Università degli Studi di Pisa
Facoltà di Scienze Matematiche Fisiche e Naturali
Dipartimento di Informatica

TESI DI LAUREA

La Gestione degli Accordi di Cooperazione nel progetto OpenSPCoop

Relatori
prof. Andrea Corradini
prof. Tito Flagella
Controrelatore
prof. Vincenzo Gervasi

Candidato
Aldo Lezza

Anno Accademico 2006-2007

Indice

Capitolo 1

Introduzione

L'informatizzazione della Pubblica Amministrazione (PA) è iniziata spontaneamente negli ultimi decenni grazie alla diffusione dei calcolatori nel nostro Paese. Questo processo è stato dettato inoltre dall'esigenza di migliorare le procedure che riguardano ogni tipo di pratica della PA. Come per ogni altra istituzione o azienda, anche per la PA il risparmio di tempo e l'ottimizzazione delle risorse equivale ad un notevole risparmio di denaro. Questo può auspicabilmente portare ad un minore carico fiscale per i contribuenti chiamati a finanziare l'amministrazione statale. Un risparmio può aversi anche in termini di utilizzazione e consumo di materiale: diminuzione dell'utilizzo di cancelleria, riduzione delle spese per l'archiviazione e il trasporto di documenti, possibilità di reimpiego del personale divenuto in esubero per colmare le carenze di organico nei settori deficitarii.
A livello informatico le varie amministrazioni hanno lavorato secondo le proprie esigenze e le proprie possibilità, senza una organizzazione a livello nazionale. Questo ha portato alla nascita di servizi applicativi con possibilità comunicative limitate. Si pensi ad esempio a due comuni che si accordano su un comune formato di scambio. Se un terzo comune che usa un proprio formato di dati necessita di comunicare con questi comuni deve adattarsi allo standard degli altri due. Il Ministero per l'Informatizzazione e le tecnologie

raggiunge, anche grazie all'insorgere di queste situazioni, la consapevolezza di dover governare questo processo di informatizzazione. Quindi, nel 2002 il Centro Nazionale per l'Informatizzazione nella Pubblica Amministrazione (CNIPA) ha avviato gli studi necessari per la definizione dello scenario futuro delle infrastrutture informatiche delle pubbliche amministrazioni italiane. Tramite il Decreto lgs. del 28 febbraio 2005, n. 42 dal titolo '*Istituzione del Sistema Pubblico di connettività e della Rete internazionale della pubblica amministrazione*', il CNIPA istituisce e disciplina il Sistema Pubblico di Connettività (SPC). Questo è definito come
'l'insieme di strutture organizzative, infrastrutture tecnologiche e regole tecniche, per lo sviluppo, la condivisione, l'integrazione e la circolarità del patrimonio informativo della pubblica amministrazione, necessarie per assicurare l'interoperabilità e la cooperazione applicativa dei sistemi informatici e dei flussi informativi, garantendo la sicurezza e la riservatezza delle informazioni'.
Il progetto è articolato in due fasi principali secondo due obiettivi:

- La definizione del SPC nel suo complesso, delle strutture organizzative per il suo governo, le infrastrutture tecnologiche e le regole tecniche per la fornitura dei servizi di connettività ed interoperabilità di base nel rispetto dei necessari requisiti di sicurezza;

- La definizione del modello e dei servizi di interoperabilità evoluta e cooperazione applicativa e lo sviluppo dell'architettura abilitante e delle relative regole di governo.

I lavori per la definizione del sistema sono stati avviati sin dalla metà del 2002 in collaborazione con le pubbliche amministrazioni, esperti del mondo accademico, rappresentanti delle aziende. Tra l'aprile 2004 e l'ottobre 2005 il CNIPA completa le specifiche per il Sistema Pubblico di Cooperazione [SPCOOP] attraverso una serie di documenti. Nell'aprile 2004 viene rila-

sciata 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. Nel 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. Nell'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.

Successivamente alla pubblicazione delle specifiche SPCoop da parte del CNIPA ha avuto inizio il progetto OpenSPCoop [OPENSPCOOP] che ha come obiettivo la realizzazione di un insieme di componenti Open Source aderenti a tali specifiche. Questo progetto nasce da una collaborazione sul tema della Cooperazione Applicativa avviata nel 2004 dal Dipartimento di

Informatica dell'Università di Pisa e la società Link.it di Pisa che decisero di approfondire alcuni aspetti innovativi su questo tema, emersi in progetti delle PA in cui erano rispettivamente coinvolti. Dopo un'ampia fase di analisi, svolta attraverso lo studio di vari progetti pilota tra cui il progetto CART della Regione Toscana [R12] e il Progetto SOLE dell'Emilia Romagna [R13], è emersa la proposta di un'architettura innovativa per la realizzazione di una infrastruttura compatibile con le specifiche CNIPA, che riducesse però significativamente l'impatto sui sistemi preesistenti rispetto alle soluzioni disponibili.

Il paradigma per l'integrazione dei servizi scelto per la porta di dominio OpenSPCoop è il paradigma *Web Service Mediator*. Questo paradigma parte dalla constatazione che l'ampia diffusione dell'XML e del paradigma dei Web Services permette di considerare i sistemi interni al Dominio di Servizio come già capaci o facilmente adattabili al dialogo tramite Web Services. Se si assume questa premessa, la componente di integrazione non dovrà più supportare tecnologie diverse per ogni possibile sistema legacy da interfacciare (CORBA, RMI,JMS, .NET, etc.), ma potrà essere invece un generico container che funga da mediatore dei messaggi SOAP in arrivo dai clienti interni per i servizi esterni e viceversa.

I componenti architetturali della specifica SPCoop introdotti in precedenza sono stati completamente implementati in OpenSPCoop rispettando le specifiche disponibili alla fine del 2005.

I componenti analizzati ed estesi durante questo lavoro di tesi riguardano il Registro dei Servizi e l'Accordo di Servizio. Il ruolo del Registro dei Servizi è quello di permettere la registrazione e la successiva interrogazione degli Accordi di Servizio (AS) di SPCoop. A partire dagli AS si snodano i vari riferimenti ai soggetti erogatori e fruitori, alle interfacce dei servizi erogati e fruiti, alle politiche di sicurezza ed in generale a tutto quanto riferisce alle politiche di cooperazione applicativa concordate tra i Soggetti interessati.

Successive versioni delle specifiche rilasciate hanno introdotto nuovi componenti dalle capacità più avanzate e complesse. Tra questi, il componente preso in considerazione in questa tesi è *l'Accordo di Cooperazione*[CN5].
Rispetto ad un Accordo di Servizio, un Accordo di Cooperazione definisce le relazioni di servizio in qualità di intermediario: la relazione che questo tipo di accordo è volta a creare coinvolge infatti più organizzazioni. Tra queste possiamo identificarne di tre tipi:

- Fruitrice, cioè l'organizzazione che intende ricevere la prestazione del servizio.
- Erogatrici, cioè le organizzazioni che garantiscono le prestazioni di servizio necessarie alla realizzazione dell'Accordo di Cooperazione.
- Referente, cioè l'organizzazione che si rende responsabile del corretto svolgimento dell'intera operazione concordata con l'organizzazione fruitrice.

E' chiaro quindi come l'organizzazione referente abbia un ruolo duale: essa è infatti erogatrice nei confronti dell'organizzazione fruitrice e al tempo stesso fruitrice dei servizi erogati dalle organizzazioni erogatrici. L'Accordo di Cooperazione definisce quindi una composizione tra servizi che può essere di arbitraria complessità. Ad esempio, il risultato di uno o più servizi può identificare il dato d'ingresso per un altro servizio. L'obiettivo di implementare gli Accordi di Cooperazione in OpenSPCoop ha richiesto lo studio e l'utilizzo di meccanismi di organizzazione di processi produttivi (*workflow*) per Web Service. Dopo l'analisi di vari linguaggi la scelta è ricaduta sul linguaggio WS-BPEL (Business Process Execution Language for Web Services [BPEL]) per la ricchezza dei costrutti e la semplicità d'uso. In seguito è stata effettuata una ricerca di un engine per workflow scritti in BPEL; tale ricerca ha portato alla scelta di JBPM-BPEL, un engine disponibile per l'application server JBoss che costuituisce la base del funzionamento di

OpenSPCoop. Il componente Accordo di Cooperazione è stato infine integrato nel Registro Servizi di OpenSPCoop e sono stati prodotti vari casi d'uso che ne mostrassero le modalità d'utilizzo e il funzionamento.

1.1 Obiettivi della tesi

L'obiettivo di questa tesi è quindi la progettazione e l'implementazione di un componente Accordo di Cooperazione che aderisca alle specifiche per la Cooperazione Applicativa nella Pubblica Amministrazione rilasciata dal CNlPA. Questo componente è diventato parte integrante del progetto OpenSPCoop dalla versione 1.0.beta.3. Il raggiungimento di questo obiettivo ha richiesto di affrontare i seguenti aspetti principali:

- Definizione dell'Accordo di Cooperazione a partire dalle specifiche del CNIPA.
- Progettazione dell'Accordo di Cooperazione a partire dalla definizione degli Accordi di Servizio e relativa integrazione nel Registro Servizi OpenSPCoop.
- Ricerca ed identificazione di un engine per processi produttivi automatizzabili (workflow).
- Realizzazione di strumenti per facilitare e velocizzare lo sviluppo di servizi scritti secondo lo standard di orchestrazione per Web Service WS-BPEL.
- Identificazione e realizzazione di casi d'uso per sperimentazione di WS-BPEL.
- Identificazione e realizzazione di casi d'uso di Accordi di Cooperazione.
- Realizzazione di test suite per i casi d'uso identificati.

La prima fase del lavoro è consistita nell'analisi della versione del codice esistente di OpenSPCoop con particolare riferimento agli Accordi di Servizio e al Registro Servizi per verificare la fattibilità dell'obiettivo. In una seconda fase si è passati ad analizzare le offerte tra engine BPEL per la realizzazione di servizi di tipo workflow. Come anticipato in precedenza, la scelta è ricaduta su JBPM-BPEL per la sua facile integrazione con gli strumenti già a disposizione nella piattaforma esistente. Successivamente si è passati a implementare i primi esempi nel linguaggio BPEL per definire scenari semplici del tipo

- verifica della disponibilità dell'engine sulla piattaforma (es. tipo *Hello World*)
- implementazione di servizi di elaborazione ed inoltro di messaggi SOAP semplici tra più servizi
- utilizzo di tipi XML e di query Xpath
- indicizzazione dinamica di servizi.

In ultimo, si è passati all'identificazione dei casi d'uso e alla loro relativa implementazione. Parte del codice scritto durante la tesi è stato reso fruibile, oltre che alla comunità OpenSPCoop, anche alla comunità dell'engine JBPM-BPEL.

1.2 Contenuto della tesi

Nel secondo capitolo vengono descritte le architetture orientate ai servizi (SOA) e i Web Service, nonchè alcune tecnologie sulle quali si basa lo sviluppo di applicazioni secondo questo paradigma.
Nel terzo capitolo viene descritta la specifica SPCoop del CNIPA con una descrizione sintetica dei componenti infrastrutturali e dell'architettura generale di un progetto aderente a tale specifica.

Nel quarto capitolo si descrive OpenSPCoop, la realizzazione Open Source della specifica SPCoop: sono mostrati sinteticamente i maggiori componenti presenti, con particolare attenzione al Registro Servizi che è stato oggetto di estensione e modifica durante questo lavoro di tesi.
Nel quinto capitolo si descrivono i linguaggi per workflow e il loro paradigma di specifica chiamato *linguaggio orientato ai grafi* (GOP). Il capitolo presenta il linguaggio WS-BPEL scelto per implementare la struttura di orchestrazione dei servizi SPCoop e descrive brevemente altri linguaggi di orchestrazione di Web Service, speigando le motivazioni che hanno portato alla scelta di WS-BPEL.
Nel sesto capitolo si descrive come sia stato progettato e realizzato l'Accordo di Cooperazione, la novità introdotta da questa tesi, a partire dall'Accordo di Servizio già presente in OpenSPCoop. In questo capitolo vengono ripresi concetti della specifica del CNIPA per SPCoop relativi sia agli Accordi di Servizio che agli Accordi di Cooperazione e viene spiegata la soluzione proposta per l'Accordo di Cooperazione nel Regisro Servizi di OpenSPCoop.
Nel settimo capitolo dapprima si descrive l'engine JBPM per processi produttivi e l'estensione utilizzata durante questa tesi, l'engine JBPM-BPEL. In seguito vengono mostrati casi d'uso scritti nel linguaggio WS-BPEL per familiarizzare con il linguaggio e mostrarne le potenzialità. Il capitolo si conclude con l'implementazione di casi d'uso decisi in fase di progettazione e i relativi passaggi necessari per configurare porte di dominio che richiedano Accordi di Cooperazione.
L'ultimo capitolo infine riassume gli obiettivi raggiunti e le conclusioni che ne conseguono, illustrando alcuni possibili sviluppi futuri che possono essere realizzati a partire dai risultati ottenuti in questa tesi.

Capitolo 2

Le Architetture basate su Web Service

Le architetture orientate ai servizi (SOA) e i Web Service forniscono un approccio standard per l'interoperabilità tra diverse applicazioni software in esecuzione su una grande varietà di piattaforme e/o di framework. In questa sezione verranno illustrati il modello concettuale e il contesto necessario per comprendere i Web Service e le relazione tra le componenti di questo modello.

2.1 Architetture Orientate ai Servizi (SOA)

Un sistema distribuito consiste di agenti software diversi che devono funzionare insieme per realizzare certi compiti. Inoltre, gli agenti nel sistema distribuito non operano nello stesso ambiente di esecuzione e quindi devono comunicare usando protocolli hardware/software su una rete. Questo significa che le comunicazioni in un sistema distribuito sono intrinsecamente meno veloci e affidabili di quelle che usano invocazioni dirette di codice e memoria condivisa. Questo ha importanti implicazioni perchè i sistemi distribuiti richiedono che gli sviluppatori (sia dell'infrastruttura che delle applicazioni)

tengano in considerazione la latenza della rete e gli accessi remoti, nonché problemi di concorrenza e di fallimenti parziali.
I sistemi a oggetti distribuiti sono sistemi distribuiti nei quali l'inizializzazione degli oggetti e l'invocazione dei metodi sono esposti a sistemi remoti attraverso meccanismi standard o proprietari per pubblicare le richieste attraverso i confini del sistema, marshall e unmarshall dei dati in argomento dei metodi, ecc....
I sistemi ad oggetti distribuiti tipicamente sono caratterizzati da oggetti che mantengono uno stato interno complesso richiesto per supportare i loro metodi e da una interazione tra un oggetto e un programma che lo usa rispettando un sistema di tipi di oggetti condiviso.
Una architettura orientata ai servizi (SOA) è una forma di architettura caratterizzata da:

- Visione logica: Il servizio è una visione astratta, logica dei programmi in esecuzione, database, processi di tipo business, ecc... definiti in termine di cosa fanno, in genere adempiendo a operazioni a livello business.

- Orientamento ai messaggi: il servizio è formalmente definito in termini dei messaggi scambiati tra gli agenti fornitori e gli agenti richiedenti e non in termini delle proprietà degli agenti stessi. La struttura interna di un agente, incluse le caratteristiche come il suo linguaggio di implementazione, la struttura del processo e anche la struttura del database sono deliberatamente rese astratte in una architettura orientata al servizio: usando la disciplina SOA non c'è bisogno di sapere come un agente che implementa un servizio sia costruito.

