco. I file memorizzati non sono semplici files temporanei ma dovranno rimanere reperibili anche in caso di riavvii della porta di dominio (e della macchina fisica su cui risiede).

6) Timeout per il re-Invio di buste e-Gov non riscontrate.

La proprietà 'org.openspcoop.riscontri.timeout' permette di definire un intervallo di minuti dopo cui, la porta di dominio effettua un re-invio di una busta, precedentemente inviata, se per quella busta non ha ricevuto un riscontro.

Inoltre, attraverso il file '**logger.log4j.properties',** è possibile impostare vari aspetti di configurazione del logging effettuato dalla porta di dominio OpenSPCoop. Il file di configurazione 'logger.log4j.properties' è inserito all'interno della distribuzione nella cartella 'deploy/pdd/properties /logger.log4j.properties'.

Gli aspetti configurabili sono i seguenti:

- **FileAppender.** Dovrà essere indicato il file da utilizzare per la registrazione dei tracciamenti delle buste gestite dalla porta di dominio, ed il file utilizzato per la registrazione dei messaggi diagnostici emessi.
- **JDBCAppender.** Dovrà essere indicato la url di connessione al DB utilizzato oltre all'username e la password necessaria per la connessione.

3.5.3 Scenario e vantaggi di utilizzo di OpenSPCoop PdD

Nell'esempio illustrato nel paragrafo 3.5.2, viene illustrato l'utilizzo della libreria e-Gov attraverso la definizione di un Proxy Delegato e di un Proxy Applicativo specifico per il contesto descritto. Ogni proxy deve essere costruito appositamente per gestire il proprio servizio, e deve utilizzare la libreria org.openspcoop.egov per costruire, validare e sbustare buste e-Gov. La complessità della specifica e-Gov è in mano all'implementatore dei proxy, e deve essere rivista ogni qualvolta si volesse definire un nuovo proxy delegato o un nuovo proxy applicativo. La Figura 3-14 illustra come sia necessario implementarsi completamente la logica di un nuovo proxy delegato, se il contesto necessita di un nuovo soggetto (SIL3) che possa richiedere un servizio erogato dal solito SIL2.

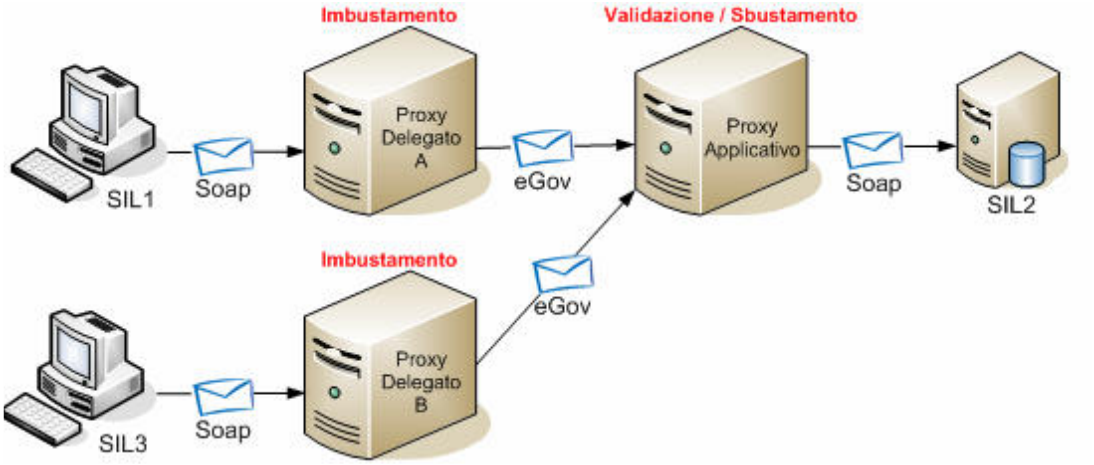


Figura 3-14, Aggiunta di un fruitore nello scenario di Figura 3-8

Con la porta di dominio OpenSPCoop è possibile 'ignorare' le complesse e difficili implementazioni dei proxy, dato che la implementazione di una porta delegata, o di una porta applicativa può essere effettuata semplicemente definendo qualche riga di codice XML. Non servirà quindi più la figura di un programmatore che con diverse centinaia di ore-uomo implementa un proxy applicativo specifico per il contesto. Lo scenario, illustrato dalla precedente Figura 3-14, attraverso la porta di dominio OpenSPCoop, evolve nella maniera descritta dalla Figura 3-15.

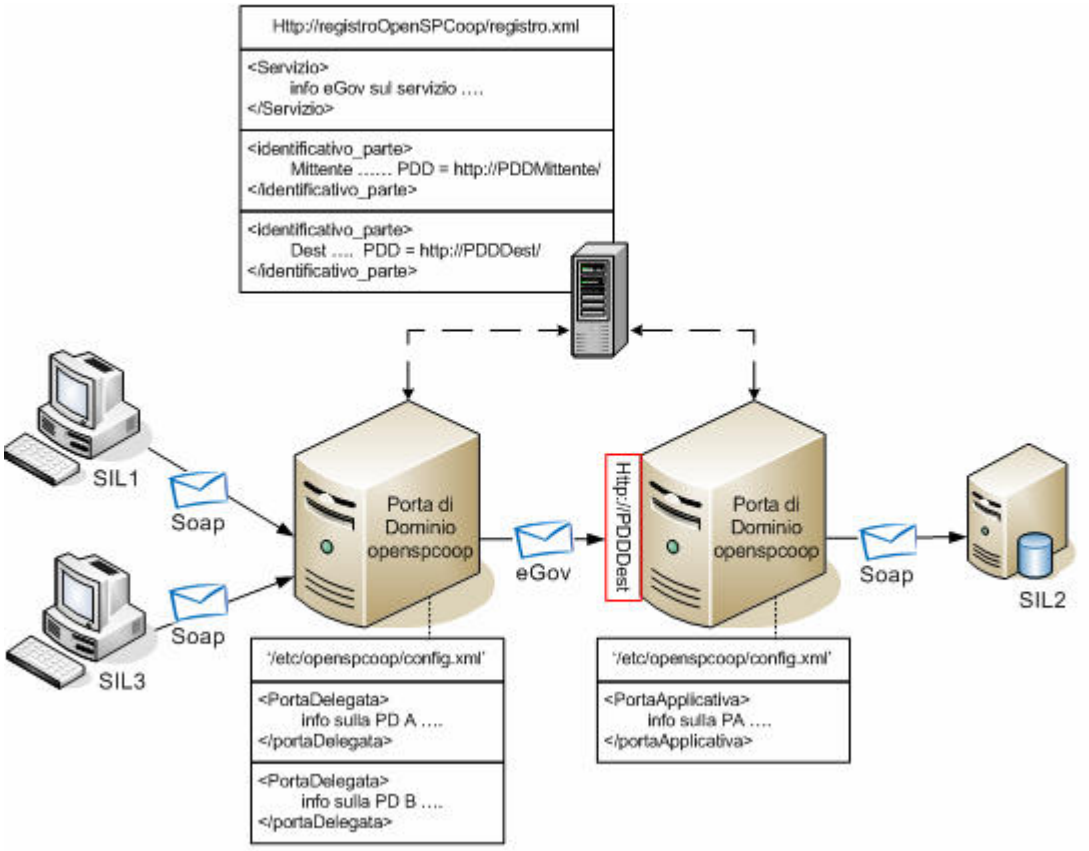


Figura 3-15, Scenario di utilizzo del prodotto OpenSPCoop PdD

Per evidenziare i vantaggi che comporta l'utilizzo di OpenSPCoop, esaminiamo lo stesso scenario d'uso, illustrato nel paragrafo 3.3.2, dove un client, vuole spedire un messaggio Soap (contenente dati applicativi) ad un Server. La spedizione non è più mediata da un proxyDelegato specifico ma dalla porta di dominio OpenSPCoop (in particolare da una porta delegata apposita), con cui il Client è forzato a dialogare. Anche il servizio offerto dal server è mediato da una porta applicativa configurata nella porta di dominio destinataria. Il server, ricevuti i dati dal cliente, genera una risposta applicativa che dovrà essere restituita al client sempre dietro la mediazione delle porte di dominio. La Figura 3-16 illustra il nuovo scenario.

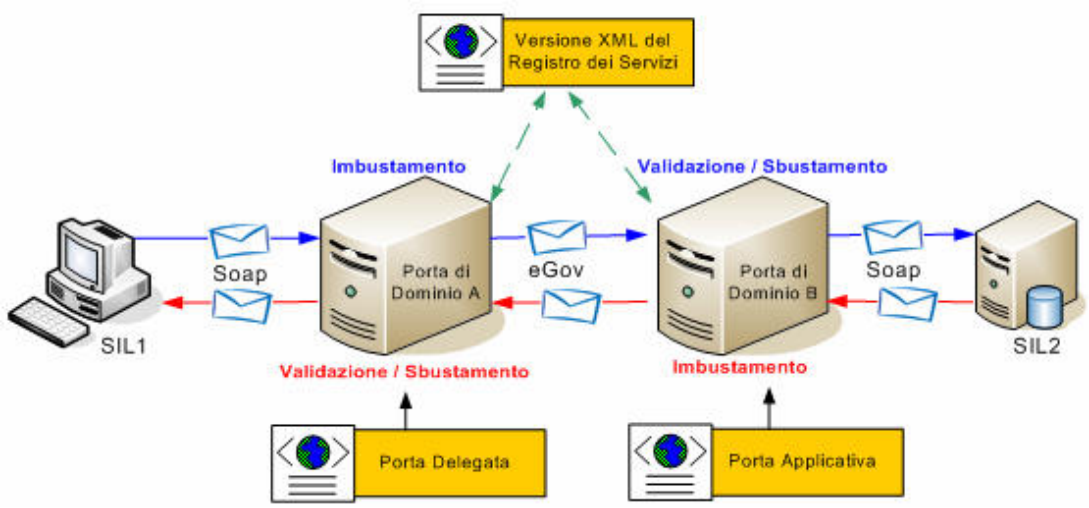


Figura 3-16, Scenario di utilizzo del paragrafo 3.3.2 evoluto con il prodotto OpenSPCoop

L'installazione dello scenario descritto non richiede più diverse centinaia di righe di codice specifico per il contesto, come specificato nel paragrafo 3.3.2, ma semplicemente la definizione di tre file XML, due per la configurazione di una porta delegata ed una porta applicativa, ed uno per la configurazione del registro dei servizi.