- Orientamento alla descrizione: un servizio è descritto da metadati processabili da una macchina. La descrizione esposta al pubblico contiene solo dettagli importanti per l'uso del servizio. La semantica del servi-

zio dovrebbe essere documentata, indirettamente o direttamente, dalla sua descrizione.

- Modularità: I servizi tendono ad usare un piccolo numero di operazioni con messaggi relativamente larghi e complessi
- Orientazione alla rete: I servizi tendono ad essere progettati per essere offerti attraverso la rete, sebbene questo non sia un requisito assoluto.
- Neutralità rispetto alla piattaforma: I messaggi sono inviati in maniera neutrale rispetto alla piattaforma e recapitati in formati standard attraverso le interfacce. XML è il più ovvio formato che permette di rendere questa caratteristica.

2.2 Web Service: una Tecnologia per SOA

Le architetture orientate ai servizi sono un modello di architettura informatica distribuita che è diffusa grazie alla maturità e al consolidamento degli standard e delle tecnologie Web Service[w3cArch]. E' utile comunque precisare la differenza concettuale tra architetture orientate ai servizi e Web Service:
L'architettura SOA definisce un modello concettuale di architettura informatica distribuita costituita da un insieme di sistemi autonomi che comunicano per mezzo di messaggi scambiati attraverso interfacce standardizzate.
Il termine *Web Service* ricopre invece un insieme di standard tecnologici che permettono la realizzazione di architetture SOA su larga scala, garantendo al tempo stesso l'interoperabilità e l'autonomia di implementazione dei sistemi componenti l'architettura.
In questa sezione verranno illustrati i concetti alla base dei Web Service e come si possa usarli per erogare servizi. Inoltre verranno brevemente illustrate alcune delle tecnologie che sono critiche per il funzionamento dei Web Service e il ruolo che ricoprono.

2.2.1 Cos'è un Web Service?

Un Web Service è un sistema software progettato per supportare interazione machine-to-machine su una rete. Il Web Service è descritto in un formato processabile dalle macchine (nella fattispecie WSDL). Altri sistemi interagiscono con i Web Service secondo quanto definito nella loro descrizione usando messaggi SOAP, in genere trasportati attraverso HTTP con una serializzazione XML in combinazione con altri standard legati al Web.

2.2.2 Agenti e Servizi

Un Web Service è una nozione astratta che deve essere implementata da un agente concreto. Con agente si intende un pezzo di software o di hardware capace di inviare e ricevere messaggi, mentre il servizio è la risorsa caratterizzata da un insieme astratto di funzionalità che vengono fornite. Per illustrare questa distinzione è possibile implementare un particolare Web Service usando un giorno un agente (scritto ad esempio in un linguaggio di programmazione) e un altro agente il giorno successivo (magari usando un linguaggio differente) mantenendo le stesse funzionalità. Sebbene questi agenti possano differire, il Web Service rimane lo stesso.

2.2.3 Fornitori e Richiedenti

Lo scopo di un Web Service è di fornire la descrizione dell'invocazione di un servizio tra due entità, quella fornitrice del servizio e quella richiedente. L'entità fornitrice è la persona o l'organizzazione che fornisce un agente appropriato per implementare un particolare servizio. Una entità richiedente è una persona o una organizzazione che desidera usare l'entità fornitrice del Web Service. Questa entità userà un agente richiedente per scambiare messaggi con l'agente dell'entità fornitrice. In molti casi, l'agente richiedente è colui che inizia lo scambio di messaggi. Per consistenza di trattazione, continueremo ad usare il termine *agente richiedente* per l'agente che interagisce

con l'agente fornitore anche nel caso in cui l'agente fornitore inizi lo scambio di messaggi. Per realizzare uno scambio di messaggi corretto, l'entità richiedente e l'entità fornitrice devono prima accordasi sia sulla semantica sia sui meccanismi di scambio di messaggi.

2.2.4 Descrizione del Servizio

I meccanismi di scambio di messaggi sono documentati in una descrizione di servizio (WSD). Tale descrizione è descritta in un linguaggio testuale basato su XML di nome *WSDL, Web Service Description Language*. Questa descrizione definisce i formati dei messaggi, i tipi dei dati, i protocolli di trasporto e il trasporto per i formati di serializzazione per i dati tra l'agente richiedente e l'agente fornitore. Nella descrizione inoltre troviamo una o più locazioni di rete alle quali un agente fornitore può essere invocato e può fornire alcune informazioni sul pattern di scambio di messaggi che bisogna aspettarsi. In sintesi, la descrizione del servizio rappresenta l'accordo che regola i meccanismi per l'interazione con i Web Service.

2.2.5 La Semantica

La semantica di un Web Service è la definizione del comportamento del servizio, condiviso tra gli agenti, che definisce nello specifico la risposta ai messaggi che il servizio riceve. Questo è quello che si può definire il *contratto* tra l'entità richiedente e l'entità fornitrice che riguarda le finalità e le conseguenze dell'interazione. Sebbene questo contratto rappresenti l'accordo globale tra l'entità richiedente e l'entità fornitrice su come e perchè i loro rispettivi agenti interagiranno, non deve essere necessariamente scritta o esplicitamente negoziata. Può essere implicita o esplicita, orale o scritta, processabile da macchine o no e può essere un accordo legale o informale. Mentre la descrizione del servizio rappresenta un contratto che regola i meccanismi di interazione con un particolare servizio, la semantica rappresenta

un contratto che regola i significati e il motivo di quella interazione. La linea di divisione tra questi due concetti non è necessariamente rigida. Così quanto più ricco è semanticamente un linguaggio usato per descrivere il meccanismo di interazione tanto più le informazioni essenziali possono spostarsi da una semantica informale alla descrizione di un servizio. Quando questo spostamento si verifica, la maggior parte del lavoro richiesto per raggiungere un'interazione corretta può essere automatizzato.

2.2.6 Schema di Attivazione di un Web Service

Ci sono vari modi con cui una entità richiedente può attivare e usare un Web Service. In generale, i passi che seguono possono essere identificati in quelli illustrati nella figura:

1. le entità fornitrice e richiedente diventano note l'una all'altra
2. le entità in qualche modo si accordano sulla descrizione del servizio e sui meccanismi che regoleranno l'interazione tra i rispettivi agenti
3. la semantica del servizio viene realizzata da entrambi gli agenti (e.g. il codice per processare l'input e calcolare la risposta)
4. gli agenti si scambiano i messaggi secondo il protocollo concordato nella descrizione del servizio.

Alcuni dei passaggi precedenti possono essere eseguiti in maniera automatica, altri in maniera manuale.

2.3 Tecnologie per Web Service

L'architettura Web Service coinvolge molte tecnologie stratificate e correlate. Ci sono molti modi per visualizzare queste tecnologie proprio come ci sono molti modi per costruire e usare i Web Service. La figura seguente fornisce una illustrazione di alcune di queste tecnologie.

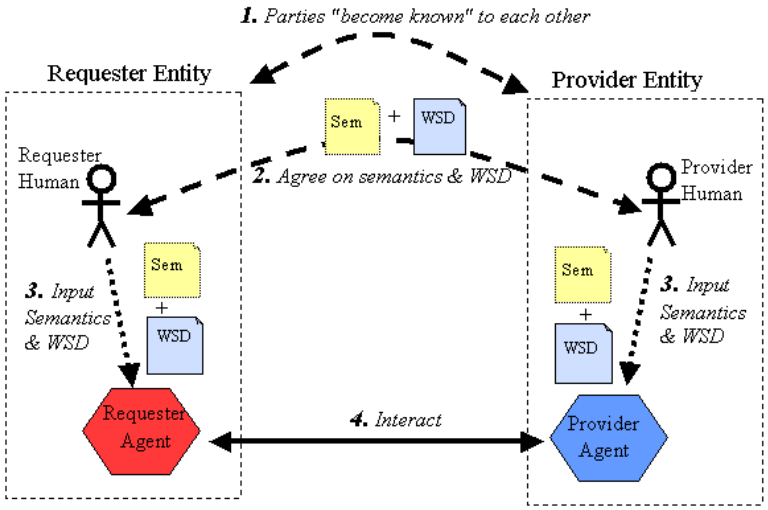

Figura 2.1: Schema di Attivazione di un Web Service.

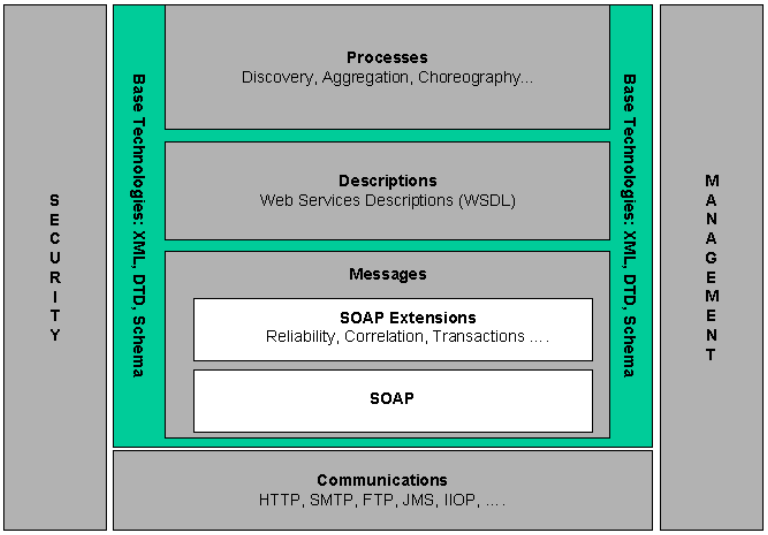

Figura 2.2: Web Service: L'organizzazione dell'architettura.

2.3.1 XML

XML (eXtensible Markup Language, ovvero 'Linguaggio di marcatura estensibile') è un meta linguaggio utilizzato per creare nuovi linguaggi, atti a descrivere documenti strutturati. XML, come HTML, utilizza dei marcatori,

detti tag, per assegnare una semantica al testo. L'estensibilità di XML si riconosce nella capacità di poter esprimere tag personalizzati. Ad esempio, se di un tipo *persona* vogliamo memorizzare solo il nome e il cognome, una istanza di questo oggetto in XML è rappresentabuile come segue:

```
<persona>
 <nome>Mario</nome>
 <cognome>Rossi</cognome>
</persona>
```

XML risolve un requisito chiave di tecnologia che appare in molti contesti. Offrendo un formato per i dati standard e al tempo stesso flessibile, XML riduce significativamente il peso di sviluppare le molte tecnologie richieste per assicurare il successo dei Web Service.
Gli aspetti rilevanti di XML, per le qualità dell'architettura, sono la sintassi di base stessa, il concetto di XML Infoset, XML schema e XML namespace.
Gli XML Infoset [INFOSET] sono formati di dati astratti e il loro compito è di fornire un insieme consistente di definizioni che altre specifiche possono usare per riferire parti di informazioni in un documento XML ben formato. Una di queste informazioni ad esempio è l'insieme dei vincoli per costruire un *element* in un documento XML: unicità del nome nel documento, namespace di riferimento, elemento padre ecc...
Gli XML Schema [SCHEMA] permettono di registrare tipi e regole dei tipi. Per fare questo, forniscono costrutti per definire le strutture, i contenuti e la semantica dei documenti XML. Ad esempio un documento che rispetti un XML Schema dovrà avere un certo numero di elementi e di ogni tipo di elementi un numero che è regolato dal XML Schema.
Gli XML Namespace [NS] permettono di definire spazi di nomi con lo stesso significato che questo termine assume nei comuni linguaggi di programmazione spesso sotto il concetto di *package*. Questo, ad esempio, permette di avere due o più elementi con il medesimo nome all'interno dello stesso documento

purchè appartengano a namespace differenti. Essendo XML un linguaggio completamente testuale permette la trasmissione attraverso la rete in maniera abbastanza flessibile. La flessibilità di scelta dei formati di serializzazione permette una interoperabilità più vasta tra gli agenti del sistema. In futuro, una codifica binaria di XML Infoset può essere un ottimo candidato per rimpiazzare la serializzazione testuale. Una codifica binaria potrebbe essere più efficiente e più adatta per interazioni machine-to-machine.

2.3.2 SOAP

SOAP (Simple Object Access Protocol)[SOAP] è un protocollo per lo scambio di messaggi tra componenti software, tipicamente per l'invocazione di Web Service ed è basato su XML. Lo scambio di messaggi SOAP permette l'esecuzione di procedure remote come accade per l'interazione con Web Service. Un messaggio SOAP si divide tipicamente in due parti:

- *Header (Intestazioni)*: in questa parte vengono racchiuse le informazioni che non appartengono ai dati della applicazione ma che sono necessarie al corretto svolgimento del protocollo. Un esempio può essere la data di invio del messaggio. L'elemento Header non è obbligatorio.
- *Body (Corpo)*: in questa parte vengono scambiati i dati relativi alle applicazioni, tipicamente in formato XML. L'elemento body è obbligatorio.

SOAP 1.1 fornisce un framework standard componibile ed estensibile per lo scambio di messaggi XML. I messaggi SOAP possono essere trasportati su una gran varietà di protocolli di rete come HTTP, SMTP, FTP, RMI/IIOP o protocolli per messaggi proprietari. Un messaggio SOAP definisce, inoltre, tre componenti opzionali: un insieme di regole di codifica per esprimere le istanze dei tipi di dati definiti dall'applicazione, una convenzione per rappresentare le chiamate a procedure remote (RPC) e ritorni e un insieme

di regole per utilizzare SOAP con HTTP 1.1. SOAP quindi rappresenta un protocollo per architetture orientate al servizio. Un messaggio SOAP rappresenta l'informazione necessaria per invocare un servizio o riflette i risultati di una invocazione di servizio e contiene le informazioni specificate nella definizione dell'interfaccia di un servizio.

2.3.3 WSDL

WSDL (Web Service Description Language) è un linguaggio per descrivere Web Service. WSDL rappresenta un formato XML per descrivere servizi di rete come un insieme di nodi basati su messaggi. WSDL descrive i Web Service iniziando con la definizione dei messaggi che devono essere scambiati tra l'agente richiedente e l'agente fornitore. I messaggi stessi sono descritti in maniera astratta e sono legati ad un concreto formato di messaggi e a un protocollo di rete ben definito. Un documento WSDL si divide tipicamente in due parti:

- *Astratta*: Le operazioni sono definite tramite lo scambio di messaggi. Operazioni e messaggi sono identificati da tipi XML.
- *Concreta*: Le operazioni astratte sono concretizzate aggiungendo loro le informazioni di rete relative al nodo su cui saranno disponibili (indirizzo, tipo di chiamata (es. RPC) e protocollo di trasporto).

Le definizioni di Web Service possono essere mappate su un qualunque linguaggio, piattaforma, modello a oggetti o sistema di messaggi. Semplici estensioni all'infrastruttura Internet esistente possono implementare Web Service per l'interazione via browser o direttamente dall'interno dell'applicazione. L'applicazione può essere implementata usando COM, JMS, CORBA, COBOL e un qualsiasi numero di soluzioni proprietarie di integrazione. Fin quando mittente e destinatario sono d'accordo sulla descrizione del ser-

vizio (i.e. condividono lo stesso file WDSL), l'implementazione dietro al Web Service può essere qualunque cosa.