Il file di configurazione necessario alla configurazione della porta delegata e alla gestione dell'autorizzazione del SIL1 è il seguente (questo file di configurazione viene letto dalla porta di dominio A):

```
<openspcoop xmlns="http://www.openspcoop.org/pdd/config/xml"
      xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
      xsi:schemaLocation=
            "http://www.openspcoop.org/pdd/config/xml config.xsd">

<porta_delegata url="ServiceExample" tipo="static">
      <service_provider tipo="SPC">Server</service_provider>
      <servizio tipo="SPC">Xmethods</servizio>
      <azione>Get</azione>
      <sil_mittente>SIL_1</sil_mittente>
</porta_delegata>

<sil>
      <principal>SIL_1</principal>
      <authentication tipo="HTTP-BASED" password="123456" />
      <provider tipo="SPC">MittenteDiProva</provider>
</sil>

</openspcoop>
```

Il file di configurazione necessario, invece alla configurazione della porta applicativa è il seguente (questo file di configurazione viene letto dalla porta di dominio B):

```
<openspcoop xmlns="http://www.openspcoop.org/pdd/config/xml"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
            "http://www.openspcoop.org/pdd/config/xml config.xsd">

<porta_applicativa>
      <service_provider tipo="SPC">Server</service_provider>
      <servizio tipo="SPC">Xmethods</servizio>
      <azione>Gec</azione>
      <consegna>
           <http url="http://64.124.140.30/soap "
                 sbustamento_soap="false" />
      </consegna>
</porta_applicativa>

</openspcoop>
```

Inoltre dovrà essere creato un registro dei servizi di OpenSPCoop, dove è possibile specificare le funzionalità SPCoop da associare al servizio. Un semplice cambio di qualche opzione, modificherà facilmente il tipo di buste scambiate tra le porte di dominio. Il file di configurazione del registro dei servizi per questo scenario è il seguente:

```
<registro_servizi xmlns="http://www.openspcoop.org/uddi/xmlregistry"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.openspcoop.org/uddi/xmlregistry
      registroServizi.xsd">

<porta_applicativa>
      <service_provider tipo="SPC">Server</service_provider>
      <servizio valore="Xmethods">
            <tipo_servizio valore="SPC" />
      </servizio>
      <default utilizzo_senza_azione="false">
            <azioni_permesse valore="Get" />
            <profilo_asincrono_asimmetrico utilizzo="false" />
            <profilo_asincrono_simmetrico utilizzo="false" />
            <profilo_sincrono utilizzo="true" />
            <profilo_oneway utilizzo="false" />
            <collaborazione utilizzo="false" />
            <ordine_consegna utilizzo="false" />
            <inoltro_senza_duplicati utilizzo="true" />
            <conferma_ricezione utilizzo="true" />
      </default>
</porta_applicativa>

<identificativo_parte>
      <service_provider tipo="SPC">
            MittenteDiProva
      </service_provider>
      <id_porta_di_dominio>PisaLinkSPCoopIT</id_porta_di_dominio>
</identificativo_parte>
```

```
<identificativo_parte>
      <service_provider tipo="SPC">Server</service_provider>
      <id_porta_di_dominio>PisaLinkSPCoopIT</id_porta_di_dominio>
</identificativo_parte>

<porta_di_dominio>
      <id>PisaLinkSPCoopIT</id>

      <end_point>http://localhost:8080/axis/services/OpenSPCoop_PA</
end_point>
</porta_di_dominio>

</registro_servizi>
```

Il tempo speso per la configurazione di questi file XML è pochissimo, ma sufficiente per implementare lo stesso scenario di utilizzo illustrato nel paragrafo 3.3.2.

4. Il progetto OpenSPCoop

In questo capitolo viene presentata l'infrastruttura di progetto realizzata per OpenSPCoop, esaminando anzitutto i motivi per cui è stata ritenuta strategica la scelta di un'implementazione Open Source (sezione 4.1) e descrivendo quindi i vari componenti dell'infrastruttura predisposta per favorire la crescita e la diffusione del software (sezione 4.2). Infine, nella sezione 4.3, verranno riportati alcuni risultati significativi ottenuti con l'attuale implementazione del prodotto OpenSPCoop, in particolare evidenziando come la disponibilità di una comunità di utenti e sviluppatori esperti della specifica SPCoop abbia apportato concreti benefici nello sviluppo del software.

4.1 Motivazione di una implementazione OpenSource

Il software libero (open source) ha richiamato negli ultimi anni sempre maggiore interesse da parte delle Pubbliche Amministrazioni (PA) europee, a causa di tre fattori principali:

1. la nascita di applicativi open source di qualità in numerose aree di interesse delle PA;
2. le caratteristiche intrinseche al modello di sviluppo stesso di questo tipo di software;
3. le potenzialità insite in esso di stimolare l'innovazione nel campo dell'ICT.

Il software open source, tradizionalmente riservato a sviluppatori di software di sistema (emacs, gcc, gdb, …), già agli inizi degli anni '90 poteva vantare ottime soluzioni nel campo del *software d'infrastruttura* (es. servizi Internet, sendmail, postfix, apache, etc…). Solo dalla seconda metà degli anni '90, invece, il software a codice aperto comincia a diventare un'alternativa appetibile per le PA nel campo degli applicativi per l'utente finale. Fu solo da questo momento che vennero sviluppate le prime *graphical user interfaces* per Linux, Gnome e Kde, che hanno permesso l'utilizzo di questo sistema operativo anche agli utenti inesperti.

Nel campo della scrittura di testi e fogli elettronici, il dominio incontrastato del software proprietario trova un'alternativa solo nel 2000, anno in cui viene offerta da Sun Microsystems una versione open source della sua suite da ufficio Star Office, che allo stato attuale si è rivelato un programma di ottima qualità, ricco di funzionalità e basato su un formato aperto, basato su XML.

I benefici percepiti nell'adozione di un software libero da parte delle pubbliche amministrazioni vanno rintracciati soprattutto nel **basso costo iniziale di azione**, nell'**indipendenza dai fornitori**, nella **sicurezza**, nella **flessibilità** e nell'**interoperabilità**. I *costi iniziali* dell'adozione di software open source, rispetto all'utilizzo di software proprietario, si abbassano soprattutto in merito ai costi delle licenze e degli aggiornamenti. Molto spesso gli aggiornamenti del software proprietario vengono adottati, non tanto perché apportano delle vere migliorie al programma, ma piuttosto per incompatibilità con le versioni precedenti. Nel confrontare le spese di adozione da parte delle PA di software libero o proprietario non basta fare riferimento soltanto ai costi iniziali ma è bene valutare il TCO (*Total Cost of Ownership*). Con questo termine si prende in considerazione non soltanto i costi delle licenze ma anche i servizi di consulenza e supporto, la formazione degli impiegati, i costi di gestione e, non ultimi, i costi di migrazione. Questi in particolare possono essere molto onerosi in quanto spesso richiedono una ristrutturazione dei modi operativi dell'ente e investimenti sulla formazione del personale. Il software libero si rivela un'ottima soluzione soprattutto nei casi in cui si necessita di software custom, ovvero realizzato su misura da qualche azienda per l'Amministrazione che ne fa richiesta. I costi di sviluppo, in questo caso, si abbassano molto proprio grazie alla contribuzione volontaria di sviluppatori da ogni parte del mondo (nella misura in cui il **progetto riesca effettivamente ad attrarre un alto numero di partecipanti**). La libera consultazione dei sorgenti può allo stesso tempo fornire maggiori garanzie di sicurezza, rendendo possibile effettuare controlli sulla presenza di bachi e *back doors*.

Il progetto OpenSPCoop sposa il mondo open source. Rendendo il progetto OpenSPCoop a codice aperto si possono ottenere i seguenti vantaggi:

- ***Interoperabilità***. OpenSPCoop può rappresentare un riferimento per comprendere ed esaminare le diverse interpretazioni delle specifiche SPCoop.
- ***Sicurezza***. Il progetto può assicurare quelle caratteristiche di trasparenza del codice ormai considerate un atto dovuto in molti settori della sicurezza informatica.
- ***Comunità d'Utenza***. Un progetto di successo può fungere da catalizzatore per le esperienze e le competenze degli utenti, permettendo di ricapitalizzarle in risultati concreti e riusabili.
- ***Innovazione.*** Un'implementazione Open Source è il veicolo ideale per proporre delle implementazioni condivisibili di quanto non ancora trattato nelle specifiche *SPCoop*.

4.2 Infrastruttura a supporto

Il progetto OpenSPCoop è gestito dalla società 'Link.it' di Pisa (http://www.link.it), che, oltre a essere il principale sviluppatore del codice sorgente, mantiene una infrastruttura di supporto per il progetto, necessaria per raggiungere gli obiettivi prefissati dal progetto, tra cui una comunità d'utenza che discute, propone, critica, permette insomma la crescita di OpenSPCoop. In questa sezione saranno presentati i vari componenti dell'infrastruttura a disposizione di utenti e sviluppatori interessati a collaborare al supporto e allo sviluppo del progetto.

4.2.1 Sito Web

OpenSPCoop dispone di un sito web, che funge da punto di accesso al progetto, ed è raggiungibile all'indirizzo http://openspcoop.org. Il sito web è suddiviso in varie sezioni. La Figura 4-1 illustra una pagina web del sito elencando nel menù della parte sinistra le varie sezioni accessibili da un utente.

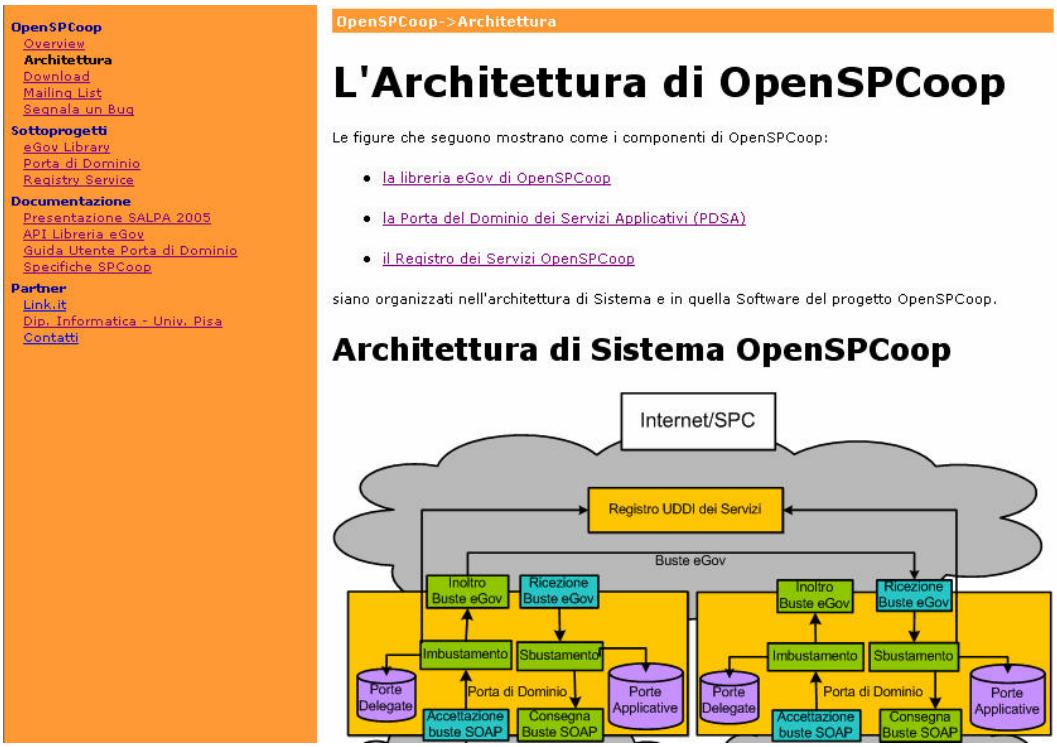


Figura 4-1, Grafica del sito 'http://openspcoop.org'

Nella **sezione 'OpenSPCoop'**, è possibile accedere ad una pagina di Overview dove si può notare come il progetto sia pienamente attivo ed in crescita, visto i brevi intervalli di tempo che passano tra il rilascio di due versioni successive del prodotto. Per ogni versione sono forniti dei file di ChangeLog che indicano le modifiche e/o le aggiunte effettuate nel codice sorgente del prodotto.