2.3.4 XPath

XPath è un linguaggio per espressioni che permette l'elaborazione di oggetti XML. Tale elaborazione è resa possibile da un modello dei dati che rappresenta documenti XML tramite alberi navigabili. Il risultato di una espressione XPath può essere la selezione di nodi da documenti in input, o valori atomici oppure in generale una qualunque sequenza ammessa nel modello dei dati. Il nome XPath deriva dalla sua caratteristica più distintiva, l'uso di 'cammini' per individuare specifiche informazioni all'interno della struttura ad albero di un determinato oggetto XML. Una tipica espressione XPath per ottenere il nome della persona sul seguente tipo XML è '/persona/nome'.

```
<persona>
 <nome>Mario</nome>
 <cognome>Rossi</cognome>
</persona>
```

2.3.5 Il Protocollo di Comunicazione per i Web Service

Concentriamo ora la nostra attenzione sui passi richiesti per utilizzare un Web Service. Sebbene questi passi siano necessari potrebbero non essere sufficienti: molti scenari richiederanno passi aggiuntivi o raffinamenti significanti di questi passi fondamentali. Inoltre, l'ordine con cui questi passi vengono eseguiti può variare a seconda della situazione.

1. Le entità richiedente e fornitrice si conoscono nel senso che l'iniziatore deve conoscere l'altra parte. Ci sono due casi:

   (a) nel caso tipico, l'agente richiedente sarà l'iniziatore. In questo caso, diremo che l'entità richiedente deve conoscere l'entità fornitrice, cioè l'agente richiedente deve in qualche modo ottenere

l'indirizzo dell'agente fornitore. Questo avviene tipicamente in due modi: l'agente richiedente può ottenere l'indirizzo dell'agente fornitore direttamente dall'entità fornitrice oppure l'entità richiedente può usare un servizio di discovery per localizzare una descrizione del servizio adatta (che contiene l'indirizzo per invocare l'agente fornitore) attraverso una descrizione funzionale oppure attraverso un discovery manuale o una selezione automatica.

(b) in altri casi, l'agente fornitore può iniziare lo scambio di messaggi tra l'agente richiedente e fornitore. In questo caso, dire che le due entità si conoscono in genere vuol dire che l'entità fornitrice conosce l'entità richiedente; in altre parole l'agente fornitore in qualche modo ottiene l'indirizzo dell'agente richiedente. Come ciò possa avvenire dipende dall'applicazione ed è irrilevante per questa architettura. Sebbene questo caso sia meno diffuso, è importante in scenari di tipo abbonamento (il client si abbona ad un servizio e l'erogatore del servizio lo contatta per erogarlo).

2. Le due entità si mettono d'accordo sulla descrizione del servizio (un documento WDSL) e sulla semantica che regolerà l'interazione tra i due rispettivi agenti. Questo non significa necessariamente che le due entità devono comunicare o negoziare l'un l'altra ma semplicemente che devono condividere la stessa descrizione del servizio (o una compatibile) e la relativa semantica e intendono sostenerla. Si può raggiungere questo scopo in vari modi:

   - Le due entità possono comunicare direttamente l'una con l'altra e mettersi esplicitamente d'accordo sulla descrizione del servizio e sulla semantica.
   - L'entità fornitrice può pubblicare e offrire sia la semantica del

servizio che la sua descrizione che il fruitore accetta senza alcuna obiezione.

- La descrizione del servizio e la sua semantica possono essere definiti da una organizzazione industriale e usate da molte entità fornitrici ed erogatrici. In questo caso, il consenso di entrambe le parti è raggiunto indipendentemente grazie all'adesione allo stesso standard.
- La descrizione del servizio e la sua semantica possono essere definiti e pubblicati dall'entità richiedente e offerti dalle entità fornitrici su basi che non vengono discusse. Questo può succedere, ad esempio, se una grande compagnia richiede ai suoi fornitori di fornire Web Service che siano conformi ad una particolare descrizione del servizio e semantica. In questo caso l'accordo è raggiunto tramite l'utilizzo da parte dell'entità fornitrice delle informazioni pubblicate dal richiedente.

A seconda dello scenario il passo 2 può essere eseguito prima del passo 1.

3. La descrizione del servizio e la sua semantica sono noti a entrambi gli agenti. In altre parole, l'informazione di cui dispongono deve essere input di entrambi o comunque fornita nell'implementazione. Ci sono molti modi in cui si può raggiungere questo obiettivo e l'architettura non lo specifica e non ne ha bisogno. Per esempio

   - Un agente potrebbe avere la descrizione del servizio e la relativa semantica direttamente cablati nel codice.
   - Un agente potrebbe essere implementato in una maniera più generica e la descrizione del servizio e la sua semantica possono essere input dinamici.

- Un agente può essere dapprima creato e la descrizione del servizio e la sua semantica possono essere dedotti o generati dal codice dell'agente. Per esempio un tool può esaminare un insieme di informazioni e generare da un insieme di classi esistenti la descrizione del servizio (come il tool Java2WSDL fornito da Apache Axis).

  Qualunque sia l'approccio usato la descrizione del servizio e la sua semantica devono essere noti ad entrambi gli agenti prima che i due agenti interagiscano.

4. I due agenti si scambiano messaggi SOAP per eseguire le comunicazioni.

Capitolo 3

La specifica SPCoop del CNIPA

In questo capitolo viene presentata la specifica della Cooperazione Applicativa del Centro Nazionale per l'informatica nella Pubblica Amministrazione (CNIPA) per il Servizio Pubblico di Cooperazione (SPCoop). Per Cooperazione Applicativa si intende un paradigma finalizzato alla gestione delle informazioni e dei servizi tra Pubbliche Amministrazioni (PA). L'esigenza di una regolamentazione del paradigma di Cooperazione Applicativa è maturata in questi ultimi anni come conseguenza del forte processo d'informatizzazione che ha coinvolto le Pubbliche Amministrazioni, comunemente riferito come e-Government, ed ha infine portato alla pubblicazione della specifica SPCoop da parte del CNIPA. La trattazione si sviluppa in maniera sintetica sui componenti SPCoop in quanto marginali per la comprensione del lavoro di questa tesi. Argomenti centrali come gli Accordi di Servizio e gli Accordi di Cooperazione sono invece trattati con un livello di dettaglio maggiore per la loro centralità nel lavoro oggetto di questa tesi nel sesto capitolo, *Dagli Accordi di Servizio agli Accordi di Cooperazione*.

3.1 La Specifica SPCoop di Cooperazione Applicativa

Il Sistema Pubblico di Cooperazione (SPCoop) è un insieme di standard tecnologici e di servizi infrastrutturali il cui obiettivo è di permettere l'interoperabilità e la cooperazione di sistemi informatici per la realizzazione di adempimenti amministrativi. Tali sistemi sono sotto la responsabilità di soggetti pubblici, appartenenti ad amministrazioni centrali, enti pubblici, regioni, provincie, comuni, comunità di enti locali, e soggetti privati (imprese e associazioni accreditate). L'insieme dei soggetti pubblici e privati operanti sul S.P. (Servizio Pubblico) di Cooperazione costituiscono la comunità dei soggetti del S. P. di Cooperazione. Tale cooperazione è motivata da due esigenze fondamentali:

- L'esigenza di coordinamento di processi realizzati con il concorso di trattamenti distribuiti tra sistemi informatici di cui sono responsabili soggetti pubblici e privati, al fine di assecondare l'esecuzione di procedimenti amministrativi e la produzione di atti e provvedimenti amministrativi. Il coordinamento e la collaborazione di detti sistemi devono essere corredati dalla capacità di ispezionare in ogni momento lo stato di avanzamento (gli adempimenti amministrativi effettuati e quelli ancora da effettuare) dei processi applicativi e l'origine di ogni atto amministrativo effettuato nell'ambito del processo applicativo, al fine di realizzare concretamente la trasparenza dell'azione amministrativa nel doveroso rispetto delle norme sulla confidenzialità e riservatezza dei dati.

- Il coordinamento e la collaborazione dei trattamenti distribuiti tra più sistemi informatici appartenenti a più soggetti pubblici e privati, al fine di assicurare il funzionamento interno delle amministrazioni e di fornire servizi di utilità alle altre amministrazioni, ai cittadini, alle

imprese e alle associazioni, in conformità con i compiti istituzionali delle diverse amministrazioni.

Il modello generale di riferimento dell'architettura del Sistema pubblico di cooperazione è il modello dell'architettura di servizi sulla base di tecnologie Web Service [w3cArch]. I vantaggi principali che derivano dall'adozione di un tale modello per il Sistema Pubblico di Cooperazione sono:

- Il modello dell'architettura dei servizi è pienamente distribuito e permette l'interoperabilità e la cooperazione dei sistemi informatici sulla base di accordi sullo scambio di funzionalità, sulle interfacce che permettono tale scambio e sui suoi requisiti di sicurezza e qualità di servizio, nella piena autonomia delle scelte implementative e gestionali dei sistemi componenti l'architettura. L'indipendenza delle scelte implementative delle funzionalità applicative e delle funzionalità di infrastruttura (gestione dei messaggi, sicurezza, qualità di servizio) 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 delle imprese e delle associazioni qualificate.

- La realizzazione delle funzionalità infrastrutturali di interfaccia, sicurezza e qualità di servizio necessarie alla cooperazione applicativa, sulla base degli standard tecnologici Web Service [w3cArch], permette di minimizzare l'impatto sull'implementazione delle funzionalità applicative già realizzate nei sistemi informativi dei soggetti, pubblici e privati. Il modello non solo garantisce la salvaguardia del patrimonio applicativo dei soggetti pubblici e privati, ma ne stimola la valorizzazione attraverso il riuso in nuovi contesti di funzionalità applicative già implementate. La scelta del modello dell'architettura di servizi sulla base delle tecnologie Web Service è quindi conforme alle esigenze di

economia e di controllo della spesa pubblica e della qualità del servizio amministrativo.

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. Ne consegue che due sistemi interagenti devono essere 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 [SPC]. Il S. P. di Connettività fornisce al S.P. di Cooperazione:

- l'infrastruttura di base per il trasporto dei messaggi,
- un insieme di funzionalità infrastrutturali di sicurezza e qualità di servizio applicabili al trasporto dei messaggi.

Dal punto di vista dei sistemi interagenti nel S. P. di Cooperazione, il S. P. di Connettività è semplicemente e esclusivamente il fornitore della infrastruttura per il trasporto dei messaggi. L'architettura logica e fisica dei sistemi partecipanti al S.P. di Cooperazione è indipendente dell'architettura logica, fisica, di rete del S.P. di Connettività: i soli punti di collegamento si trovano nell'uso del protocollo di trasporto utilizzato dal S.P. di Connettività (IP), e, eventualmente nell'uso di funzionalità di sicurezza (IPSec) e di qualità di servizio applicabili al trasporto fornite dal S.P. di Connettività.

3.2 I Componenti Infrastrutturali previsti in SPCoop

Prima dell'avvento del paradigma di cooperazione applicativa, le comunicazioni nella P.A. avvenivano tramite collegamenti punto-punto tra i server applicativi interessati. Poiché tipicamente questi server erano dislocati

su reti private, raggiungibili quindi esclusivamente da altri server dislocati sulla loro stessa rete privata, era necessario realizzare reti virtuali private (VPN), ad esempio tra tutte le strutture afferenti a una certa amministrazione centrale, o anche appositamente realizzate tra due amministrazione per permettere il collegamento punto-punto tra due server applicativi appartenenti a quelle amministrazioni. Ovviamente questa soluzione si è dimostrata inadatta nella gestione del processo di eGovernment, che prevede che due qualunque enti debbano essere potenzialmente in grado di comunicare tra loro. SPCoop risolve questo problema imponendo un'infrastruttura standard di comunicazione tra le amministrazioni pubbliche. In tal modo, una volta che l'infrastruttura sia diventata completamente operativa, sarà sufficiente il collegamento di un'amministrazione all'infrastruttura SPCoop per abilitarla alla comunicazione con qualunque altra amministrazione italiana ed europea. La figura seguente mostra i principali componenti parte dell'infrastruttura SPCoop, in particolare: la busta eGov, la Porta di Dominio, le porte delegate e applicative e il Registro dei Servizi.

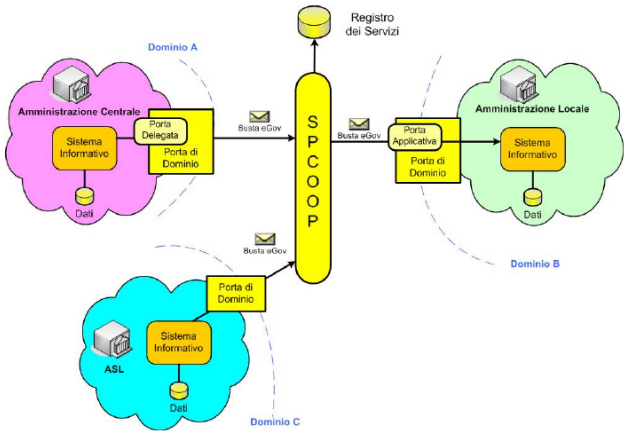

Figura 3.1: I principali componenti dell'infrastruttura SPCoop.

3.2.1 La Porta di Dominio

Nella specifica SPCoop, un dominio è definito come il confine di responsabilità di un ente o soggetto amministrativo e racchiude al suo interno tutte le applicazioni da esso gestite. Le comunicazioni da e verso un dominio devono attraversare la sua Porta di Dominio (PdD). Le Porte di Dominio si parlano tra di loro scambiandosi richieste e risposte in un formato standard, denominato busta eGov. Le *porte delegate* e *le porte applicative* costituiscono gli elementi della Porta di Dominio che mediano gli accessi tra i sistemi interni agli enti e l'infrastruttura SPCoop. In particolare una Porta Delegata è utilizzata come proxy per l'accesso al servizio destinazione, mentre una Porta Applicativa deve essere in grado di gestire la consegna dei contenuti applicativi a un server interno al dominio destinazione. Poiché il formato della busta eGov non è parlato nativamente (e non è previsto che lo sia) dalle applicazioni, la Porta di Dominio deve anche occuparsi di convertire le richieste applicative in formato proprietario nel formato di SPCoop, la busta eGov (vedi 3.2.2). Facendo riferimento a questa problematica, i compiti della Porta di Dominio vengono solitamente divisi in due componenti: il componente di integrazione e quello di cooperazione. Mentre la specifica SPCoop copre completamente gli aspetti relativi al componente di cooperazione, si limita a presentare un esempio di massima di una possibile realizzazione per quanto attiene al componente di integrazione.

3.2.2 La Busta eGov

Lo standard CNIPA sulla Busta di eGovernment è stato il primo ad essere ratificato, il 21 Aprile 2004, tra quelli relativi a SPCoop. La Busta di eGovernment specifica il formato dei messaggi scambiati tra le PdD nelle interazioni di cooperazione applicativa e ne costituisce di fatto l'elemento informativo di base. Una Busta di e-Government è sostanzialmente una

specializzazione di un messaggio SOAP, esteso con un apposito header per definire le caratteristiche del protocollo SPCoop.