In questa sezione è possibile anche ottenere una visione dell'architettura del progetto OpenSPCoop, sia dal punto di vista architetturale che dal punto di vista della struttura del software, accedendo alla voce 'architettura'. Attraverso la voce 'download', invece, si accede alla sezione per l'accesso al codice sorgente, che è distribuito sia tramite archivi binari completi, che ne permettono un uso immediato, sia tramite accesso diretto al server cvs che contiene tutte le varie versioni del software in formato esclusivamente sorgente.

Tramite la voce ‘mailing-list’, si accede ad una sezione che permette di iscriversi alle varie liste di discussione del progetto: Annunci, Utenti e Sviluppatori. Infine la voce ‘Segnala un Bug’ permette di segnalare eventuali Bug scoperti all’interno del codice del progetto. Per ulteriori dettagli si rimanda al paragrafo 4.2.2 e 4.2.4

Nella **sezione ‘Sottoprogetti’** del sito è possibile accedere alla descrizione dei tre sottoprogetti attivi parte del progetto OpenSPCoop:

- Libreria e-Gov. In questa sezione viene fornito un esempio di utilizzo della libreria.
- Porta di Dominio. Viene illustrata l’architettura della porta di dominio e il compito di ogni nodo che ne forma la struttura.
- Registro dei servizi. Viene descritta l’attuale implementazione del registro dei servizi OpenSPCoop.

Nella **sezione ‘Documentazione’** è invece accessibile la documentazione del progetto. In particolare è possibile ottenere una guida utente del progetto, le API della libreria e-Gov disponibile ed i lucidi della presentazione del progetto OpenSPCoop effettuata al convegno Open Source SALPA. Inoltre è possibile scaricare le specifiche CNIPA, fonte principale dell’implementazione OpenSPCoop.

Infine nella **sezione ‘Partner’** è possibile accedere ai siti web dei partner del progetto.

4.2.2 Mailing List

Uno degli obiettivi del progetto OpenSPCoop è il coinvolgimento di esperti della specifica SPCoop interessati a scambiarsi informazioni, suggerimenti e critiche utili alla crescita del progetto. Questo obiettivo viene realizzato attraverso la creazione di una comunità elettronica realizzata tramite alcune Mailing List accedibili dal sito di OpenSPCoop all’indirizzo:

http://www.openspcoop.org/openspcoop/jsp/index.jsp?sel=mail.

Un utente può essere interessato al progetto con diverse prospettive. Un **utente** del prodotto è interessato solo a discussioni che trattano il puro utilizzo del prodotto e non gli aspetti tecnici che stanno alla base del prodotto stesso. Uno **sviluppatore** è interessato a discussioni tecniche che riguardano lo sviluppo del codice sorgente, la portabilità e l'architettura di OpenSPCoop. Infine si può essere interessati al progetto, senza però voler partecipare attivamente alla sua crescita. In questo caso si può essere interessati esclusivamente a notifica di annunci relativi a novità importanti sul progetto.
Per andare incontro a tutte queste esigenze, OpenSPCoop dispone di tre diverse mailing list, le quali rispecchiano gli attori precedentemente descritti (Annunci, Utenti, Sviluppatori). Per iscriversi alle liste di proprio interesse è necessario fornire la propria e-mail. A questo punto il moderatore della lista deve accettare l'iscrizione. Quando l'iscrizione è avvenuta, l'iscritto riceverà tutte le e-mail che sono scambiate sulla lista interessata, ed inoltre può anche partecipare attivamente rispondendo o proponendo quesiti agli altri iscritti.
Come sarà evidenziato nel paragrafo 4.3.1, l'area di mailing list del sito si sta sviluppando con un certo successo. Per questo motivo, di recente, sono stati implementati anche altri servizi infrastrutturali per facilitare una rapida visione/modifica dei file sorgenti (CVS, paragrafo 4.2.3) e la pubblicazione e la discussione di bug o nuove feature relative al software (BugZilla, paragrafo 4.2.4).

4.2.3 CVS

Il Concurrent Versions System (CVS) [A56] è un sistema client/server che permette agli sviluppatori di raccogliere i propri progetti in un archivio centrale chiamato repository. Molti progetti opensource possiedono un proprio server CVS, usato come archivio centrale di tutto il lavoro. Quando gli sviluppatori saranno pronti, sarà scaricato parte del loro lavoro dal CVS e dopo una compressione con uno dei formati più comuni (tar, zip, gzip, ecc…) sarà rilasciata una nuova versione ufficiale del programma.

Il CVS fornisce quindi un servizio per la gestione ottimale del codice sorgente:

- gli utenti provvisti di diritti di accesso in sola lettura potranno ottenere una copia del codice sorgente;
- gli utenti sviluppatori potranno effettuare delle modifiche al codice sorgente e memorizzarle nel repository.

Il sistema garantisce il tracciamento delle modifiche effettuate, poiché insieme alle modifiche al codice sorgente viene memorizzata anche quando è stata effettuata la modifica e chi l'ha fatta. Un ulteriore servizio offerto è la garanzia di consistenza delle modifiche effettuate. Viene garantito che le modifiche effettuate da un utente non siano in conflitto con le modifiche fatte da altri utenti, provvedendo a notificare all'utente l'eventuale conflitto. Viene anche mantenuto un History dei files modificati, in modo da rendere possibile una visione di una vecchia versione di un file sorgente.

La gestione del codice, nel progetto OpenSPCoop, viene realizzata con un particolare prodotto che supporta il protocollo CVS: Subversion [A57]. Per l'accesso remoto al repository sono disponibili numerosi client Java che per la loro stessa natura sono quindi disponibili sia su piattaforma Windows che Linux.

4.2.4 Bugzilla

Nel progetto OpenSPCoop viene utilizzato il prodotto Bugzilla [A58] per la gestione e segnalazione dei malfunzionamenti (bug) riscontrati nel programma da parte degli utenti e degli sviluppatori. Un bug, nel caso specifico di OpenSPCoop, identifica un malfunzionamento presente all'interno di un modulo applicativo o funzionale dell'architettura.

Bugzilla è un prodotto utilizzato anche nei progetti Mozilla (browser Web come Firefox e Mozilla, client di posta elettronica come Thunderbird ed altri) e soprattutto è un prodotto a codice aperto (open source).

Possiede le seguenti caratteristiche:

- Database strutturato per ottimizzare le performance e la scalabilità
- Un query tool avanzato che permette di mantenere traccia delle query effettuate e di fare ricerche in testo libero (full-text searches)
- Possibilità di integrare funzioni di notifica attraverso email
- Gestione dei profili utenti e preferenze di mail
- Funzioni di autenticazione basato su strumenti LDAP [A54, A55]
- Sistema di autorizzazione per la gestione dei permessi integrato

Segnalazione e Risoluzione di un Bug. Come si può notare dalla Figura 4-2, nel prodotto Bugzilla, un bug viene inizialmente segnalato al sistema. Opzionalmente (a seconda della modalità di apertura del bug o dalla configurazione del modulo in cui il bug viene riscontrato) la segnalazione del bug può essere non attivata immediatamente, ma aperta in modalità "da confermare" e viene attivata solo dopo che ha ricevuto un determinato numero di conferme o di voti. Una volta aperto, il bug deve essere assegnato ad un owner, cioè ad una risorsa che prende in carico le attività necessarie per la sua risoluzione. Una volta assegnato il bug viene preso in carico e può essere risolto. La risoluzione di un bug comporta il suo passaggio nello stato "risolto". Una volta nello stato "risolto" può essere preso in carico dalla sezione di Quality Assurance per verificare l'effettiva risoluzione e/o l'eventuale introduzione di nuovi bug. Nel primo caso al bug sarà assegnato lo stato risolto, mentre nel secondo caso il bug sarà assegnato allo stato riaperto, rientrando nel circuito di risoluzione.

Ricerca di un bug. Oltre alle funzionalità per la gestione dei bug (legati ai passaggi di stato descritti in precedenza), il sistema fornisce funzionalità di ricerca di bug e reportistica sui bug gestiti. L'interfaccia di ricerca dei bug permette di impostare un criterio di ricerca basato sullo stato, sul prodotto e su eventuali parole chiave contenute nella descrizione o nelle note di commento del bug.
Viene fornita anche una funzionalità di ricerca avanzata che permette di impostare un criterio di ricerca basato su uno qualsiasi degli elementi che caratterizzano i bug: lo stato di risoluzione, la severità del problema, la priorità, le piattaforme (hardware o sistema operativo) su cui si manifesta, eccetera.

Le funzioni per la gestione dei report permettono di estrarre i dati di gestione in formato tabellare o grafico. I criteri di selezione dei dati rappresentati nel report sono analoghi ai criteri utilizzati nelle funzioni di ricerca avanzata descritti in precedenza.

Alla url 'http://www.openspcoop.org/bugzilla/' è disponibile il sistema bugzilla offerto da OpenSPCoop.