3.2.3 Il Registro Servizi

La specifica CNIPA prevede la presenza di un componente architetturale per la memorizzazione della descrizione dei soggetti, dei servizi disponibili e degli accordi di servizio. In particolare la specifica prevede di basare quanto sopra su un registro in tecnologia UDDI e su un repository degli accordi di servizio, tipicamente espressi in XML. La funzione di registrazione permetterà agli enti di registrarsi e di 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. Tramite il registro UDDI è possibile:

- registrare enti o altre istituzioni eroganti servizi in standard SPCoop;
- registrare il servizio offerto da un punto di vista descrittivo;
- 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 (ad es. l'indirizzo del file WSDL).

La specifica SPCoop prevede l'esistenza di un registro di primo livello, gestito dal CNIPA e che includa tutti i servizi ufficiali SPCoop, e di registri di secondo livello che possano contenere un sottoinsieme dei servizi SPCoop.

3.2.4 Il Gestore Eventi

Il Gestore Eventi è un servizio a valore aggiunto, previsto dalla specifica SPCoop per permettere lo scambio di buste eGov secondo l'architettura

EDA (Event Driven Architecture), permettendo quindi ai Sistemi Applicativi iscritti di ricevere le buste inviate dai Sistemi Applicativi che le pubblicano. Da questo punto di vista, il Gestore Eventi può essere considerato un normale servizio SPCoop, accessibile come mostrato nella figura seguente.

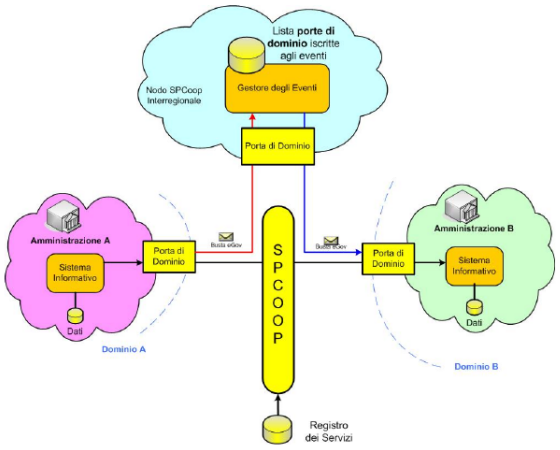

Figura 3.2: Il Gestore Eventi.

3.2.5 I Componenti di Integrazione

La necessità di utilizzare la busta eGovernment per la comunicazione tra i due Server tramite SPCoop pone il problema di come costruire le buste a partire dai dati forniti o attesi dall'applicativo. In sostanza, di come realizzare il componente di integrazione. La soluzione tipicamente adottata è quella mostrata nella figura seguente, che consiste nell'installare una coppia di proxy, sviluppati ad-hoc per ogni servizio, sulle due porte di dominio.

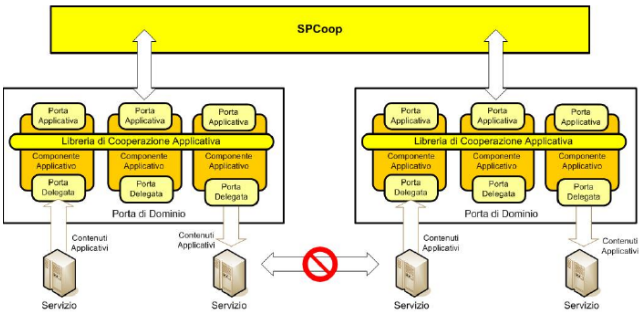

Figura 3.3: Integrazione di SPCoop.

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 eGov. Il proxy ha anche un altro compito, che è quello di conoscere i *metadati* relativi alla comunicazione, come il profilo di collaborazione (sincrono, oneway, asincrono simmetrico, asincrono asimmetrico) attivo per quel servizio e l'indirizzo del destinatario reale del servizio al quale la richiesta dovrà essere diretta (Ente, Servizio e Azione da utilizzare nell'header della busta). Tutti questi dati sono sostanzialmente cablati nello stesso proxy. Questo problema è stato brillantemente risolto nel progetto OpenSPCoop (vedi capitolo 4) potendo adottare la porta di dominio come proxy SOAP. Non è quindi necessario installare un proxy per ogni servizio offerto ma basta configurare opportunamente i servizi.

3.2.6 I Web Service come Soluzione per l'Integrazione

Di recente la grande diffusione dell'XML e dei Web Service, che usano l'XML non solo per il trasporto dei dati (buste SOAP), ma anche per la definizione del loro formato (schemi XML), sta semplificando la realizzazione della componente di Integrazione. Si può infatti considerare un sistema interno già capace o facilmente adattabile al dialogo tramite Web Service 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 (CORBA, RMI, JMS, .NET, etc.), ma potrà essere invece un generico container (ad esempio J2EE) capace di ospitare Web Service che fungano da proxy per le richieste di servizio formulate dai server interni. Una soluzione di questo tipo ha il vantaggio di semplificare la realizzazione dei proxy, ma richiede ancora che i proxy conoscano i metadati relativi alla comunicazione. Anche basandosi sull'uso dei Web Service, sarà quindi ancora necessario realizzare ed installare sulle porte di dominio una coppia di proxy per ogni servizio da raggiungere.

Capitolo 4

OpenSPCoop

OpenSPCoop [OPENSPCOOP] è un progetto finalizzato alla realizzazione di un insieme di componenti Open Source aderenti alle specifiche SPCoop descritte nel capitolo precedente. Il progetto OpenSPCoop nasce sostanzialmente con l'obiettivo di una implementazione di riferimento Open Source che permetta di sperimentare in maniera condivisa l'implementazione dei concetti proposti nella specifica, evidenziando e proponendo possibili soluzioni per le potenziali ambiguità o debolezze della stessa. I vantaggi dell'approccio Open Source adottati da OpenSPCoop si possono evidenziare nei seguenti aspetti:

- Interoperabilità, OpenSPCoop intende rappresentare un riferimento per disambiguare diverse possibili interpretazioni della specifica SPCoop;
- Sicurezza, l'apertura del codice assicura quelle caratteristiche di trasparenza del codice ormai considerate un atto dovuto in molti settori della sicurezza informatica;
- Comunità d'Utenza, OpenSPCoop tende a 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.

Dopo un'ampia fase di analisi, svolta attraverso lo studio di vari progetti pilota tra cui il progetto CART di Regione Toscana e il progetto SOLE di Regione Emilia Romagna, è emersa la proposta di un'architettura innovativa per la realizzazione di un'infrastruttura compatibile con le specifiche CNIPA, che riducesse però significativamente l'impatto sui sistemi preesistenti rispetto alle altre soluzioni disponibili. La prima release del software OpenSPCoop, la 0.1, ha richiesto dapprima una approfondita fase di analisi e di progettazione che ha costituito il lavoro oggetto della tesi di Ruggero Barsacchi [RB]. In una fase successiva, la progettazione è stata estesa ed è stata prodotta la prima versione del codice OpenSPCoop grazie al lavoro di tesi di Andrea Poli [AP]. Questa prima release, rilasciata il 27 ottobre 2005, è stata seguita da frequenti nuovi rilasci, anche grazie al feedback e al contributo dei primi utenti del software. Attorno al sito http://www.openspcoop.org ha cominciato subito a svilupparsi una comunità di utenti e sviluppatori molto qualificati, provenienti da grandi aziende italiane, pubbliche amministrazioni locali e centrali e centri di ricerca. L'interesse diffuso per il progetto ha dimostrato tra l'altro la grande disponibilità di tutti i soggetti interessati verso una soluzione open source in un settore così critico come quello indirizzato dalla specifica SPCoop.

4.1 L'Architettura generale del Progetto

Il progetto OpenSPCoop è strutturato nei seguenti sottoprogetti:

- 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.

La figura successiva mostra l'architettura software del progetto.

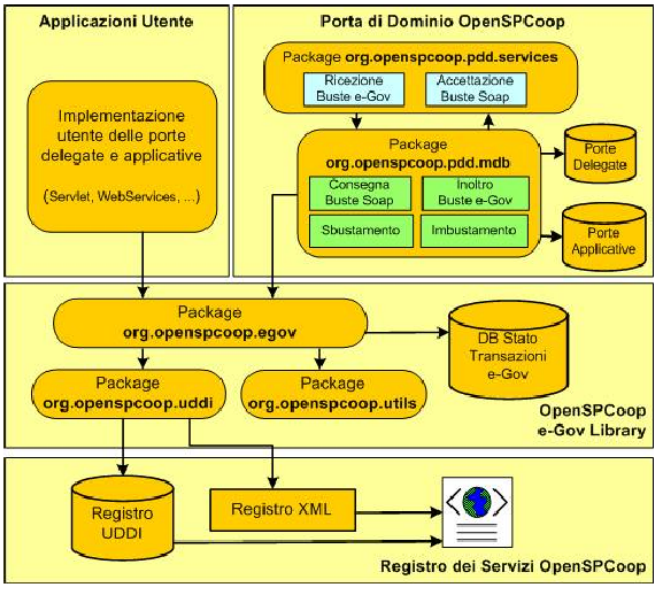

Figura 4.1: Architettura OpenSPCoop

La descrizione dell'architettura di OpenSPCoop viene ora presentata approfondendo i vari componenti fondamentali. La seguente descrizione si presenterà più dettagliata su aspetti rilevanti per questa tesi come l'interazione tra le porte di dominio per l'erogazione/fruizione di un servizio e il Registro Servizi.

4.2 La Porta di Dominio OpenSPCoop

Uno degli aspetti innovativi del progetto OpenSPCoop è la soluzione adottata per la componente di integrazione della Porta di Dominio. L'abilitazione di un nuovo servizio in OpenSPCoop, non richiede infatti la programmazione di componenti applicativi ad-hoc, ma vengono invece messi a disposizione dei servizi generici, utilizzabili da qualunque applicazione. Questi servizi interagiscono con dei repository XML che descrivono le specifiche porte delegate e applicative residenti sulla Porta di Dominio per decidere come gestire le

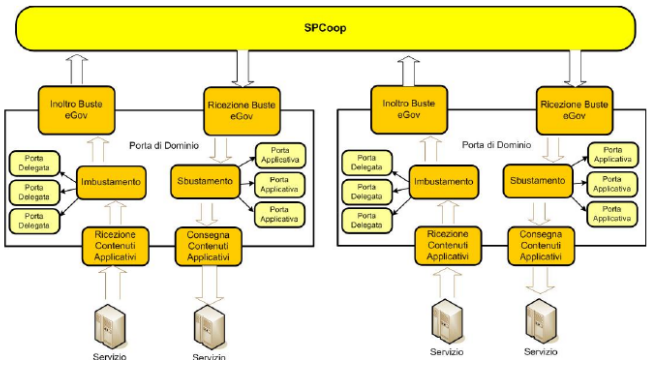

Figura 4.2: La porta di dominio OpenSPCoop

richieste in arrivo. In funzione di queste descrizioni e dello stato delle transazioni eGov in corso, le richieste ricevute dalla Porta di Dominio attraversano quindi una pipeline di moduli specifici, arricchendosi di informazioni utili al loro trattamento mantenute negli header dei messaggi. Infine vengono recapitate al destinatario o girate alla porta di Dominio competente. Questa soluzione, illustrata nella figura 4.2, ha il grande vantaggio di non richiedere l'installazione di un proxy per ogni servizio da utilizzare (porta delegata) o da fornire (porta applicativa), lasciando comunque immutata l'interfaccia d'uso del Servizio, in maniera quindi del tutto trasparente rispetto ai Server Applicativi preesistenti.

OpenSPCoop interviene insomma come un proxy SOAP trasparente, in grado di assicurare tutti gli stessi livelli di servizio di un proxy applicativo, senza però bisogno di realizzarlo come un'applicazione ad hoc. Altro vantaggio dell'approccio seguito in OpenSPCoop deriva dall'evitare che il codice applicativo del proxy venga eseguito direttamente sul server applicativo che funge da porta di dominio, prevenendo così significative controindicazioni dal punto di vista dell'affidabilità e della sicurezza di un componente così critico dell'infrastruttura SPCoop.

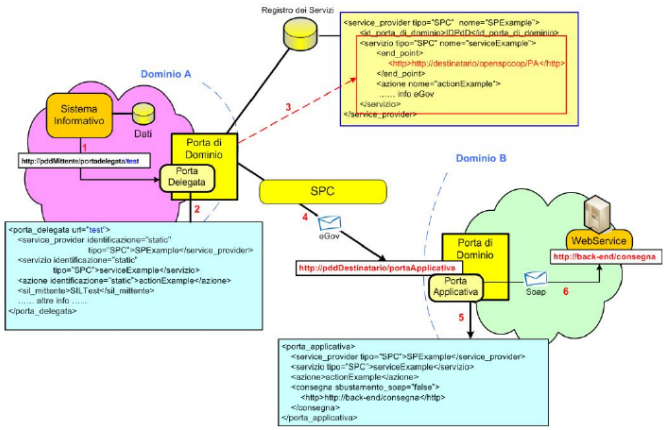

Figura 4.3: OpenSPCoop: esempio d'uso

4.3 Un Esempio d'Uso di Servizi in OpenSPCoop

La figura 4.3 fornisce un esempio completo di una interazione SOA, utilizzando il profilo OneWay della specifica SPCoop.

L'esempio mostra l'invocazione di una porta delegata da parte di un Sistema Informativo Locale (SIL) residente nel dominio mittente, l'interazione delle porte di dominio mittente e destinataria con il registro dei servizi, lo scambio delle buste eGov tra le porte di dominio e infine la consegna del contenuto applicativo al SIL che eroga effettivamente il servizio nel dominio destinatario. In dettaglio, le operazioni eseguite sono le seguenti:

1. il Sistema Informativo interno al dominio di cooperazione mittente (Dominio A in figura), invoca la porta delegata `http://pddMittente/portadelegata/test`.

2. La porta di dominio mittente utilizza la parte variabile del servizio portadelegata (test nell'esempio), per identificare la porta delegata effettivamente invocata nella configurazione locale della porta (file config.xml); dalla descrizione XML della porta delegata, sarà possibile identificata la tripla (fornitore del servizio, servizio, azione);

3. la tripla (fornitore del servizio, servizio, azione) viene quindi utilizzata come chiave d'accesso al Registro dei Servizi, per ottenere l'Accordo di Servizio relativo al servizio destinazione (endpoint, profilo eGov, etc.);

4. sulla base delle informazioni ottenute, la porta di dominio mittente costruisce la busta e la spedisce all'endpoint della porta applicativa indicato nel registro, http://pddDestinatario/portaApplicativa;

5. La porta di dominio destinataria, una volta ricevuta la busta, ne effettua la validazione e, in caso di successo, ne identifica i valori destinatario, servizio e azione; questi valori sono quindi usati per identificare la porta applicativa effettivamente indirizzata nella configurazione locale della porta (file config.xml);

6. a questo punto, utilizzando le informazioni appena reperite relative alle modalità di consegna, la Porta di Dominio destinatario si occuperà di effettuare la consegna dei contenuti applicativi al servizio locale abbinato alla porta applicativa riferita nella busta eGov.