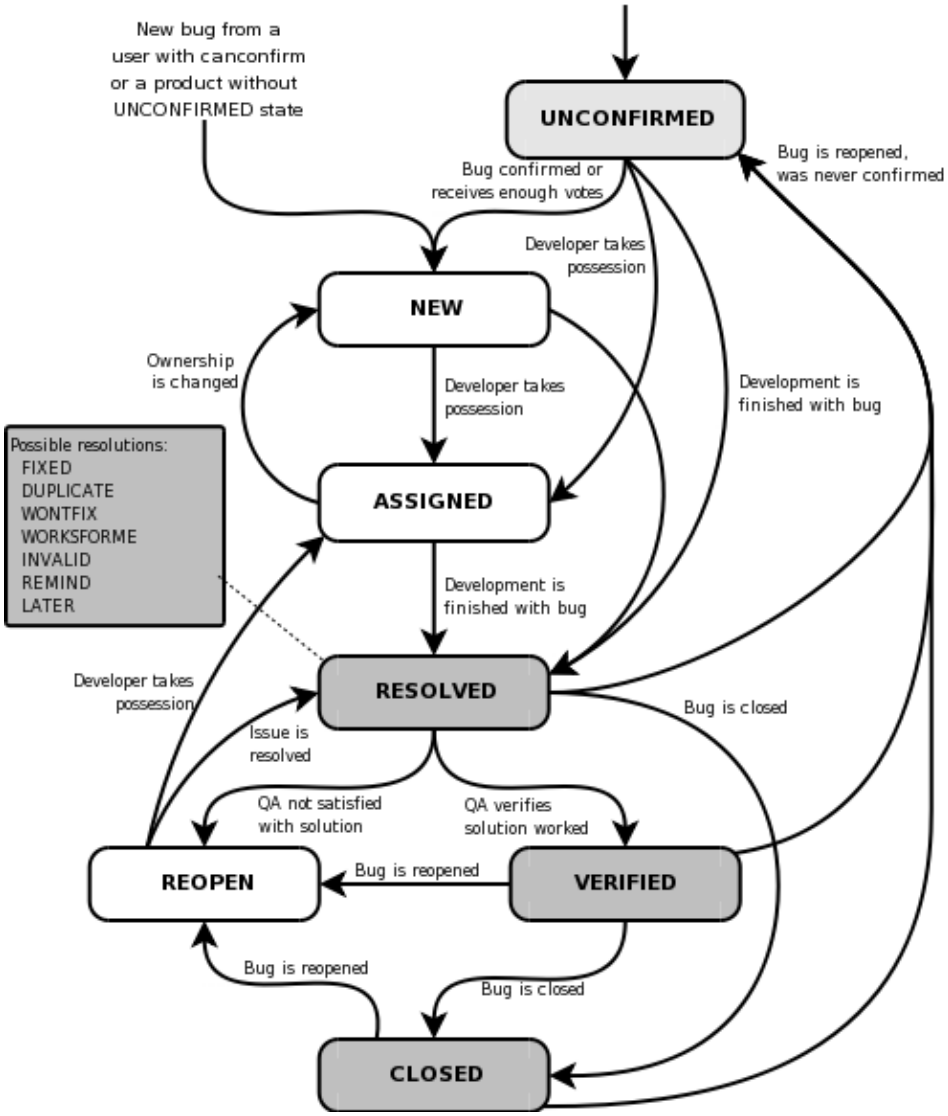


Figura 4-2, Il Ciclo di Vita di un Bug per Bugzilla

4.3 Risultati Ottenuti

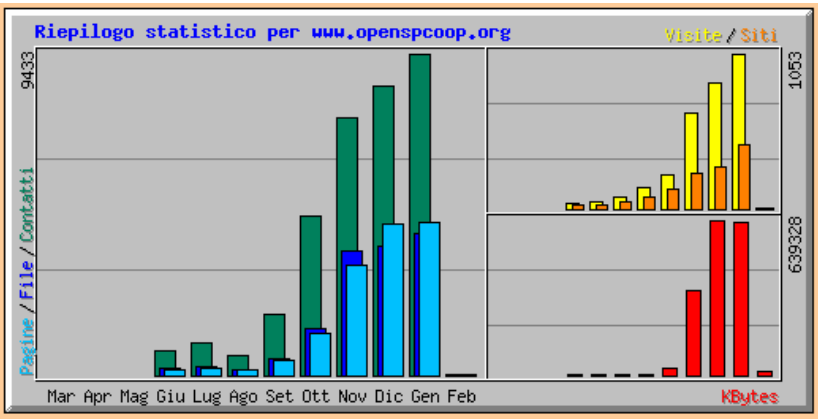
In questa sezione verranno descritti alcuni risultati ottenuti durante questi mesi (Settembre 2005 – Febbraio 2006) di vita del progetto OpenSPCoop.

4.3.1 Comunità OpenSPCoop

L'attivazione di una mailing list, accessibile all'interno del sito Web di OpenSPCoop ('http://www.openspcoop.org/openspcoop/jsp/index.jsp?sel=mail') ha permesso la creazione di una comunità di utenti e sviluppatori OpenSPCoop che hanno portato la loro esperienza sia per la crescita del prodotto (vedi paragrafo 4.3.3 sull'interoperabilità di OpenSPCoop con altre porte di dominio esistenti) sia per l'individuazione di eventuali inconsistenze o lacune presenti nella specifica rilasciata dal CNIPA (4.3.4).
La comunità di utenti e sviluppatori presenti sulla mailing-list, è formata da esperti della specifica SPCoop che in molti casi hanno già realizzato importanti progetti con altri prodotti commerciali.
Interessante è mostrare il crescente interesse nato dietro il progetto OpenSPCoop considerando che il progetto è 'online' solo dal 27 Ottobre 2005. La Figura 4-3 descrive le statistiche di accesso al sito web aggiornate al 1 Febbraio 2006.
Un altro dato interessante è mostrato nella Tabella 4-1, dove viene evidenziato il numero di download effettuati per le varie versioni del prodotto, e la data di rilascio della versione (vedi paragrafo 4.3.2).

Versione	N° Download	Data di Rilascio
0.2	46	04/11/2005
0.3	73	14/11/2005
0.4	77	02/12/2005
0.5	39	22/12/2005
0.5.1	6	16/01/2006

Tabella 4-1, Numero di download effettuati per le versioni rilasciate di OpenSPCoop



Riepilogo mensile

Mese	Media giornaliera				Totale mensile					
	Contatti	File	Pagine	Visite	Siti	KBytes	Visite	Pagine	File	Contatti
Feb 2006	41	28	31	4	5	18480	4	31	28	41
Gen 2006	304	134	144	33	437	629048	1053	4476	4171	9433
Dic 2005	274	121	143	27	285	639328	859	4445	3768	8502
Nov 2005	251	121	107	21	242	349543	648	3221	3659	7540
Ott 2005	150	44	39	7	130	29477	234	1219	1377	4674
Set 2005	59	16	14	4	85	3372	149	425	508	1789
Ago 2005	18	4	5	2	50	1112	76	170	153	575
Lug 2005	30	7	6	1	26	1643	47	199	239	938
Giu 2005	36	11	8	1	22	1132	34	153	209	698
Totali						**1673135**	**3104**	**14339**	**14112**	**34190**

Figura 4-3, Grafico degli accessi al sito Web di OpenSPCoop aggiornato al 01/02/2006

4.3.2 Versioni pubblicate del progetto OpenSPCoop

Il progetto OpenSPCoop è in continua evoluzione, cercando sia di risolvere eventuali bug riscontrati dagli utenti del prodotto, sia di completare l'implementazione della specifica rilasciata dal CNIPA.

E' interessante vedere come il progetto si è evoluto dalla prima versione rilasciata. Verranno evidenziati sia la data e la versione di rilascio che i miglioramenti e le aggiunte di funzionalità effettuate, oltre alle risoluzione dei problemi che sono stati risolti nella versione considerata.

Versione 0.1 (27 Ottobre 2005). Rilasciata una prima versione della libreria e-Gov.

Versione 0.2 (1 Novembre 2005). OpenSPCoop dispone di una prima versione della porta di dominio, con cui è possibile configurare porte delegate e porte applicative e attivare scambi di buste con profilo di collaborazione 'OneWay'. Inoltre è stata migliorata la libreria e-Gov.

Versione 0.3 (14 Novembre 2005). Viene introdotto nel prodotto OpenSPCoop un registro dei servizi OpenSPCoop 'fittizio', realizzato tramite un file xml, ed una prima versione sperimentale del registro dei servizi UDDI. Il tipo di registro dei servizi (UDDI o XML) è impostabile attraverso il file openspcoop.properties. Per quanto riguarda il registro dei servizi XML è possibile specificare sia una url che un path per specificare dove è possibile reperire il registro. Viene introdotta anche una prima gestione 'primitiva' della sicurezza, aggiungendo un controllo sull'autorizzazione da parte di un SIL-Mittente ad utilizzare una porta delegata richiesta. Un primitivo Logger per la registrazione dei messaggi diagnostici e dei tracciamenti delle buste SPCoop viene aggiunto al prodotto. Infine è stata introdotta la classe XMLUtils nella libreria e-Gov che include metodi per la creazione di messaggi XML che possono includere tracciamenti di buste, messaggi diagnostici o eccezioni applicative (Fault verso le applicazioni). Sono stati anche risolti dei bug riscontrati nella libreria e-Gov.

Versione 0.3.1 (17 Novembre 2005). Aggiunta 'primitiva' gestione del profilo di collaborazione sincrono e aggiunto utilizzo di un database relazionale, i cui parametri di accesso sono impostabili da file 'openspcoop.properties'.

Versione 0.3.2 (21 Novembre 2005). Modificato il registro dei servizi di OpenSPCoop (versione xml). Aggiunta la possibilità di definire un profilo di default utilizzato dalle azioni registrate con un servizio. Inoltre è possibile configurare azioni particolari del servizio, con un profilo diverso. E' stata aggiunta anche la possibilità di definire end-point associabili direttamente ad una specifica azione, o a un servizio, o a un service provider o infine ad una porta di dominio. Solo l'associazione dell'end-point alla porta di dominio è obbligatoria, le altre sono opzionali.

Versione 0.3.3 (24 Novembre 2005). Modificato il registro dei servizi di OpenSPCoop (versione xml). Aggiunta la possibilità di definire tipi da associare al servizio. Inoltre è possibile configurare azioni particolari del servizio, con un profilo diverso, senza dover elencare tutte le funzionalità del profilo, ma solo quelle che saranno diverse dal profilo di default. E' stata aggiunta anche la possibilità di definire end-point associabili direttamente ad un tipo, oltre che ad una specifica azione, o ad un servizio, o a un service provider o infine ad una porta di dominio. Solo l'associazione dell'end-point alla porta di dominio è obbligatoria.

E' stato modificato anche il file contente le porte delegate e le porte applicative. Nell'individuazione di una porta applicativa concorre adesso anche il tipo di un servizio. Nella definizione della porta delegata deve essere definito anche un tipo.
Infine sono stati risolti dei problemi riscontrati nella libreria e-Gov.