4.4 Il Registro Servizi di OpenSPCoop

OpenSPCoop supporta due tipi di registri, uno realizzato completamente tramite una rappresentazione XML e un altro che prevede l'uso di un registro UDDI che indicizza un insieme di oggetti descritti in XML e accessibili via http. Prima di analizzare la struttura dei due tipi di registro, vediamo quali sono i due oggetti principali indirizzati nel registro: i Soggetti SPCoop e i Servizi. Il Soggetto SPCoop, come illustrano le specifiche SPCoop, viene utilizzato per identificare un'organizzazione/dominio che eroga o fruisce di servizi applicativi. Un soggetto e il proprio dominio vengono identificati da un nome simbolico il più possibile autoesplicativo della missione istituzionale del soggetto stesso. Il dominio di cooperazione dovrà quindi assumere

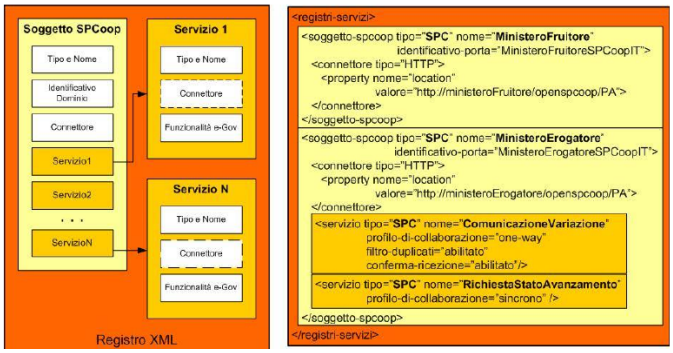

Figura 4.4: OpenSPCoop: Registro Servizi XML

il nome del soggetto con suffisso *SPCoopIT*. Nel registro XML è possibile indicare, oltre al nome del soggetto (inteso come coppia tipo/identificatore, ad esempio SPC/MinisteroInterni), anche l'identificativo del dominio e il punto di accesso della porta di dominio utilizzata dal soggetto stesso. La registrazione di un Servizio, oltre alla definizione di un nome (inteso come coppia tipo/identificatore, ad esempio SPC/ComunicazioneVariazione) e la descrizione dell'Accordo di Servizio, permette la definizione opzionale di un ulteriore punto di accesso per l'erogazione del servizio, se diverso dal punto di accesso della sua porta di dominio.

4.4.1 Il Registro XML

Nella versione XML tutti i soggetti SPCoop, sia erogatori che fruitori di servizio, e le descrizioni dei servizi (gli accordi di servizio, in terminologia SPCoop) sono registrati in un unico file XML che deve essere accessibile alla porta di dominio. La struttura del registro XML di OpenSPCoop è mostrata nella figura 4.4, insieme ad un esempio di file XML che descrive la cooperazione tra il soggetto SPC/MinisteroFruitore e il soggetto SPC/MinisteroErogatore che eroga il servizio one-way SPC/ComunicazioneVariazione e il servizio sincrono SPC/RichiestaStatoAvanzamento.

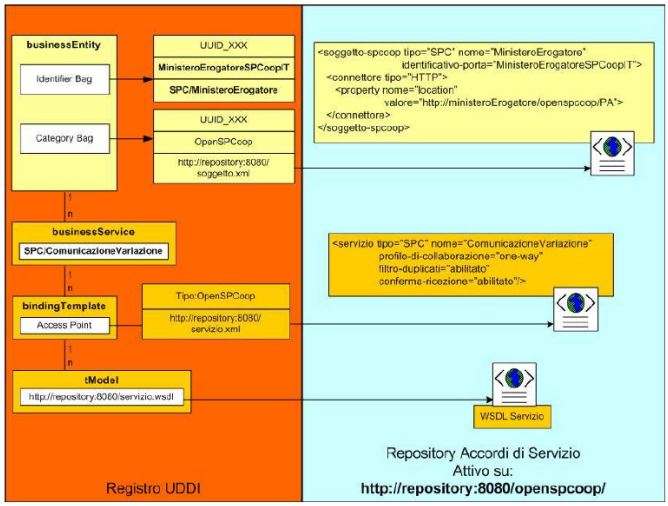

Figura 4.5: OpenSPCoop: Registro Servizi UDDI

4.4.2 Il Registro UDDI

Nella versione UDDI del registro OpenSPCoop esistono invece un insieme di file XML diversi, tanti quanti sono i soggetti in gioco e i servizi erogati, ognuno dei quali descrive uno specifico soggetto o servizio. I file XML, memorizzati in un server Web e reperibili quindi via http, sono riferiti a partire da oggetti registrati in un registro UDDI. Nella figura che segue viene mostrata la struttura del registro UDDI, istanziata su un esempio di definizione del servizio SPC/ComunicazioneVariazione erogato dal soggetto SPC/MinisteroErogatore.

Registrazione Soggetti SPCoop

Per la registrazione di un soggetto nel registro UDDI viene utilizzata una chiave di tassonomia appositamente definita. La registrazione del soggetto comporta la creazione di una *Business Entity* che possiede un Identifier Bag che memorizza una KeyedReference a cui è associata l'identificativo del dominio nel campo KeyName e l'identificatore del soggetto nel campo KeyValue;una Category Bag che memorizza una KeyedReference a cui è

associata nel campo KeyValue la url che indirizza il file XML che descrive il soggetto.

Registrazione Servizi

La registrazione di un servizio richiede la precedente registrazione del Soggetto che lo eroga. Viene creato un Business Service con il campo nome contenente l'identificatore del servizio. Associato al Business Service viene creato un Binding Template contenente nel campo Access Point il riferimento al file XML che descrive le funzionalità SPCoop del servizio. Inoltre, associata al Binding Template esiste una TModel specifica per il servizio nella quale, nel campo OverviewURL, è presente il riferimento al file WSDL che descrive il servizio.

4.4.3 Il Gestore Eventi di OpenSPCoop

OpenSPCoop supporta due tipi di Gestori Eventi, il primo realizzato come servizio SPCoop, mediato quindi da una specifica porta applicativa sulla porta di dominio del dominio che eroga il servizio di Gestore Eventi, il secondo implementato come un broker di messagistica su cui le porte di dominio mittente e destinatario scrivono e leggono usando direttamente jms. Per ulteriori informazioni sul Gestore Eventi si rimanda alla documentazione di OpenSPCoop.

4.4.4 Gli Aspetti di Sicurezza in OpenSPCoop

OpenSPCoop, come richiesto dalla specifiche SPCoop, supporta la gestione della sicurezza sia a livello trasporto (SSL) che a livello messaggio (WS-Security). SSL a livello trasporto può essere configurato sia per gestire i collegamenti dei sistemi informativi locali alla porta di dominio che quelli tra le porte di dominio, usando il supporto SSL degli Application Server che ospitano la porta di dominio. WSSecurity è invece implementato in OpenSPCoop utilizzando la libreria Apache WSS4J (http://wss.apache.org/wss4j/)

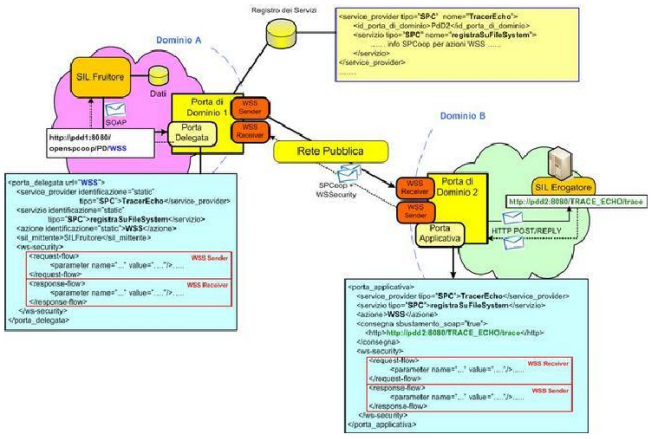

Figura 4.6: OpenSPCoop: WS Security

e sfruttandone i moduli forniti per l'interazione con Axis (WSSDoAllSender e WSSDoAllReceiver). In particolare, per ogni porta delegata e per ogni porta applicativa potranno essere specificate le proprietà di WSSecurity che si vogliono applicare durante la spedizione (WSSDoAllSender) di buste SPCoop, rispettivamente nell'elemento *request flow* di una porta delegata e nell'elemento *response flow* di una porta applicativa, e durante la ricezione (WSSDoAllReceiver) di buste SPCoop, rispettivamente nell'elemento *response flow* di una porta delegata e nell'elemento *request flow* di una porta applicativa. La figura 4.6 mostra un'esempio d'uso di WSSecurity in OpenSPCoop.

Capitolo 5

Linguaggi per Workflow

In questo capitolo verranno illustrati i concetti di processo produttivo automatizzazato o *workflow* e le tecnologie esistenti che ne forniscono l'implementazione. La trattazione presenta dapprima le problematiche collegate ai processi produttivi in merito alla loro descrizione e realizzazione. In seguito sarà presentato il linguaggio WS-BPEL per la sua centrale rilevanza con il lavoro di questa tesi e le argomentazioni che ne hanno motivato la scelta rispetto ai principali concorrenti disponibili.

5.1 Cos'è un Workflow

Con il termine *workflow* si intende un pattern ripetibile di attività in cui è possibile evidenziare una organizzazione sistematica delle risorse, una definizione di ruoli e un flusso di energie e informazioni in un processo produttivo che può essere documentato e compreso. I workflow sono sempre progettati per eseguire vari tipi di manipolazione da quella fisica a quella informatica. I workflow sono strettamente collegati a concetti usati per descrivere la struttura organizzativa come funzioni, team, progetti, gerarchie ecc.. I workflow possono essere visti come uno dei blocchi primitivi di costruzione delle organizzazioni. La connotazione più generica che questo termine ha

assunto è l'identificazione di un tipo di interazione tra l'uomo e le macchine. I workflow sono generalmente caratterizzati da:

- comportamenti dipendenti dai dati. Per esempio un processo di tipo supply-chain dipende da dati tra cui il numero degli oggetti in un dato ordine, il valore totale di un ordine e una data di scadenza per la spedizione. Definire l'intento del processo in questi casi richiede l'uso di costrutti condizionali e di time-out.

- condizioni eccezionali e le loro conseguenze. Il recupero è importante per i processi produttivi almeno quanto l'abilità di definire il comportamento del processo in casi senza fallimenti.

- interazioni a lungo termine che includono lavori multipli e spesso innestati, ognuno dei quali con i propri requisiti sui dati. I processi produttivi frequentemente richiedono coordinazione fra più piattaforme dei prodotti delle unità di lavoro (sia in casi di successo che di fallimento) a vari livelli di granularità.

5.2 I Workflow per Applicazioni di Rete

L'integrazione di sistemi richiede più dell'abilità di interazioni semplici usando protocolli standard. Il pieno potenziale dei Web Service come una piattaforma di integrazione sarà raggiunto solo quando le applicazioni e i processi produttivi sapranno integrare le loro complesse interazioni usando modelli di integrazione di processi standard. Il modello di interazione che è direttamente supportato da WSDL è essenzialmente un modello senza stato di interazione richiesta-risposta e interazioni one-way non correlate.
I modelli per processi produttivi tipicamente assumono sequenze di scambi di messaggi peer-to-peer sia di tipo richiesta-risposta che di tipo one-way, con interazioni con stato e di lunga durata tra due o più partner di comunicazione. Per definire tali processi di interazione bisogna definire una

descrizione formale dei protocolli usati per lo scambio di messaggi usato dal processo produttivo nelle loro interazioni.
Un processo astratto può essere usato per descrivere il comportamento di ognuna delle parti durante lo scambio di messaggi, senza rivelare implementazioni interne.
Ci sono due buone ragioni per separare gli aspetti pubblici del comportamento del processo produttivo dagli aspetti formali o privati. Una ragione è che i processi ovviamente non vogliono rivelare tutte le loro decisioni e la gestione dei propri dati ai partner del processo. L'altra ragione è che separare il processo privato da quello pubblico garantisce la libertà di cambiare aspetti privati dell'implementazione del processo senza modficare il comportamento esterno osservabile e quindi non variando le interfacce. Tale comportamento deve essere chiaramente descritto in una maniera che sia indipendente dalla piattaforma.

5.3 Programmazione Orientata ai Grafi (GOP)

A differenza della programmazione orientata agli oggetti nella quale si pensa in termini di oggetti, campi di oggetti e metodi, nella programmazione orientata ai grafi si ragiona in termini di *attività* ed *eventi* che avviano, interrompono, sincronizzano o terminano le attività. Questo non significa che la programmazione di questi meccanismi non sia realizzabile in un linguaggio object oriented, ma solo che tale realizzazione sia un'operazione più macchinosa. In questi linguaggi, infatti, la strutturazione di più attività che si sincronizzano non è fatta in maniera naturale ma tramite meccanismi realizzati ad hoc: in Java, ad esempio, si ricorre all'utilizzo di *monitor*, cioè oggetti realizzati con meccanismi hardware/software per realizzare la mutua esclusione.
Nella GOP lo sviluppo di applicazioni concorrenti è facile grazie al supporto nativo per i grafi. Il concetto *di esecuzione di un programma in GOP* cor-

risponde alla navigazione di un grafo diretto.

Alcuni aspetti dello sviluppo del software possono beneficiare molto da un approccio basato su grafi. La gestione dei processi produttivi (BPM, Business Process Management) è uno dei domini di applicazione più diffusi di linguaggi basati su grafi.

Com'è facile intuire questo tipo di programmazione può avvalersi di strumenti di editing che ne aumentano il potere espressivo. La rappresentazione grafica è un concetto chiave della GOP. Tramite gli strumenti è infatti possibile realizzare grafi di complessità arbitraria delegando allo strumento di editing stesso la generazione automatica del codice che realizza la logica del grafo. La rappresentazione grafica può essere usata per avvicinare analista di processi produttivi e sviluppatore come fosse il linguaggio naturale.

Uno dei linguaggi esistenti che permette di rappresentare un grafo è il linguaggio XML. Il linguaggio BPEL, ad esempio, è basato su XML.

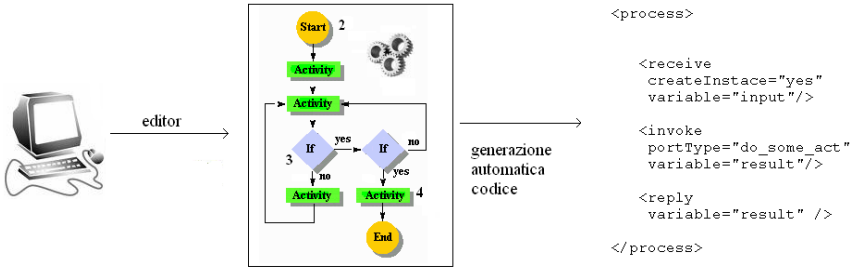

Figura 5.1: Rappresentazione grafica e generazione automatica di codice.

Programmare in GOP può essere molto veloce se opportunamente supportata da strumenti.

Per la natura delle attività che può descrivere, la GOP deve necessariamente fornire un supporto nativo per stati d'attesa per sincronizzare le attività parallele e la gestione di eventi asincroni. Anche questi meccanismi sono realizzabili in un linguaggio object oriented con semafori e monitor, ma è un meccanismo che è più difficile da controllare attraverso il codice.

E' chiaro quindi perchè la programmazione orientata ai grafi costituisca allo stato attuale la base per realizzare linguaggi basati su grafi, come quelli per workflow.

5.3.1 Orchestrazione e Coreografia

Quando la realizzazione di un compito richiede un complesso grafo di attività che richiede la partecipazione di più agenti, identificare un meccanismo di coordinazione diventa fondamentale. In letteratura per esprimere questa coordinazione vengono adottati due approcci:

- *Orchestrazione*: il corretto svolgimento dell'intero compito viene controllato da un unico agente che coordina tutte le attività. Tale agente può anche essere uno dei partecipanti. I partecipanti eseguono compiti in maniera trasparente rispetto alla coordinazione.

- *Coreografia*: il corretto svolgimento del compito è delegato alla trasmissione dello stato globale dell'esecuzione da un agente ad un altro. Nessuno coordina le attività ma ogni agente controlla che lo stato di avanzamento del compito sia al punto giusto quando l'attività dell'agente deve essere eseguita. I partecipanti, a differenza del caso precedente, sono consapevoli della coordinazione.

Tramite la GOP è possibile eseguire solo linguaggi con coordinazione di tipo *orchestrazione*.

5.3.2 GOP a confronto con Reti di Petri

Il mondo accademico, per un lungo periodo, si è concentrato sull'utilizzo di reti di Petri per workflow e modellazione di processi produttivi principalmente perchè le rete di Petri erano l'unico strumento matematico che supportava cammini concorrenti d'esecuzione. Grazie al fondamento matematico è possibile definire algoritmi per la validazione e la completezza. La

maggior differenza tra le reti di petri e la GOP è la loro natura. Le reti di Petri sono un modello matematico mentre la GOP può essere vista come una tecnica di implementazione o un design pattern. La programmazione orientata ai grafi può essere impiegata per implementare le reti di Petri. I places delle reti di Petri e le transizioni possono essere implementati come due differenti tipi di nodi. Il token game di una rete di Petri corrisponde all'esecuzione GOP. Le estensioni al livello più alto che sono state definite sopra le reti di Petri possono anche essere definiti in termini di programmazione orientata ai grafi. La GOP non supporta algoritmi di analisi come sono definiti sulle reti di Petri. Questo perchè la GOP non ha una interpretazione concreta. Gli algoritmi di analisi possono essere definiti su modelli che hanno una interpretazione deterministica a tempo di design. La programmazione orientata ai grafi inoltre prevede la definizione di nodi che hanno un comportamento non deterministico a tempo di design. L'implementazione dei nodi GOP può potenzialmente fare un qualunque tipo di calcolo a tempo d'esecuzione e decidere solo in quel momento come l'esecuzione deve essere propagata. Algoritmi di analisi possono solo essere definiti su linguaggi a processi concreti per i quali l'implementazione dei nodi dà una interpretazione deterministica a tempo di design ai tipi dei nodi.

5.4 BPEL e WS-BPEL

BPEL (Business Process Execution Language) è un linguaggio basato su XML costruito per descrivere formalmente i processi commerciali ed industriali in modo da permettere una suddivisione dei compiti tra attori diversi. Un'applicazione BPEL viene invocata come Web Service ed interagisce con il mondo esterno esclusivamente invocando altri Web Service. In questo senso, essa stessa rappresenta una forma di coordinazione di servizi Web, permettendo altresì di comporre questi ultimi in maniera ricorsiva. L'ambiente runtime all'interno del quale viene eseguito il generico processo è detto motore

(o *engine*) BPEL. Lo standard che definisce l'uso di BPEL nelle interazioni tra Web Service è chiamato BPEL4WS o WS-BPEL. BPEL è nato come integrazione delle ricerche svolte da IBM e Microsoft su WSFL e XLANG, entrambi superati da BPEL. Nell'aprile del 2003 BPEL è stato sottoposto ad OASIS che lo ha standardizzato. Il linguaggio BPEL permette di descrivere un processo business mediante un insieme di attività, che possono essere semplici o strutturate. Le attività semplici esprimono una generica azione (ad es. invoca servizio, ricevi risposta, assegna valore ad una variabile, termina processo, etc...), mentre quelle strutturate sono normalmente utilizzate per raggruppare attività semplici allo scopo di esprimere loop, operazioni condizionali, esecuzione sequenziale, esecuzione concorrente, etc... L'intero processo è descritto mediante un'unica attività strutturata (top-level activity), generalmente di tipo sequenziale. Un tag `scope` racchiude l'insieme di attività che compone una transazione atomica, ovvero un processo che può terminare con un `commit` o un `abort`, senza stati intermedi, nel quale l'arresto del processo su un'attività comporta l'interruzione del processo e la cancellazione delle modifiche in scrittura al database durante le attività precedenti (`undo` delle attività). Questo è necessario ad esempio in una transazione bancaria/finanziaria nella quale ad ogni addebito deve corrispondere un accredito di somme.

Un diagramma di workflow contiene tipicamente operazioni, messaggi, attori (umani o applicativi), applicazioni che definiscono il Web-Service, condizioni logiche (IF), parallelismi, loop e task di sincronizzazione fra operazioni.

BPEL è in particolare adatto a modellare workflow completamente automatizzati, per la composizione di Web Service e l'integrazione di servizi eterogenei per hardware che li esegue, architetture di rete e linguaggio del relativo codice.

BPEL mette altresì a disposizione dei costrutti per esprimere le cosiddette transazioni di lungo periodo (long running transactions, LRT), che rappre-

sentano un'estensione delle transazioni ACID al caso di processi di lunga durata mediante la nozione di compensazione delle attività eseguite. Ancora, il meccanismo della correlazione è utilizzato per tener traccia di una certa conversazione a livello business, identificando così una sorta di sessione tra più partecipanti ad una stessa istanza di processo.

BPEL consente di descrivere un workflow esistente oppure un processo astratto non eseguibile. Con processo astratto si intende un processo specificato solo in parte: non si prevede la sua esecuzione e deve essere dichiarato astratto. Un processo concreto è un processo astratto che contiene i requisiti necessari a renderlo eseguibile. Quindi un processo astratto e uno concreto condividono lo stesso potere espressivo.

I processi astratti forniscono un ruolo descrittivo con più di un solo caso d'uso. Un caso d'uso potrebbe essere la descrizione del comportamento osservabile di alcuni o tutti i servizi offerti da un processo eseguibile. Un altro caso d'uso potrebbe essere definire il template del processo che incorpora le migliori pratiche specifiche del dominio. Tale template catturerebbe la logica essenziale del processo in una maniera compatibile con la rappresentazione a tempo di design, mentre escluderebbe i dettagli di esecuzione da includere quando lo si mappa su un processo eseguibile.

Il linguaggio definisce un formato di esecuzione portabile per i processi produttivi che si basano esclusivamente su risorse di tipo Web Service (WSDL) e dati XML. Tali processi, inoltre, vengono eseguiti e interagiscono con i loro partner in maniera consistente senza tener conto delle piattaforme supportanti o dei modelli di programmazione usati dall'implementazione dell'ambiente ospitante.

La continuità tra processi astratti e eseguibili in WS-BPEL rende possibile esportare e importare aspetti pubblici incorporati nel processo astratto come processo o come template di ruolo mentre si mantiene l'intento e la struttura del comportamento osservabile.

Questo si applica anche quando gli aspetti di implementazione privata usano funzionalità dipendenti dalla piattaforma.
Questa è una caratteristica chiave per l'uso di WS-BPEL dal punto di vista dello sblocco delle potenzialità dei Web Service perchè permette lo sviluppo di tool e di altre tecnologie che aumentano enormemente il livello di automazione e inoltre abbassano il costo di stabilire processi produttivi automatizzati cross enterprise.
WS-BPEL definisce un modello e una grammatica per descrivere il comportamento di processi produttivi basati su interazioni tra il processo e i suoi partner. L'interazione con ogni partner avviene attraverso interfacce Web Service e la struttura delle relazioni al livello di interfacce è incapsulata in quello che è chiamato *partnerLink* (vedi *Breve presentazione del linguaggio BPEL*). Il processo WS-BPEL definisce come più interazioni tra servizi con questi partner sono coordinate per raggiungere l'obiettivo del processo così come lo stato e la logica necessaria per la coordinazione.
WS-BPEL introduce anche un meccanismo sistematico per gestire le eccezioni dei processi e processare gli errori. Inoltre, WS-BPEL introduce un meccanismo per definire come attività individuali o composte all'interno di una unità di lavoro debbano essere compensate quando occorre una eccezione o un partner richiede un riscontro.
WS-BPEL utilizza diverse specifiche XML già viste nel capitolo 2: WSDL 1.1, XML Schema 1.0, XPath 1.0 e XSLT 1.0. I messaggi WSDL e le definizioni di tipo XML forniscono il modello dei dati usato dal processo WS-BPEL. XPath e XSLT forniscono il supporto per la manipolazione dei dati. Tutte le risorse esterne e i partner sono rappresentati come servizi WSDL. WS-BPEL fornisce estensibilità per rendere possibile future versioni di questi standard, specialmente per Xpath e gli standard collegati alle computazioni XML.
Un processo WS-BPEL, quindi, puo essere visto come una definizione riusa-

bile che può essere sviluppata in molti modi e in molti scenari mantenendo sempre un comportamento uniforme a livello applicativo attraverso tutti questi scenari.

5.4.1 Breve Presentazione del Linguaggio BPEL

BPEL è un linguaggio basato su XML per orchestrazione di Web Service. L'intera esecuzione del processo è regolata da azioni di tipo <activity> che vengono intraprese a seconda dell'occorrenza di determinati eventi come la ricezione dei messaggi, la conclusione di operazione sincrona o asincrona e il raggiungimento di rami decisionali. A livello sintattico un processo BPEL nella sua versione per Web Service è identificato come segue:

```
< process name='ncname'
          targetNamespace='uri'
          queryLanguage='anyURI'
          expressionLanguage='anyURI'
          abstractProcess='yes|no'
          xmlns='http://schemas.xmlsoap.org/ws/2003/03/business-process/'>
          <partnerLinks>
           <partnerLink name='ncname' partnerLinkType='qname'
           myRole='ncname' partnerRole='ncname'>+
           </partnerLink>
          </partnerLinks>
          <variables>
           <variable name='ncname' messageType='qname'
           type='qname' element='qname'/>+
           </variables>
          activity
</process>
```

Tra gli attributi di <process> il linguaggio per esprimere le query (default è XPath 1.0) se il processo è astratto, cioè non istanziabile, o concreto e altre informazioni come il namespace che identifica il processo. L'attribu-

to *abstractProcess* denota un processo astratto, come è stato descritto nel paragrafo introduttivo. Tra gli elementi contenuti all'interno di <process> troviamo una lista di `partnerlink`: con questo termine si indica una relazione tra il processo e uno dei servizi nella quale vengono definite le operazioni che il processo ha a disposizione su quel servizio. In termini concreti, la relazione vincola il processo a contattare il servizio coinvolto solo per un gruppo ristretto di operazioni (portType). Nell'implementazione di JBPM-BPEL, questo tipo di variabile mantiene a run-time le informazioni necessarie a contattare il servizio per quell'operazione, quindi l'indirizzo del servizio (SOAP Address) e l'implementazione dell'interfaccia del servizio (Service e Port). Oltre a `partnerlink` troviamo una lista di `variables` : le variabili costituiscono di fatto lo stato del processo e hanno gli usi delle variabili di una classe di un comune linguaggio di programmazione, come mantenere risultati intermedi per operazioni o parti di messaggi ricevuti. I tipi che possono assumere queste variabili vanno da quelli default XML a quelli definiti dai WSDL dei servizi supportati in termini di tipo di messaggi o tipi XML. L'intera esecuzione di un processo BPEL è regolata dalle `<activity>` . Le `<activity>` ammesse in BPEL sono:

- `<receive>` , per eseguire una attesa di messaggio bloccante
- `<reply>` , per rispondere ad una precedente operazione di <receive> e completare una interazione request-response (in genere col client).
- `<invoke>` , per invocare l'operazione di un determinato servizio.
- `<if>` , per rami decisionali di tipo if-then-else. Contiene una <activity> per ramo.
- `<switch>` , per rami decisionali con salti multipli. Contiene una <activity> per salto.
- `<assign>` , per assegnamenti tra variabili.

- <pick> ,per eseguire una attesa di messaggio bloccante con timeout.
- <sequence>, per eseguire una sequenza di attività. Una sua istanza è obbligatoria nella definizione di un processo (simile al concetto di *main*).
- <throw> , per generare una eccezione all'interno del processo.
- <terminate> , per una terminazione non normale del processo.
- <while> , per iterare su una condizione. Contiene una scope <actvity>.
- <wait> , per effettuare una attesa con timeout o con un criterio impostato.
- <flow>, per arbitrare l'accesso ai dati da una attività ad un'altra in esecuzione concorrente
- <scope> , per definire una attività con variabili, partnerlink e <activity> locali.
- <compensate> , per gestire eccezioni all'interno del processo.

Come è facile intuire, le <activity> sono componibili e innestabili e permettono di coprire un'ampia gamma di situazioni tipiche di un processo produttivo. La scelta fatta per realizzare i casi d'uso è ricaduta su BPEL perche ne esiste una implementazione completa su un engine facilmente integrabile con JBoss, l'application server usato da OpenSPCoop (vedi JBPM).

5.5 Altri Linguaggi per Orchestrazione

In questa sezione mostriamo le due maggiori alternative a WS-BPEL. Windows Workflow Foundation non è stato scelto perchè non è open-source mentre YAWL non è stato scelto per mancanza di implementazioni integrabili.

5.5.1 Windows Workflow Foundation

Windows Workflow Foundation è una tecnologia Microsoft per definire, eseguire e gestire processi produttivi. Questa tecnologia fa parte del .NET Framework 3.0 che è disponibile in maniera nativa in Windows Vista e può essere installato su macchine con sistemi operativi Windows XP Service Pack 2 o Windows Server 2003 [WF].