Versione 0.3.4 (26 Novembre 2005). Modificati gli schemi che definiscono il registro dei servizi XML e il file di configurazione contenente porte delegate e porte applicative. Gli schemi sono adesso validati correttamente da un qualunque programma che controlla uno schema XSD.

Versione 0.4 (2 Dicembre 2005). Nuova gestione degli attachments presenti nei messaggi Soap. Il contenuto (byte[]) viene memorizzato su file temporanei posti in una directory impostabile attraverso il file openspcoop.properties. I file temporanei permettono di non dover trasportare i bytes di un allegato attraverso le code dell'architettura di OpenSPCoop, e quindi permettono maggiori prestazioni.
Viene introdotta anche una prima versione della gestione dei riscontri (modalità senza PIGGYBACKING) e della gestione della consegna senza duplicati (profilo di trasmissione con attributo ALPIUUNAVOLTA). Inoltre viene fornita una History che permette la memorizzazione delle buste inviate e ricevute per una successiva riesamina.
Altra funzionalità che questa versione include è la gestione dell'attributo 'indirizzoTelematico' di un identificativo del sistema informativo locale che sta partecipando ad una cooperazione applicativa. Se presente, la eventuale busta di ritorno (contenente errori, riscontri, risposte) viene inviata all'indirizzo indicato nell'attributo, invece che direttamente nella connessione come carico HTTP Reply.
Il profilo di collaborazione sincrono, è stato migliorato ed è pienamente funzionante per quanto riguarda lo scambio di buste e-Gov tra porte di dominio. Non viene ancora gestito il protocollo SIL-to-PdD di OpenSPCoop.
Nella libreria e-Gov è stata introdotta la possibilità di differenziare il tipo di validazione effettuato su di una busta SPCoop, tra una validazione rigorosa secondo lo schema XSD della busta e-Gov e una validazione che offre prestazioni maggiori, a discapito però di una validazione che non garantisce la piena compatibilità con lo schema XSD.

E' stata anche migliorata la gestione dei log effettuata tramite Log4j ed utilizzando un file 'logger.log4j.properties' in cui sono configurati due logger, uno per i tracciamenti ed uno per i msgDiagnostici, a cui, per ognuno, è possibile associare ogni tipo di appender possibile fornito dalla tecnologia Log4j.

Versione 0.5 (22 Dicembre 2005). E' stato aggiunto un contenitore di threads di servizi che attualmente include il solo thread 'RiscontriScaduti', che si occupa di re-inviare una busta in caso di un suo riscontro non ricevuto dopo un determinato intervallo di tempo impostabile da file 'openspcoop.properties'.

Inoltre è stata risolta una lacuna dovuta alla non possibilità di definire il nome della classe del driver JDBC utilizzato all'interno del file 'openspcoop.properties'. Adesso la definizione del nome è possibile e questo dovrebbe permettere di utilizzare openspcoop con un qualsiasi database relazionale.

Il progetto OpenSPCoop ha la seguente struttura:

- src. Sorgenti del prodotto
- deploy. File necessari per una compilazione ed installazione del prodotto.
- doc. Documentazione relativa al prodotto (API ecc…)
- example. Aggiunti diversi client d'esempio per testare le funzionalità fornite dalla porta di dominio.

E' stato introdotto inoltre un ambiente di compilazione ed installazione. Il file 'build.xml' definisce Target che permettono di effettuare i seguenti aspetti:

- **Compilazione.** ('ant compile') Vengono generate le directory 'dist' e 'build'. In 'dist' sono presenti due file jar, 'openspcoop_egov.jar' e 'openspcoop_pdd.jar' che contengono rispettivamente la libreria e-Gov e la libreria della porta di dominio. Questi andranno installati sotto Axis per creare i Web Services necessari alla porta di dominio OpenSPCoop. Inoltre viene generato il file 'OpenSPCoopMDB.jar' che contiene la definizione dell'architettura di OpenSPCoop, e di cui andrà eseguito il deploy nell'application server. Nella directory 'build' sono presenti i file.class che sono stati utilizzati per produrre i file.jar precedentemente descritti.
- **Installazione**. ('ant install') I file.jar prodotti dalla compilazione sono installati nell'application server e nel contesto di Axis.
- **Deploy dei servizi axis** ('ant axis'). Effettua il deploy dei servizi axis.

- **Generazione della documentazione API** ('ant api'). Le api saranno generate nelle cartelle 'doc/api', 'doc/egov/api' e 'doc/pdd/api'.
- **Pulizia di precedenti operazioni** ('ant clean').

E' stata rimodellata, infine, la gestione della consegna del carico applicativo contenuto in una busta SPCoop pervenuta alla porta di dominio. E' stata aggiunta la possibilità di consegna su coda JMS con la possibilità di effettuare lo sbustamento Soap e la sola consegna del contenuto del SoapBody. E' possibile inoltre scegliere la coda dove effettuare la consegna ('queue' o 'topic') e il tipo di messaggio JMS di consegna ('TextMessage' e 'BytesMessage').

Versione 0.5.1 (16 Gennaio 2001). E' stata ridefinita la gestione del DB e delle code JMS per utilizzare la tecnologia JCA interna all'application server e sfruttare quindi un pool di connessione verso il DB e il provider JMS.

4.3.3 Interoperabilità con altre porte di dominio esistenti

La specifica CNIPA descrive esattamente come le buste SPCoop possono essere scambiate tra porte di dominio durante una cooperazione applicativa che avviene tra sistemi informativi. In questo contesto, una qualsiasi implementazione di una porta di dominio dovrebbe riuscire (in teoria) a dialogare con un'implementazione diversa, magari realizzata da un'altra azienda produttrice. Siamo in un contesto di interoperabilità fra porte di dominio.

OpenSPCoop, durante i primi rilasci, versioni 0.1, 0.2 e prime 0.3 non era interoperabile con porte di dominio realizzate da altre aziende quali Microsoft e Oracle e questo fatto comportò una serie di interazioni sulla mailing-list degli sviluppatori per riuscire a fare evolvere il progetto verso una piena interoperabilità. Ad esempio un problema importante fu l'adeguamento del registro dei servizi verso una flessibilità maggiore per l'accesso ai servizi. Inizialmente ogni servizio non disponeva di un punto di accesso al servizio stesso (es. URL) ma era associato ad una porta di dominio la quale disponeva di un 'entry point' dove doveva essere spedita la busta e-Gov contenente la richiesta di servizio.

Questa versione del registro, era stata così implementata poiché ogni servizio deve essere gestito da una porta di dominio, come descritto nella specifica SPCoop. Purtroppo, altre porte commerciali non forniscono una vera e propria porta di dominio da porre davanti alle applicazioni, ma solo una 'libreria' da utilizzare direttamente con le applicazioni. In questo contesto, chiaramente, ogni applicazione possiede un entry point diverso. Le discussioni sulla mailing-list hanno causato quindi l'evoluzione del registro dei servizi in una versione meno restrittiva che permettesse l'associazione di un entry point in una modalità gerarchica a partire dal tipo di servizio, passando per l'azione di un servizio e per il servizio stesso, fino ad arrivare alla porta di dominio che gestisce il sistema che eroga il servizio (vedi paragrafo 3.4).

Dopo il rilascio della versione 0.4 sulla mailing-list sono comparsi degli annunci da parte di sviluppatori che utilizzano porte di dominio commerciali, che evidenziano la piena interoperabilità della porta di dominio OpenSPCoop con le porte utilizzate. Allo stato attuale, il progetto OpenSPCoop è stato certificato interoperabile con il prodotto di Oracle e con quello di Microsoft da uno sviluppatore che stava già utilizzando i prodotti SPCoop di Oracle. Inoltre vi è stata una installazione di OpenSPCoop in un contesto di cooperazione applicativa effettuata con un altra porta di dominio rilasciata dall'azienda HP che ha passato il test di interoperabilità.

4.3.4 Quesiti sulla specifica rilasciata dal CNIPA

Durante l'implementazione della porta di dominio e degli altri componenti del progetto OpenSPCoop sono sorti diversi quesiti che evidenziavano alcune lacune della specifica rilasciata dal CNIPA. Alcuni dubbi sono stati discussi solamente all'interno della mailing-list, altri sono stati rediretti, dopo una discussione interna, al CNIPA. E' possibile citare ad esempio, un problema riscontrato sull'utilizzo di un namespace che comportava un fallimento di interoperabilità tra diverse implementazioni delle porte di dominio. Una e-mail spedita direttamente al CNIPA risolse ogni dubbio.

Un'altro esempio di inconsistenza della specifica, scoperta durante un test di interoperabilità tra la porta di dominio Oracle e quella OpenSPCoop consiste nella nomenclatura, da utilizzare per gli attori coinvolti in SPCoop, specificata dai documenti CNIPA. Gli esempi e la documentazione delle specifiche riportano un utilizzo di identificativi separati da punti, trattini, virgole ecc... Questo fatto risulta però inconsistente con la definizione dello schema XSD della busta e-Gov, per quanto riguarda alcuni campi che contengono proprio gli identificativi di alcuni soggetti SPCoop. Lo schema XSD impone dei valori che non possono presentare punti, virgole, ecc..

Una discussione, ancora aperta, è nata anche sull'utilizzo dei riscontri nel contesto di una cooperazione applicativa; sono stati evidenziati dubbi e lacune sulla modalità di utilizzo. Altre discussioni sul protocollo e-Gov, come ad es. la gestione del profilo asincrono, sono state attivate nella mailing-list.

I messaggi scambiati sulla mailing-list di OpenSPCoop, evidenziano comunque che uno degli obiettivi del progetto sta riuscendo, e cioè di diventare uno dei punti di riferimento per discutere le diverse interpretazioni delle specifiche SPCoop rilasciate dal CNIPA.

5. Conclusione e Sviluppi Futuri

In questa tesi è stata presentato il processo tramite il quale è stato perseguito l'obiettivo di trasformare in un prodotto finito e in un progetto Open Source di successo un'architettura di massima, precedentemente progettata, di un framework aderente alle specifiche *SPCoop* per la *Cooperazione Applicativa nella Pubblica Amministrazione Italiana* recentemente rilasciate dal CNIPA [A51, A52].

Risultati Ottenuti

Passare dal progetto di massima di OpenSPCoop alla sua realizzazione ha richiesto significative revisioni dell'architettura precedentemente progettata, anche in funzione del fatto che durante il periodo di svolgimento della tesi sono state rilasciate nuove e più complete versioni delle specifiche CNIPA, a cui l'architettura di OpenSPCoop è stata prontamente adeguata.