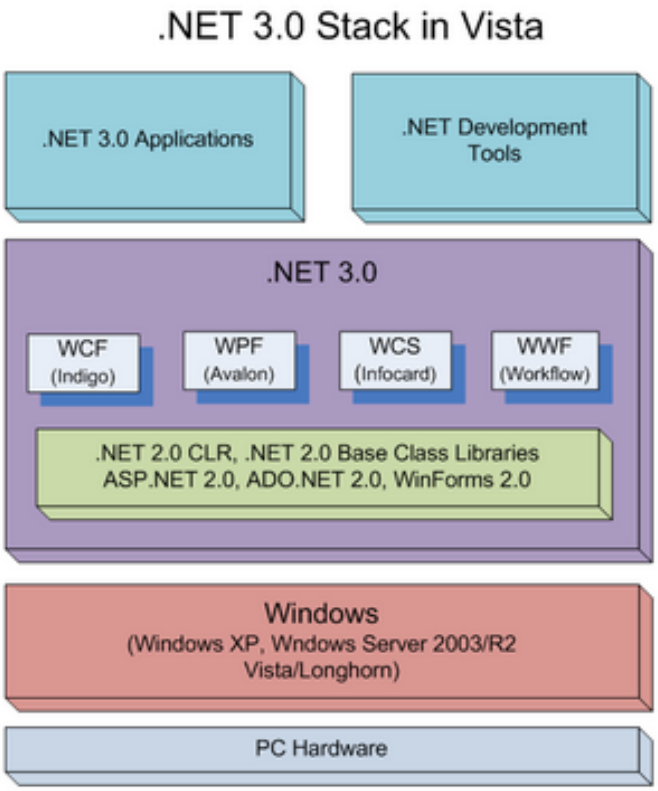

Figura 5.2: .NET 3 Stack

Un nuovo linguaggio basato su XML, XAML, è usato per descrivere la struttura del processo produttivo in Windows Workflow Foundation.
I processi produttivi comprendono le attività. Gli sviluppatori possono scrivere le attività nel loro dominio specifico e utilizzarle nei processi produttivi. Lo stesso Workflow Foundation viene distribuito con un insieme di attività general-purpose che coprono diversi costrutti di controllo di flusso.
Windows Workflow Foundation è supportato grazie ad un insieme di estensioni correlate in Visual Studio 2005. Queste estensioni contengono un design visivo di processi che permette di progettare processi produttivi, un debugger visuale che rende possibile agli utenti il debugging del processo progettato e un sistema a progetto che permette all'utente di sviluppare e compilare il processo all'interno di Visual Studio 2005.
Le attività in genere producono o ricevono dati. In Windows Workflow Foundation queste informazioni vengono esposte usando le proprietà e permettono all'autore di processi di collegarle al processo dichiarando delle dipendenze.
Il *workflow run-time* di .NET 3.0 Framework fornisce i meccanismi necessari per eseguire e gestire workflow e può essere ospitato da un comune dominio per applicazioni CLR, sia esso un Windows Service, una applicazione console, una GUI o una applicazione Web.
L'applicazione ospitante può fornire meccanismi come serializzazione se necessari all'esecuzione del workflow e può inoltre agganciarsi a eventi del workflow di tipo attesa o terminazione.
I workflow in Windows Workflow Foundation definiscono interfacce con metodi ed eventi per comunicare con il mondo esterno. Una applicazione ospitante in genere inizializza un ambiente prima di eseguire un workflow, fornendo oggetti che implementano queste interfacce.
Quando un oggetto che implementa tali interfacce solleva un evento si ottiene il corrispondente workflow per passargli i dati.

I metodi sulle interfacce possono essere usati dal workflow per comunicare con il suo agente ospitante.

5.5.2 YAWL

YAWL (Yet Another Workflow Language) [YAWL] è un linguaggio per workflow basato su pattern workflow. Il linguaggio è supportato da un sistema software che include un engine di esecuzione, un editor grafico e una handler di worklist. Il sistema è disponibile come software open source sotto licenza LGPL. Gli usi a livello di produzione del sistema YAWL includono uno sviluppo nel Regno Unito di processi per automatizzare servizi di front-end e l'utilizzo nella televisione australiana per coordinare i processi di riproduzione delle pellicole. il sistema YAWL è stato anche impiegato per l'insegnamento in oltre 20 università.
YAWL comprende le seguenti caratteristiche:

- Supporto per workflow pattern.
- Supporto per politiche di allocazione di risorse avanzato.
- Supporto per adattamento dinamico di modelli workflow tramite la nozione di worklets.
- Supporto per validazione del modello di workflow (ad esempio riconoscimento deadlock a design-time).
- Modello basato su XML per definizione di dati e manipolazione basata su schemi XML, XPath e XQuery.
- Interfacce basate su XML per monitorare e controllare istanze di workflow e accedere ai registri di esecuzione.
- Interfacce basate su XML per connettere Web Service di terze parti al sistema.

- Generazione automatica di form da schemi XML.

Inizialmente nato ispirandosi alle reti di Petri, YAWL si è evoluto in un sistema con una semantica a transizioni con etichette. Questa scelta implementativa ha reso di fatto possibile l'implementazione di diverse tecniche per analizzare i processi YAWL. In particolare, il sistema YAWL include un tool per analisi statica chiamato WofYAWL.
YAWL è spesso visto come una alternativa a BPEL. Quest'ultimo ha il vantaggio di essere diretto da una commissione di standardizzazione supportata da diverse industrie dell'information technology. Come risultato, BPEL è supportato da un insieme significativo di tool (proprietari e open-source) mentre YAWL ha una singola implementazione per ora. Inoltre, molti ricercatori hanno catturato la semantica formale di un sottoinsieme di BPEL in termini di vari formalismi tra cui le reti di Petri, l'algebra per processi e le macchine a stati finiti. Questo ha preparato la strada per lo sviluppo di tool statici per BPEL che possono competere con le capacità di analisi statica del sistema YAWL. E' stato dimostrato, però, che BPEL non riesce completamente a supportare task umani, cioè svolti da umani e che possono riguardare anche attività fisiche. Una varietà di engine BPEL forniscono estensioni di BPEL per task umani ma queste estensioni non sono ancora standard. YAWL, invece, fornisce una interfaccia unificata per i servizi worklist basati su standard Web Service. Queste interfacce permettono agli sviluppatori di creare i loro servizi worklist per supportare task umani in accordo alle loro necessità. Inoltre, il sistema YAWL viene fornito con un servizio di base worklist che supporta vari tipi di allocazione e gestione di task umani.
Un altro vantaggio di YAWL è il suo supporto per i workflow pattern, sebbene il divario tra YAWL e BPEL possa essere ridotto da nuovi costrutti che sono inclusi in BPEL 2.0.

Capitolo 6

Dagli Accordi di Servizio agli Accordi di Cooperazione

In questo capitolo viene presentato l'Accordo di Cooperazione. La trattazione inizia con la descrizione dell'Accordo di Servizio per poi arrivare all'Accordo di Cooperazione per OpenSPCoop come lavoro di comprensione delle specifiche CNIPA e relativa realizzazione sulla porta di dominio OpenSPCoop.

6.1 L'Accordo di Servizio (AS)

Nell'architettura SPCoop, per instaurare una relazione di servizio tra sistemi è obbligatorio definire un accordo esplicito sulla erogazione/fruizione delle prestazioni del servizio: l'Accordo di Servizio. L'Accordo di Servizio (AS) si basa sulle prestazioni del servizio e sulle modalità di erogazione/fruizione, ovvero più specificamente su:

- le funzionalità (classi di prestazioni) del servizio,
- le interfacce di scambio dei messaggi tra erogatore e fruitore,
- i requisiti di sicurezza dell'erogazione/fruizione,

- i requisiti di qualità di servizio dell'erogazione/fruizione.

I termini dell'Accordo di Servizio sono raccolti in un insieme di documenti, di cui una parte redatta in linguaggi formali e interpretabili da programma. Detto insieme di documenti costituisce una specifica per l'implementazione dei sistemi erogatore e fruitore ed è utilizzato dai progettisti e dagli ambienti di sviluppo nelle fasi di implementazione dei sistemi erogatore e fruitore.

Alcune modalità operative dell'erogazione/fruizione del servizio possono essere lasciate non specificate al momento della progettazione e perfezionate automaticamente all'esecuzione; ciò richiede l'implementazione di capacità specifiche di contrattazione e di riconfigurazione dinamica da parte dei sistemi erogatore e fruitore. Un caso semplice di riconfigurazione dinamica si verifica quando l'URI del punto di accesso del sistema destinatario è sconosciuto al sistema mittente in esecuzione. In questo caso il mittente deve implementare una strategia dinamica di recupero dell'indirizzo (URI) del punto di accesso del destinatario, come per esempio l'interrogazione di un servizio di rubrica. Una modalità alternativa di recupero dell'URI è la trasmissione dello stesso nel contenuto del messaggio. Nella figura 6.1 viene rappresentato lo scenario di utilizzo di un Accordo di Servizio.

La rappresentazione dei termini dell'Accordo di Servizio in un insieme di documenti permette la dissociazione delle responsabilità di definizione del servizio da quelle dell'erogazione del servizio. Un servizio standardizzato:

- è definito da una autorità di definizione, che gestisce l'accordo e la pubblicazione dei documenti che ne rappresentano i termini,

- è erogato da uno o più sistemi erogatori, gestiti da differenti responsabili, che si impegnano a fornire prestazioni conformi con i termini dell'Accordo di Servizio.

L'accordo generale di un servizio standardizzato può essere incompleto: può limitarsi a definire le funzionalità, le interfacce, i requisiti di sicurez-

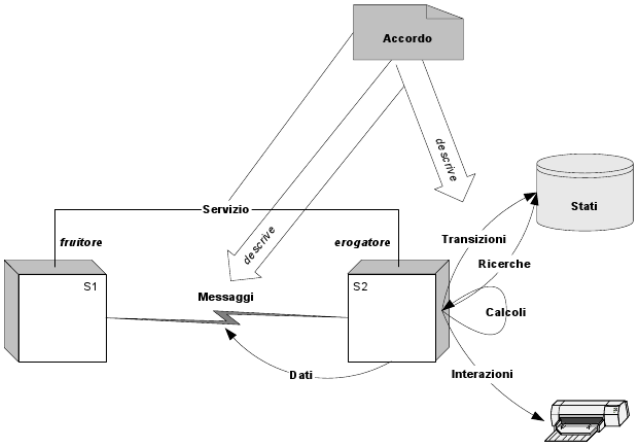

Figura 6.1: L'Accordo di Servizio in SPCoop.

za e lasciare la definizione dei requisiti di qualità di servizio all'autonomia dell'offerta dei singoli erogatori o ad accordi specifici tra singoli fruitori e erogatori.

6.1.1 L'Identificazione delle Parti

L'Accordo di Servizio SPCoop impone l'identificazione delle parti coinvolte nella erogazione/fruizione del servizio.

Le parti in questione ricoprono i seguenti ruoli:

- Erogatori del servizio
- Fruitori del servizio
- Terzi (servizi di intermediazione e di supporto, ad esempio servizi remoti per tracciamento delle operazioni)

I titolari dei sistemi erogatori del servizio e dei servizi terzi devono essere identificati nell'Accordo di Servizio. I titolari dei sistemi fruitori possono essere identificati nell'Accordo di Servizio, e in ogni caso devono essere identificati (e registrati come fruitori) dagli erogatori e terzi.

I codici identificatori dei soggetti titolari sono conformi allo standard SPCoop, sono rilasciati esclusivamente dal Comitato di gestione del S.P. di Cooperazione e sono registrati sul registro SICA nazionale generale, cioè il registro nazionale per la memorizzazione degli accordi di servizio.

6.1.2 La Descrizione delle Funzionalità

L'Accordo di Servizio si basa in primo luogo sulle funzionalità ovvero le classi di prestazioni fornite dall'erogatore del servizio. L'Accordo di Servizio si limita alla descrizione puramente funzionale di tali classi di prestazioni: l'architettura, le tecniche e le risorse d'implementazione adottate dal sistema erogatore per fornire le prestazioni, così come quelle adottate dal sistema fruitore per utilizzare i risultati delle prestazioni, sono totalmente escluse dall'accordo (principio di occultamento dell'implementazione). Lo standard di Accordo di Servizio SPCoop non impone né un linguaggio formale né un formato per la descrizione delle funzionalità del servizio. La descrizione delle interfacce di scambio dei messaggi contiene:

- la descrizione degli eventuali protocolli di conversazione,
- la descrizione del formato del corpo e delle testate dei messaggi,
- la descrizione del collegamento con il protocollo di connessione,
- gli indirizzi (URI) dei punti di accesso,

e, infine, la descrizione delle relazioni funzionali tra i protocolli di conversazione, il contenuto dei messaggi e le modalità di erogazione/fruizione del servizio, cioè la descrizione della semantica (ciò che il mittente vuole significare con l'emissione del messaggio) e della pragmatica (ciò che il destinatario deve fare alla ricezione del messaggio). L'Accordo di Servizio SPCoop V.1.0 non specifica un linguaggio standard per la descrizione dei protocolli di conversazione. Il piano di evoluzione dell'architettura e dell'Accordo di Servizio

SPCoop prevede l'introduzione in versioni successive di SPCoop del formalismo di orchestrazione OASIS WSBPEL [BPEL]. L'Accordo di Servizio SPCoop specifica il formalismo XML Schema [SCHEMA] per la descrizione del formato del contenuto (corpo e testate) dei messaggi. Inoltre prevede delle restrizioni, prescrizioni e raccomandazioni nell'uso dello standard XML Schema e specifica come linguaggio di descrizione dei messaggi, delle connessioni e dei punti di accesso WSDL 1.1 [w3cArch] con le restrizioni d'uso prescritte dalle raccomandazioni WS-I Basic Profile 1.1. La specifica dell'Accordo di Servizio SPCoop fornisce gli elementi standard XML Schema e WSDL necessari alla generazione delle estensioni SOAP proprie alla Busta SPCoop. Per la descrizione della semantica e della pragmatica dei messaggi, è raccomandato l'uso dell'approccio *Design by Contract*. Tale approccio è particolarmente adatto per specificare le richiesta di prestazioni che producono transizioni di stato delle risorse gestite dall'erogatore. L'approccio *Design by Contract* prescrive, per ogni scenario di richiesta/risposta che causa una transizione di stato, la definizione esplicita delle:

- precondizioni: condizioni che devono essere verificate prima della transizione e quindi prima della richiesta,
- postcondizioni: condizioni che devono essere verificate dopo la transizione, e quindi dopo la risposta contenente il resoconto della transizione,
- invarianti: condizioni sempre vere, prima e dopo la transizione.

La specifica dell'Accordo di Servizio SPCoop raccomanda la definizione, per ogni richiesta/risposta che causa una transizione di stato, di richieste informative sulle precondizioni, postcondizioni e invarianti della transizione di stato. Tali richieste possono essere utilizzate in vario modo (per ispezione in esecuzione, per implementare degli scenari di test, ecc.).

6.1.3 Struttura dell'Accordo di Servizio

Un Accordo di Servizio è costituito da una parte comune e da una parte specifica, dove una diversa parte specifica può essere definita per ogni coppia fornitore/erogatore. E' quindi possibile definire servizi singolo/multi fruitore e singolo/multi erogatore. La descrizione di un Accordo comprende i seguenti elementi principali:

- accordo-servizio: corrisponde al contenuto del file manifest della parte comune di un AS;
- servizio: rappresenta la parte specifica di un accordo mono-fruitore. Inoltre contiene, se richieste dal tipo di accordo (multi-fruitore), anche le parti specifiche per ogni singola istanza *fruitore-erogatore*, rappresentata ognuna attraverso l'elemento XML fruitore.
- soggetto-spcoop: rappresenta la registrazione di un soggetto nel registro, che definisce il tipo, nome, dominio di amministrazione e un eventuale connettore, che rappresenta un port di accesso generico per i vari servizi erogati dal soggetto (poi ridefinibile nei port della parte specifica di ogni servizio).

6.1.4 Descrizione della Parte Comune

Il manifesto della parte comune di un Accordo di Servizio, mostrato nella figura 6.2, è definito dall'elemento XML 'accordo-servizio' che racchiude il nome dell'accordo, una sua descrizione non formale, il nome del soggetto referente per l'accordo e riferimenti ai documenti che compongono la parte comune quali:

- specifica delle Interfacce, contenente i documenti WSDL concettuale (descrive le interfacce del servizio a livello di scenario di coordinamento), WSDL logico fruitore (descrive le interfacce e i tipi di dato neces-

sari per il fruitore) e WSDL logico erogatore (descrive le interfacce e i tipi di dato necessari per l'erogatore);

- specifica delle Conversazioni che contiene un documento in formato WSBL concettuale, un WSBL logico fruitore ed un WSBL logico erogatore, con la stessa accezione del punto precedente;
- il riferimento a schemi ed ontologie;
- documento semiformale che segue uno schema XSD, per la definizione di informazioni eGov che compongono il servizio (es. Profilo di Collaborazione).