Una volta completata la fase di progettazione, la realizzazione del software è progredita rapidamente tramite il rilascio di diverse versioni del software, dalla prima versione del 27 Ottobre 2005 (versione 0.1) all'attuale versione 0.5.1 del 15 gennaio 2006. La versione 0.5.1 include un software stabile, portabile su qualunque DB e Application Server J2EE, capace di interoperare con altri prodotti SPCoop e utilizzato in produzione in almeno due situazioni relative a progetti critici di Pubbliche Amministrazioni Centrali.

Un fattore chiave della rapidità con cui questi risultati sono stati raggiunti si è rilevato l'approccio Open Source. Come è stato descritto nel Capitolo 4, è stata realizzata un'infrastruttura di collaborazione, fruibile dal sito http://openspcoop.org che, utilizzando varie tecnologie (sito web, server CVS per l'accesso al software, mailing list per utenti e sviluppatori, sistema di bug-report pubblicamente accessibile) ha permesso la crescita di una comunità di utenti e sviluppatori esperti della specifica SPCoop, il cui contributo è stato estremamente importante.

Già in questi pochi mesi di vita, il prodotto è stato scaricato ed utilizzato da utenti e sviluppatori di Pubbliche Amministrazioni locali e centrali e di aziende specializzate, tra le quali alcune hanno già cominciato a collaborare anche allo sviluppo del software. Il progetto OpenSPCoop è stato oggetto anche di presentazione nel convegno SALPA (Sapere libero e aperto nella Pubblica Amministrazione), organizzato a Pisa il 1 Novembre 2005, data in cui era stata rilasciata appena la versione 0.2. Da allora l'interesse per il progetto è cresciuto notevolmente come è stato mostrato nei grafici della Figura 4-3 e nella tabella Tabella 4-1 presentati nel paragrafo 4.3.1.

Sviluppi Futuri

L'attuale versione di OpenSPCoop non realizza ancora tutte le funzionalità specificate nei documenti rilasciati dal CNIPA.

La **libreria e-Gov** permette di gestire completamente la creazione e lo sbustamento di buste SPCoop, oltre alla gestione dei profili di collaborazione, dei riscontri e di un History di buste inviate/ricevute. Fornisce anche la possibilità di creare messaggi diagnostici, di errore applicativo e di tracciamento delle buste specificato nei documenti CNIPA.

Successive versioni della libreria dovrebbero fornire utility per la gestione degli elementi Collaborazione e Sequenza che si possono trovare in una busta. Inoltre dovranno supportare anche la gestione dei profili di collaborazione asincroni, una gestione degli attachments più efficiente (salvataggio degli allegati in un Database) e altri aspetti che possano migliorarne la facilità di utilizzo. Anche la gestione dell'History di buste inviate e ricevute deve essere migliorato.

La **porta di dominio** permette la creazione di porte delegate e applicative semplicemente attraverso la modifica di files XML. Gli aspetti progettuali riguardanti il protocollo di integrazione SIL-to-PdD, descritto nel paragrafo 2.2.2, sono completamente implementati lato porta applicativa. Invece per quanto riguarda la porta delegata manca l'implementazione di una consegna della risposta applicativa in modalità passiva e con l'ausilio di un servizio 'GetMessage', aspetti che dovranno essere realizzati in una successiva versione. Inoltre la porta di dominio gestisce al momento le sole funzionalità offerte dall'attuale versione della libreria e-Gov, e quindi non supporta la 'consegna in ordine' delle buste e il profilo asincrono.

Successive versioni dovranno implementare completamente la gestione degli eventi con interazione indiretta, e migliorare l'implementazione, presente ancora solo in maniera sperimentale, della gestione diretta.

Il **registro dei servizi** permette attualmente la definizione di un parziale accordo dei servizi associato ad un qualsiasi servizio erogato in un dominio di cooperazione. Per ogni azione di uno specifico tipo di servizio è possibile definire quali funzionalità e-Gov associargli e un end-point che indichi la porta di dominio che racchiude il SIL che lo eroga. Nel registro dei servizi è anche possibile definire ogni soggetto che dovrà intraprendere una cooperazione applicativa e ogni porta di dominio che gestisce i vari soggetti.

Il registro dei servizi attuale è realizzato come file XML. Successive versioni della porta di dominio dovrebbero utilizzare invece una versione UDDI che racchiuda tutti gli aspetti descritti nel paragrafo 2.2.5.

Allegato A: Schema XSD del Registro dei Servizi

```
<?xml version="1.0" encoding="utf-8" ?>
<xsd:schema xmlns="http://www.openspcoop.org/uddi/xmlregistry"
  targetNamespace="http://www.openspcoop.org/uddi/xmlregistry"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" attributeFormDefault="unqualified">

<!--
        Il registro Servizi contiene: definizione di servizi,
        identificati   parte della comunicazione, porte di dominio
-->
<xsd:element name="registro_servizi">
<xsd:complexType>
<xsd:sequence>
       <xsd:element maxOccurs="unbounded" minOccurs="0"
                                        ref="porta_applicativa"/>
       <xsd:element maxOccurs="unbounded"
                           minOccurs="0"ref="identificativo_parte"/>
        <xsd:element maxOccurs="unbounded" minOccurs="0"
                                        ref="porta_di_dominio" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>

<!--  porta_applicativa   -->
<xsd:element name="porta_applicativa">
<xsd:complexType>
<xsd:sequence>
    <xsd:element name="service_provider" maxOccurs="1" minOccurs="1">
        <xsd:complexType>
            <xsd:simpleContent>
                  <xsd:extension base="xsd:string">
                       <xsd:attribute name="tipo" type="xsd:string"
                                                    use="required" />
                  </xsd:extension>
            </xsd:simpleContent>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="servizio" maxOccurs="1" minOccurs="0">
      <xsd:complexType>
           <xsd:sequence>
               <xsd:element maxOccurs="unbounded" minOccurs="1"
                                            ref="tipo_servizio" />
           </xsd:sequence>
          <xsd:attribute name="valore" type="xsd:string" use="required" />
      </xsd:complexType>
    </xsd:element>
    <xsd:element maxOccurs="unbounded" minOccurs="0" ref="azione" />
    <xsd:element maxOccurs="1" minOccurs="1" ref="default" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>

<!--  tipo_servizio  -->
<xsd:element name="tipo_servizio">
<xsd:complexType>
    <xsd:sequence>
       <xsd:element name="end_point" type="xsd:string"
                                maxOccurs="1" minOccurs="0" />
    </xsd:sequence>
    <xsd:attribute name="valore" type="xsd:string" use="required" />
</xsd:complexType>
</xsd:element>
```

```
<!--  azione  -->
<xsd:element name="azione">
<xsd:complexType>
<xsd:sequence>
   <xsd:element name="profilo_asincrono_asimmetrico"
                                maxOccurs="1" minOccurs="0">
      <xsd:complexType>
         <xsd:attribute name="utilizzo" type="xsd:boolean" use="required"/>
       </xsd:complexType>
   </xsd:element>
   <xsd:element name="profilo_asincrono_simmetrico"
                                maxOccurs="1" minOccurs="0">
      <xsd:complexType>
         <xsd:attribute name="utilizzo" type="xsd:boolean" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="profilo_sincrono" maxOccurs="1" minOccurs="0">
      <xsd:complexType>
         <xsd:attribute name="utilizzo" type="xsd:boolean" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="profilo_oneway" maxOccurs="1" minOccurs="0">
      <xsd:complexType>
         <xsd:attribute name="utilizzo" type="xsd:boolean" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="collaborazione" maxOccurs="1" minOccurs="0">
      <xsd:complexType>
         <xsd:attribute name="utilizzo" type="xsd:boolean" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="ordine_consegna" maxOccurs="1" minOccurs="0">
      <xsd:complexType>
         <xsd:attribute name="utilizzo" type="xsd:boolean" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="inoltro_senza_duplicati" maxOccurs="1" minOccurs="0">
      <xsd:complexType>
         <xsd:attribute name="utilizzo" type="xsd:boolean" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="conferma_ricezione" maxOccurs="1" minOccurs="0">
      <xsd:complexType>
         <xsd:attribute name="utilizzo" type="xsd:boolean" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="scadenza" maxOccurs="1" minOccurs="0">
      <xsd:complexType>
           <xsd:attribute name="minuti" type="xsd:integer" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="end_point" type="xsd:string"
                    maxOccurs="1" minOccurs="0" />
</xsd:sequence>
<xsd:attribute name="valore" type="xsd:string" use="required" />
</xsd:complexType>
</xsd:element>
```

```
<!--  default  -->
<xsd:element name="default">
<xsd:complexType>
<xsd:sequence>
   <xsd:element name="azioni_permesse" maxOccurs="1" minOccurs="0">
      <xsd:complexType>
            <xsd:attribute name="valore" type="xsd:string" />
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="profilo_asincrono_asimmetrico"
                                 maxOccurs="1" minOccurs="1">
      <xsd:complexType>
         <xsd:attribute name="utilizzo" type="xsd:boolean" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="profilo_asincrono_simmetrico"
                                 maxOccurs="1" minOccurs="1">
      <xsd:complexType>
         <xsd:attribute name="utilizzo" type="xsd:boolean" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="profilo_sincrono" maxOccurs="1" minOccurs="1">
      <xsd:complexType>
         <xsd:attribute name="utilizzo" type="xsd:boolean" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="profilo_oneway" maxOccurs="1" minOccurs="1">
      <xsd:complexType>
         <xsd:attribute name="utilizzo" type="xsd:boolean" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="collaborazione" maxOccurs="1" minOccurs="1">
      <xsd:complexType>
         <xsd:attribute name="utilizzo" type="xsd:boolean" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="ordine_consegna" maxOccurs="1" minOccurs="1">
      <xsd:complexType>
         <xsd:attribute name="utilizzo" type="xsd:boolean" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="inoltro_senza_duplicati" maxOccurs="1" minOccurs="1">
      <xsd:complexType>
         <xsd:attribute name="utilizzo" type="xsd:boolean" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="conferma_ricezione" maxOccurs="1" minOccurs="1">
      <xsd:complexType>
         <xsd:attribute name="utilizzo" type="xsd:boolean" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="scadenza" maxOccurs="1" minOccurs="0">
      <xsd:complexType>
           <xsd:attribute name="minuti" type="xsd:integer" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="end_point" type="xsd:string"
                                      maxOccurs="1" minOccurs="0" />
</xsd:sequence>
<xsd:attribute name="utilizzo_senza_azione" type="xsd:boolean"
                                                  use="required" />
</xsd:complexType>
</xsd:element>

<!--  identificativo_parte  -->
<xsd:element name="identificativo_parte">
<xsd:complexType>
```

```
<xsd:sequence>
   <xsd:element name="service_provider" maxOccurs="1" minOccurs="1">
      <xsd:complexType>
          <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                   <xsd:attribute name="tipo" type="xsd:string"
                                                    use="required"/>
                 </xsd:extension>
            </xsd:simpleContent>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="end_point" type="xsd:string"
                                maxOccurs="1" minOccurs="0" />
   <xsd:element name="id_porta_di_dominio" type="xsd:string"
                                maxOccurs="1" minOccurs="1" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>


<!--  PortaDiDominio  -->
<xsd:element name="porta_di_dominio">
<xsd:complexType>
<xsd:sequence>
   <xsd:element name="nome" type="xsd:string" maxOccurs="1" minOccurs="0"/>
   <xsd:element name="id" type="xsd:string" maxOccurs="1" minOccurs="1"/>
   <xsd:element name="end_point" type="xsd:string"
                                maxOccurs="1" minOccurs="1"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>


</xsd:schema>
```

Allegato B:
Schema XSD di Porte Delegate e Applicative

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema xmlns="http://www.openspcoop.org/pdd/config/xml"
   targetNamespace="http://www.openspcoop.org/pdd/config/xml"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified" attributeFormDefault="unqualified">
<!--
   openspcoop contiene: PorteDelegate, PorteApplicative, SIL, PdDInfo
 -->
<xsd:element name="openspcoop">
<xsd:complexType>
<xsd:sequence>
   <xsd:element maxOccurs="unbounded" minOccurs="0" ref="porta_delegata" />
   <xsd:element maxOccurs="unbounded" minOccurs="0"
                                ref="porta_applicativa"/>
   <xsd:element maxOccurs="1" minOccurs="1" ref="porta_di_dominio" />
   <xsd:element maxOccurs="unbounded" minOccurs="0" ref="sil" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>

<!--   PortaDelegata    -->
<xsd:element name="porta_delegata">
<xsd:complexType>
<xsd:sequence>
   <xsd:element name="service_provider" maxOccurs="1" minOccurs="0">
      <xsd:complexType>
          <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                    <xsd:attribute name="tipo" type="xsd:string"
                                                    use="required"/>
                  </xsd:extension>
              </xsd:simpleContent>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="servizio" maxOccurs="1" minOccurs="0">
      <xsd:complexType>
          <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                      <xsd:attribute name="tipo" type="xsd:string"
                                                    use="required" />
                  </xsd:extension>
              </xsd:simpleContent>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="azione" type="xsd:string"
                          maxOccurs="unbounded" minOccurs="0" />
    <xsd:element maxOccurs="1" minOccurs="0" ref="consegna" />
    <xsd:element name="sil_mittente" maxOccurs="unbounded" minOccurs="1" />
</xsd:sequence>
<xsd:attribute name="url" type="xsd:string" use="required" />
<xsd:attribute name="tipo" use="required">
             <xsd:simpleType>
                   <xsd:restriction base="xsd:string">
                         <xsd:enumeration value="static" />
                         <xsd:enumeration value="url" />
                         <xsd:enumeration value="content" />
                   </xsd:restriction>
             </xsd:simpleType>
</xsd:attribute>
</xsd:complexType>
</xsd:element>
```

```
<!--   PortaApplicativa   -->
<xsd:element name="porta_applicativa">
<xsd:complexType>
<xsd:sequence>
    <xsd:element name="service_provider" maxOccurs="1" minOccurs="1">
        <xsd:complexType>
             <xsd:simpleContent>
                  <xsd:extension base="xsd:string">
                       <xsd:attribute name="tipo" type="xsd:string"
                                                    use="required" />
                  </xsd:extension>
             </xsd:simpleContent>
         </xsd:complexType>
    </xsd:element>
    <xsd:element name="servizio" maxOccurs="1" minOccurs="0">
             <xsd:complexType>
             <xsd:simpleContent>
                  <xsd:extension base="xsd:string">
                       <xsd:attribute name="tipo" type="xsd:string"
                                                   use="required" />
                  </xsd:extension>
             </xsd:simpleContent>
           </xsd:complexType>
     </xsd:element>
     <xsd:element name="azione" type="xsd:string"
                               maxOccurs="1" minOccurs="0" />
     <xsd:element maxOccurs="1" minOccurs="1" ref="consegna" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>

<!--  Consegna  -->
<xsd:element name="consegna">
<xsd:complexType>
<xsd:sequence>
    <xsd:element name="http" maxOccurs="1" minOccurs="0">
         <xsd:complexType>
            <xsd:attribute name="url" type="xsd:string" use="required" />
            <xsd:attribute name="sbustamento_soap" type="xsd:boolean"
                           use="required" />
          </xsd:complexType>
    </xsd:element>
    <xsd:element name="jms" maxOccurs="1" minOccurs="0">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="context" maxOccurs="1" minOccurs="1">
            <xsd:complexType>
               <xsd:attribute name="initial_context_factory"
                              type="xsd:string" use="required" />
               <xsd:attribute name="url_pkg_prefixes" type="xsd:string" />
               <xsd:attribute name="provider_url"
                    type="xsd:string" use="required" />
               <xsd:attribute name="connection_factory"
                         type="xsd:string" use="required" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="properties" maxOccurs="1" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="mittente" maxOccurs="1" minOccurs="0">
                <xsd:complexType>
                  <xsd:attribute name="to_property" type="xsd:string"
                                                    use="required" />
                </xsd:complexType>
              </xsd:element>
```

```
                <xsd:element name="tipo_mittente" maxOccurs="1"
                                                        minOccurs="0">
                  <xsd:complexType>
                    <xsd:attribute name="to_property" type="xsd:string"
                                                          use="required" />
                  </xsd:complexType>
                </xsd:element>
                 <xsd:element name="service_provider" maxOccurs="1"
                                                          minOccurs="0">
                   <xsd:complexType>
                    <xsd:attribute name="to_property" type="xsd:string"
                                                        use="required" />
                   </xsd:complexType>
                 </xsd:element>
                 <xsd:element name="tipo_service_provider" maxOccurs="1"
                                                        minOccurs="0">
                   <xsd:complexType>
                    <xsd:attribute name="to_property" type="xsd:string"
                                                        use="required" />
                   </xsd:complexType>
                 </xsd:element>
                 <xsd:element name="servizio" maxOccurs="1" minOccurs="0">
                   <xsd:complexType>
                    <xsd:attribute name="to_property" type="xsd:string"
                                                        use="required" />
                   </xsd:complexType>
                 </xsd:element>
                 <xsd:element name="tipo_servizio" maxOccurs="1"
                                                        minOccurs="0">
                   <xsd:complexType>
                    <xsd:attribute name="to_property" type="xsd:string"
                                                          use="required" />
                   </xsd:complexType>
                 </xsd:element>
                 <xsd:element name="azione" maxOccurs="1" minOccurs="0">
                                <xsd:complexType>
                                      <xsd:attribute name="to_property"
                                      type="xsd:string" use="required" />
                                </xsd:complexType>
                 </xsd:element>
               </xsd:sequence>
             </xsd:complexType>
          </xsd:element>
       </xsd:sequence>
       <xsd:attribute name="name" type="xsd:string" use="required" />
       <xsd:attribute name="tipo" use="required">
             <xsd:simpleType>
                   <xsd:restriction base="xsd:string">
                         <xsd:enumeration value="queue" />
                         <xsd:enumeration value="topic" />
                   </xsd:restriction>
             </xsd:simpleType>
       </xsd:attribute>
       <xsd:attribute name="send_as" use="required">
             <xsd:simpleType>
                   <xsd:restriction base="xsd:string">
                         <xsd:enumeration value="TextMessage" />
                         <xsd:enumeration value="BytesMessage" />
                   </xsd:restriction>
             </xsd:simpleType>
       </xsd:attribute>
       <xsd:attribute name="sbustamento_soap" type="xsd:boolean"
                                                        use="required" />
   </xsd:complexType>
  </xsd:element>
</xsd:sequence>
</xsd:complexType>
```

```
</xsd:element>

<!--  PortaDiDominio  -->
<xsd:element name="porta_di_dominio">
<xsd:complexType>
<xsd:sequence>
  <xsd:element name="nome" type="xsd:string" maxOccurs="1" minOccurs="0" />
  <xsd:element name="id" type="xsd:string" maxOccurs="1" minOccurs="1" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>

<!--  SIL  -->
<xsd:element name="sil">
<xsd:complexType>
<xsd:sequence>
       <xsd:element name="principal" type="xsd:string"
                                  maxOccurs="1" minOccurs="1"/>
       <xsd:element name="authentication" maxOccurs="1" minOccurs="0">
         <xsd:complexType>
           <xsd:attribute name="tipo" use="required">
               <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                          <xsd:enumeration value="HTTP-BASED" />
                          <xsd:enumeration value="SSL" />
                          <xsd:enumeration value="ALLOW-ALL" />
                       </xsd:restriction>
              </xsd:simpleType>
           </xsd:attribute>
           <xsd:attribute name="password" type="xsd:string" />
         </xsd:complexType>
       </xsd:element>
       <xsd:element name="provider" maxOccurs="1" minOccurs="1">
         <xsd:complexType>
          <xsd:simpleContent>
             <xsd:extension base="xsd:string">
                  <xsd:attribute name="tipo" type="xsd:string"
                                                    use="required"/>
             </xsd:extension>
          </xsd:simpleContent>
        </xsd:complexType>
      </xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>

</xsd:schema>
```

Bibliografia

Documenti rilasciati dal Centro Nazionale per l'Informatica nella Pubblica Amministrazione (CNIPA):

[CN1] SPC, "Sistema pubblico di cooperazione: Architettura, Versione 1.0", CNIPA, 25 Novembre 2004.

[CN2] SPC, "Sistema pubblico di cooperazione: Porta di Dominio, Versione 1.0", CNIPA, 14 Ottobre 2005

[CN3] SPC, "Specifiche della Busta di e-Government, Edizione 1.0", CNIPA, 21 Aprile 2004

[CN4] SPC, "Sistema pubblico di cooperazione: Busta di e-Gov, Versione 1.1", CNIPA, 14 Ottobre 2005

[CN5] SPC, "Sistema pubblico di cooperazione: Accordo di Servizio, Versione 1.0", CNIPA, 14 Ottobre 2005

[CN6] SPC, "Sistema pubblico di cooperazione: Servizi di Registro, Versione 1.0", CNIPA, 14 Ottobre 2005

Documenti rilasciati dal W3C:

[W1] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, D. Winer, "Simple Object Access Protocolo (SOAP) 1.1", W3C, 8 Maggio 2000

[W2] J. Barton, S. Thatte, H. F. Nielsen, "SOAP Messages with Attachments", W3C, 11 Dicembre 2000

[W3] E. Christensen, F. Curbera, G. Meredith, S. Weerawarena, "Web Services Description Language (WSDL) 1.1", W3C, 15 Marzo 2001

Articoli e Siti Web Consultati:

[A1] M. Bigatti, "Web Services, Soap e d'intorni", MokaByte, Maggio 2001, *"http://www.mokabyte.it/2001/05/webservices.htm"*

[A2] A. Giovannini, "SOAP e Java Integrazione di applicazioni con Java e il nuovo protocollo SOAP", MokaByte, Gennaio 2001, *"http://www.mokabyte.it/2001/01/soap.htm"*

[A3] M. Bigatti, “JAXM e SAAJ”, MokaByte, Aprile 2003, *“http://www.mokabyte.it/2003/04/jws-3.htm”*

[A4] R. Eckstein, “Introduzione a SAAJ”, 2005, *“http://java.sun.com/developer/EJTechTips/txtarchive/2005/Apr25_05_RobertE.txt”*

[A5] “SOAP”, Wikipedia, *“ http://it.wikipedia.org/wiki/SOAP”*

[A6] “How to pass SOAP Attachments with JAX-RPC Web Service”, Oracle, Settembre 2003, *“http://www.oracle.com/technology/sample_code/tech/java/codesnippet/webservices/attachment/index.html”*

[A7] F. Sommers, “Send and receive binary Web services content using SAAJ 1.2”, JavaWorld, Settembre 2003 *“http://www.javaworld.com/javaworld/jw-09-2003/jw-0912-webservices-p1.html”*

[A8] *Q. H. Mahmoud, “*Accessing and Interacting with Remote SOAP-enabled Services”, Sun, Settembre 2004, *“http://java.sun.com/developer/technicalArticles/WebServices/SOAP/index.html”*

[A9] “An introdution to SAAJ”, Sun 2005, *“http://java.sun.com/developer/EJTechTips/2005/tt0425.html”*

[A10] “SOAP with Attachments API for Java Sample Applications”, Sun 2005, “*http://java.sun.com/webservices/docs/1.6/saaj/samples.html”*

[A11] M. Bigatti, “Capire I WSDL”, MokaByte, Giugno 2001, *“http://www.mokabyte.it/2001/06/webservices.htm”*

[A12] “WSDL”, Wikipedia, “*http://it.wikipedia.org/wiki/WSDL”*

[A13] M. Bigatti, “La tecnologia UDDI”, MokaByte, Luglio-Agosto 2001, *“http://www.mokabyte.it/2001/07/webservices.htm”*

[A14] “UDDI”, Wikipedia, “*http://it.wikipedia.org/wiki/UDDI”*

[A15] A. Giovannini, “JAXP Java API for XML Parsing”, MokaByte, Maggio 2000, *“http://www.mokabyte.it/2000/05/java_xml.htm”*

[A16] A. Giovannini, “Validazione dei Dati con Java e XML”, Mokabyte, Novembre 2002, *“http://www.mokabyte.it/2002/11/jxml_validators.htm”*

[A17] “XML Schema Tutorial (XSD)”, w3schools, 2005, “*http://www.w3schools.com/schema/default.asp”*

[A18] "Validator for XML Schema REC (20010502) version", W3C, 2005, "*http://www.w3.org/2001/03/webdata/xsv"*

[A19] "XSD Schema Validator", "*http://apps.gotdotnet.com/xmltools/xsdvalidator/Default.aspx"*

[A20] "XML Schema Validator", DecisionSoft *"http://tools.decisionsoft.com/schemaValidate.html"*

[A21] "The JBoss 4 Application Server Guide", JBoss Open Source Company 2005, "*http://docs.jboss.org/jbossas/jboss4guide/r4/html/"*

[A22] "Architecture Overview: Microkernels, Services and Aspects", JBoss Open Source Company 2005, "*http://www.jboss.org/products/jbossas/architecture"*

[A23] "Il JBoss dell'impresa", Rivista 'IOProgrammo', Maggio 2005, "*http://www.ioprogrammo.it"*

[A24] A. Giovannini, R. Spazzoli, "Rassegna di application server EJB", MokaByte, Settembre 2000, *"http://www.mokabyte.it/2000/09/ejb.htm"*

[A25] "Overview Of J2EE Technology and Concepts", TUSC, *"http://www.tusc.com.au/tutorial/html/chap2.html"*

[A26] "Creating a Stateless Session Bean", TUSC, "*http://www.tusc.com.au/tutorial/html/chap3.html"*

[A27] R. Monson H. & D. Chappell, "Java Message Service, Chapter 2", O'Reilly, Dicembre 2000, *"http://www.oreilly.com/catalog/javmesser/chapter/ch02.html"*

[A28] S. Stark, "CHAPTER 6 Messaging on JBoss - JMS Configuration and Architecture", huihoo.com, *"http://www.huihoo.com/jboss/online_manual/3.2.3/Chap6.html"*

[A29] "JMS Provider", huihoo.com, *"http://www.huihoo.com/jboss/online_manual/3.0/ch08s07.html"*

[A30] R. Kalidindi, "Best practices to improve performance in JMS", PreciseJava, 2005, "*http://www.precisejava.com/javaperf/j2ee/JMS.htm"*

[A31] "Message Driven Beans and JMS", Borland, *"http://info.borland.com/techpubs/bes/v6/html_books/developersguide/mdb_jms.html"*

[A32] G. Puliti, "La specifica EJB 2.0 I Message Driven Beans", MokaByte, Dicembre 2002, "*http://www.mokabyte.it/2002/12/ejb_mdb.htm"*

[A33] R. Monson, “Chapter 13, Message Driven Beans”, O’Reilly, Settembre 2001 *“http://www.oreilly.com/catalog/entjbeans3/chapter/ch13.html”*

[A34] M. Riveill, “Message Driven Beans”, I3S, *“http://rangiroa.essi.fr/cours/car/01-tp-jonas/doc/MsgDrvBean.html”*

[A35] “Creating a Message Driven Bean”, TUSC, *“http://www.tusc.com.au/tutorial/html/chap7.html”*

[A36] “Message Driven EJBs”, Bea, *“http://e-docs.bea.com/wls/docs81/ejb/message_beans.html”*

[A37] A. Giovannini, “Apache Axis Il SOAP per Java”, MokaByte, 7 Settembre 2002, “*http://www.mokabyte.it/2002/09/axis.htm”*

[A38] “Creating Web Services ”, TUSC, *“http://www.tusc.com.au/tutorial/html/chap9.html”*

[A39] “Axis Architecture Guide”, Apache Software, 2005, *“http://ws.apache.org/axis/java/architecture-guide.htm”l*

[A40] “Axis: web services in Java”, JavaStaff.com, 9 Settembre 2005, *“http://www.javastaff.com/article.php?story=20050909171653855”*

[A41] “Web Service”, Wikipedia, *“http://it.wikipedia.org/wiki/Web_Service”*

[A42] M. Bigatti, “Domande e Risposte sui Web Services”, MokaByte, Gennaio 2004, *“http://www.mokabyte.it/2004/01/jws-faq.htm”*

[A43] “Apache Ant 1.6.5 Manual”, Apache, *“http://ant.apache.org/manual/”*

[A44] A.J.S. Mills,“ANT Tutorial”, University Of Birmingham 2005, *“http://supportweb.cs.bham.ac.uk/documentation/tutorials/docsystem/build/tutorials/ant/ant.html”*

[A45] S. Rossini, A. D’Angeli,“ Pratiche di sviluppo del software”, Mokabyte, 7 Settembre 2004, *“http://www.mokabyte.it/2004/09/softwaredev-3.htm”*

[A46] ”Log4j documentation”, Logging Services, *“http://logging.apache.org/log4j/docs/documentation.html”*

[A47] G. Puliti, ”Log di Applicazioni J2EE”, MokaByte, Maggio 2005, *“http://www.mokabyte.it/2005/05/log4j.htm”*

[A48] M. Chauhuan,”Logging with Log4j - An Efficient Way to Log Java Applications”, Developer.com, *“http://www.developer.com/java/ent/article.php/10933_3097221_3”*

[A49] A.J.S. Mills,"Log4j", University Of Birmingham 2005, *"http://supportweb.cs.bham.ac.uk/documentation/tutorials/docsystem/build/tutorials/log4j/log4j.html"*

[A50] D. M. Sosnoski "JiBX Binding Tutorial", 2005, *"http://jibx.sourceforge.net/tutorial/binding-tutorial.html"*

[A51] CNIPA, "Servizi di interoperabilità evoluta e cooperazione applicativa", *"http://www.cnipa.gov.it/site/it-IT/Attivit%c3%a0/Grandi_reti_della_PA/Sistema_Pubblico_di_Connettivit%c3%a0_(SPC)/Servizi_di_interoperabilit%c3%a0_evoluta_e_cooperazione_applicativa/#set1"*

[A52] CNIPA, "Chi Siamo", *"http://www.cnipa.gov.it/site/it-IT/Il_Centro_Nazionale/chi_siamo/"*

[A53] CNIPA, "Sistema Pubblico di Connettivita (SPC)", "*http://www.cnipa.gov.it/site/it-IT/Attivit%c3%a0/Grandi_reti_della_PA/Sistema_Pubblico_di_Connettivit%c3%a0_(SPC)/"*

[A54] M. Ferrante, T. Podesta, "Tutorial LDAP", Università di Genova, *"http://www.garr.it/ws5/pdf/LDAP.pdf"*

[A55] "What is LDAP?", Gracion Software, "*http://www.gracion.com/server/whatldap.html"*

[A56] "CVS - Concurrent Versions System", GNU, *"http://www.nongnu.org/cvs/"*

[A57] "SUBVERSION", Tigris.org, *"http://subversion.tigris.org/"*

[A58] "BugZilla Home Page", Mozilla, "*http://www.bugzilla.org/"*

[A59] "DTD Tutorial", W3Schools, "*http://www.w3schools.com/dtd/default.asp*"

Libri di testo consultati:

[T1] E. R. Harold, "Programmare in rete con JAVA", Jackson Libri 1998.

[T2] J. Hunter, W. Crawford, "Java Servlet Programming", O'Reilly 2001.

[T3] M. Ferrero, "Laboratorio di SQL", Apogeo 2000.

Software utilizzato (Implementazione del progetto OpenSPCoop):

[S1] "Application Server JBoss",
http:// www.jboss.com

[S2] "Web Services - Axis"
http://ws.apache.org/axis/

[S3] "JiBX – Binding XML to Java Code",
http://jibx.sourceforge.net/

[S4] "Apache ANT"
http://ant.apache.org/

[S5] "Log4j - Logging services",
http://logging.apache.org/log4j/docs/

[S6] "Database PostgreSQL",
http://www.postgresql.org/

[S7] "Database MySQL",
http://www.mysql.com/