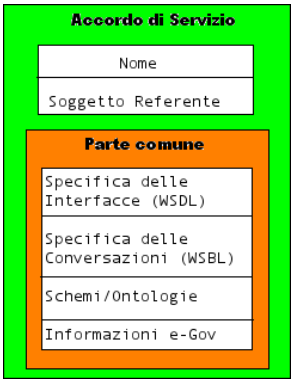

Figura 6.2: Accordo di Servizio: Parte Comune

Un esempio di manifesto XML per la parte comune di un Accordo di Servizio è mostrato in figura 6.3.

6.1.5 Descrizione degli Aspetti relativi al Protocollo SPCoop

La Parte Comune di un Accordo di Servizio deve contenere anche una descrizione (formale o informale) degli aspetti relativi al protocollo SPCoop. In particolare dovrà contenere una descrizione di:

- Servizio

```
<accordo-servizio nome="ComunicazioneVariazioneAS" descrizione="Esempio di Accordo">
  <soggetto-referente tipo="SPC" nome="MinisteroInterni" />
  <specifica-interfaccia  wsdl-definitorio="ComunicazioneVariazioneAS/definitorio.wsdl"
                          wsdl-concettuale="ComunicazioneVariazioneAS/concettuale.wsdl"
                          wsdl-logico-erogatore="ComunicazioneVariazioneAS/logicoErogatore.wsdl"
                          wsdl-logico-fruitore="ComunicazioneVariazioneAS/logicoFruitore.wsdl" />
  <specifica-conversazioni  wsbl-concettuale="ComunicazioneVariazioneAS/concettuale.wsbl"
                            wsbl-logico-erogatore="ComunicazioneVariazioneAS/logicoErogatore.wsbl"
                            wsbl-logico-fruitore="ComunicazioneVariazioneAS/logicoFruitore.wsbl" />
  <catalogo-schemi-ontologie  riferimento="ComunicazioneVariazioneAS/indice" />
  <informazioni-egov  riferimento="ComunicazioneVariazioneAS/informazioniEGov.xml" />
</accordo-servizio>
```

Figura 6.3: Esempio di Manifesto della parte comune di un AS

- Azioni presenti
- Profilo di Collaborazione del servizio (o di una specifica azione del servizio)
- Requisiti di funzionalità eGov come consegnaAffidabile e filtroDuplicati (elemento ProfiloTrasmissione della busta eGov), consegna in ordine e conversazione (elemento Collaborazione e Sequenza della busta eGov)
- Scadenza temporale di una busta

Le funzionalità eGov richieste per un servizio possono essere definite una volta per tutte per qualsiasi azione del servizio. Dopodichè possono essere specializzate per particolari azioni che richiedono funzionalità diverse. Un esempio di descrizione XML delle informazioni eGov di un servizio è il seguente:

```
<informazioni-egov   profilo-di-collaborazione="sincrono"
                     conferma-ricezione="disabilitato"   filtro-duplicati="disabilitato"
                     id-collaborazione="disabilitato"    consegna-in-ordine="disabilitato"
                     scadenza="5g" >
    <azione nome="azione1"  />
    <azione nome="azione2"  />
    <azione nome="azione3" profilo-di-collaborazione="oneway"
                           conferma-ricezione="abilitato"   filtro-duplicati="abilitato" />
    <azione nome="azione4" profilo-di-collaborazione="asincronoSimmetrico"
                           id-collaborazione="abilitato" />
     ....
</informazioni-egov>
```

Figura 6.4: Documento semiformale con informazioni e-gov

6.1.6 Descrizione della Parte Specifica

Il manifesto della parte specifica di un'Accordo di Servizio è definito dall'elemento XML 'servizio'. Il Manifesto racchiude i riferimenti a documenti che definiscono la parte specifica di una precisa erogazione del servizio. All'interno del manifesto possono essere presenti la definizione di una singola parte specifica (per servizio mono-fruitore) o la definizione di diverse parti specifiche, una per ogni fruitore del servizio (servizio multi-fruitore). La parte specifica può essere composta da: un unico file zip che racchiude il manifesto e i vari documenti per un Accordo di Servizio [monofruitore-monoerogatore] o [multifruitore-monoerogatore]; diversi file zip che racchiudono ognuno il manifesto e i vari documenti di una parte specifica di ogni servizio erogato, nel cao di un contesto [monofruitore-multierogatore] o [multifruitore-multierogatore]. Un manifesto di una parte specifica comprende il tipo e il nome del servizio erogato, il riferimento alla parte comune che viene estesa e la definizione della parte specifica. La parte specifica può essere fornita in diverse copie nel caso di istanziazione di servizio multi-fruitore, dove ogni singola istanza [fruitore,erogatore] avrà la propria parte specifica. La definizione di una parte specifica comprende riferimenti ai documenti quali:

- Porti di Accesso del servizio.
- Specifica delle Interfacce, composta da documenti in formato WSDL implementativo fruitore e WSDL implementativo erogatore.
- Definizione di livelli di servizio (SLA).
- Politiche di Sicurezza del Servizio.
- Documento semiformale per la ri-definizione di informazioni relative al protocollo SPCoop richieste dall'Accordo per la specifica istanza del servizio.

Il manifesto di un servizio sarà associato logicamente ad un soggetto spcoop, erogatore di tale servizio. Lo schema logico è il seguente:

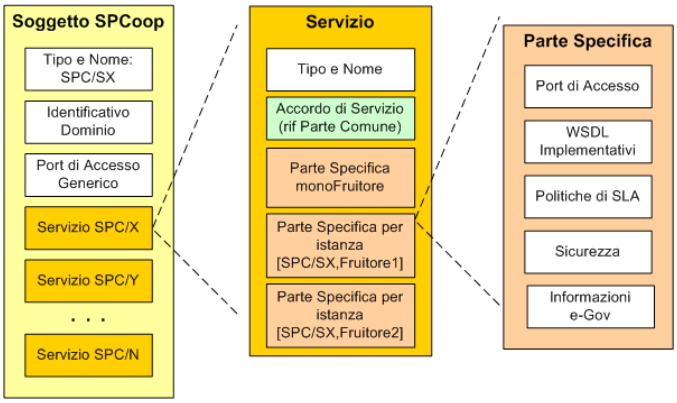

Figura 6.5: Accordo di Servizio:Parte Specifica

Un esempio di manifesto XML è il seguente:

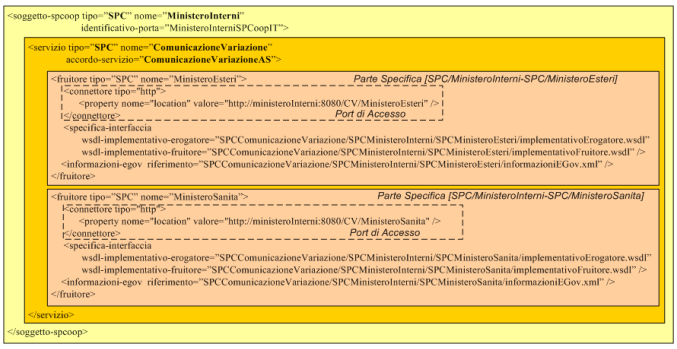

Figura 6.6: Esempio di Manifesto della parte specifica di un AS

6.2 L'Accordo di Cooperazione

Da un punto di vista astratto, un Accordo di Servizio descrive una comunicazione/collaborazione tra due soggetti, in cui uno offre un servizio applicativo SPCoop ed l'altro fruisce di tale servizio. Pertanto, tutti i servizi applicativi offerti da un Dominio rientrano nella responsabilità del singolo soggetto. In

realtà molti procedimenti e compiti istituzionali non riguardano l'operato di una singola amministrazione, ma vedono altresì il concorso dell'azione di più soggetti. Tale situazione, che in base ai processi organizzativi di decentramento e di delega dallo Stato centrale verso le Regioni e gli Enti locali è sempre più frequente, si esemplifica in due principali tipologie di interazione:

- procedimenti inter-amministrativi, nei quali più amministrazioni concorrono, con compiti diversi, al conseguimento di un risultato complessivo, riconducibile ad uno o più servizi integrati erogati sia a fruitori esterni alla PA (ad esempio Sportello unico alle imprese, Sportello unico per l'immigrazione, ecc.) che a fruitori interni alla PA. Questo tipo di procedimenti sono incentrati sulla amministrazione che eroga il servizio integrato finale;

- procedimenti di razionalizzazione, coordinamento e controllo, in cui è individuato normativamente un soggetto vigilante e/o di regolazione, a livello centrale o territoriale (ad es., Regioni, Provincie, ecc.), mentre le funzioni amministrative sono attribuite a soggetti periferici, tipicamente enti locali (ad esempio Anagrafi, Demanio ecc.) che erogano sul territorio una stessa gamma di servizi.

Il Dominio di Cooperazione rappresenta la formalizzazione della volontà di associazione tra diversi soggetti per cooperare nella informatizzazione di un insieme di procedimenti amministrativi pertinenti. Nel Dominio di Cooperazione deve essere individuato un soggetto coordinatore responsabile, che assicura l'efficacia organizzativa e tecnica della cooperazione ed il coordinamento degli adempimenti di ciascuno dei soggetti partecipanti, e l'insieme dei servizi applicativi composti che il Dominio di Cooperazione eroga verso l'esterno. Infatti esternamente il Dominio di Cooperazione è un erogatore di servizi al pari dei Domini di responsabilità delle singole amministrazioni; la differenza è nella realizzazione di tali servizi, che nel caso

del Dominio di Cooperazione nascono dall'integrazione e composizione dei servizi offerti dai singoli domini regolata sulla base di accordi specifici tra le parti in causa, mentre nel caso del singolo dominio la realizzazione si appoggia ad applicazioni completamente sotto la responsabilità della singola amministrazione.

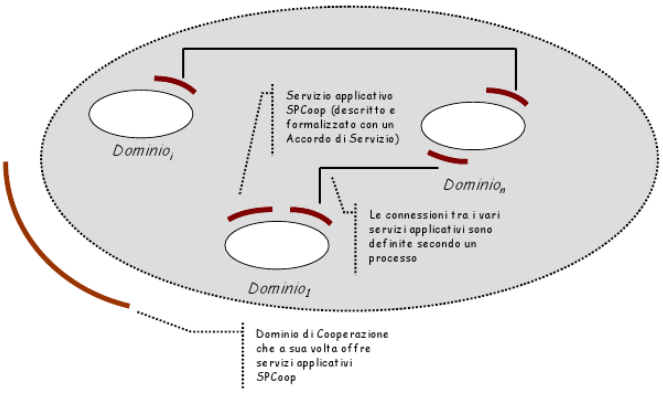

Figura 6.7: Dominio di Cooperazione in SPCoop

Il Dominio di Cooperazione deriva spesso dalla trasposizione di una norma di legge, dove sono individuati i soggetti coinvolti e quello che ha responsabilità del coordinamento o della vigilanza. Ove i ruoli non siano esplicitamente previsti dalla normativa, vale il principio generale di pariteticità delle amministrazioni che, in tal caso, concordano quale tra di loro è responsabile del Dominio di Cooperazione. Il principio di titolarietà (e quindi di responsabilità) dell'azione amministrativa sul singolo adempimento/procedimento è comunque valido. Ne consegue che nel Dominio di Cooperazione ciascun adempimento o parte di procedimento è associato al soggetto pubblico che istituzionalmente ne ha la responsabilità. Lo stesso rimane responsabile dei dati e dei servizi scambiati di cui è normativamente titolare. Un Accordo di Cooperazione (ACoop) è in sintesi la specifica dei servizi applicativi offerti da un Dominio di Cooperazione. Tre elementi fondamentali caratterizzano l'erogazione di servizi applicativi da parte di un Dominio di Cooperazione e

sono:

- i servizi applicativi che il Dominio di Cooperazione offre all'esterno. Dal punto di vista del fruitore, questi sono indistinguibili da qualsiasi altro servizio applicativo offerto direttamente da un Dominio, e vengono descritti attraverso un Accordo di Servizio; nel seguito verranno indicati come servizi composti;
- i servizi applicativi che il Dominio di Cooperazione utilizza internamente, componendoli, per offrire i servizi composti; anche questi sono descritti attraverso il proprio Accordo di Servizio; nel seguito verranno indicati come servizi componenti;
- per ognuno dei servizi applicativi composti, la specifica della modalità secondo cui i servizi componenti sono coordinati al fine di offrire il servizio composto. Tale specifica può essere definita secondo due diverse ottiche:
  - dal punto di vista interno al servizio composto, ovvero descrivendo il processo secondo cui i servizi componenti devono essere coordinati per offrire il servizio composto. In tal caso si parla di orchestrazione;
  - dal punto di vista esterno, ovvero descrivendo i vincoli sugli scambi di messaggi tra i vari servizi componenti; in tal caso si ha cioè la specifica di una collaborazione N-party. Si parla allora di coreografia.

Attualmente esiste uno standard maturo per la descrizione di un'orchestrazione, WS-BPEL (Web Service Business Process Execution Language [BPEL], che abbiamo visto nel capitolo 5), e sono in corso delle iniziative di standardizzazione della coreografia come ad esempio WS-CDL (Web Service Choreography Description Language). Pertanto un Accordo di Cooperazione si compone di:

- un documento istitutivo in linguaggio naturale che descrive le finalità ed il fondamento normativo o istituzionale del Dominio di Cooperazione;
- un insieme di riferimenti ordinati agli Accordi di Servizio che descrivono i servizi composti offerti dal Dominio di Cooperazione;
- un insieme di documenti WS-BPEL che descrivono il processo di coordinamento tra i servizi composti; tali documenti possono anche servire per l'esecuzione diretta, attraverso opportune tecnologie di orchestrazione che interpretano dinamicamente i documenti WS-BPEL, del servizio composto da parte dell'organizzazione responsabile;
- un insieme di liste di riferimenti agli Accordi di Servizio che descrivono i servizi componenti.

In termini formali si ha:

ACoop= <
Documento,
{ AS 1 , . . . , AS i , . . . , AS n },
{ O 1 , . . . , O i , . . . , O n },
{ AS 1 1 , . . . , AS j 1 , . . . AS m 1 ,}
{ . . . }
{ AS 1 i , . . . , AS j i , . . . AS m i ,}
{ . . . }
{ AS 1 n , . . . , AS j n , . . . AS m n }
>

da cui emerge che il servizio composto descritto dall'Accordo di Servizio ASi , si compone a partire dall'insieme dei servizi descritti da altrettanti Accordi di Servizio {AS 1 i , . . . , AS m i }, secondo il processo descritto dall'orchestrazione Oi .

6.3 L'Accordo di Cooperazione nel Registro Servizi OpenSPCoop

Rispetto ad un Accordo di Servizio, un Accordo di Cooperazione definisce le relazioni di servizio in qualità di intermediario: la relazione che questo tipo di accordo è volta a creare coinvolge infatti più organizzazioni. Tra queste possiamo identificarne di tre tipi:

- Fruitrice, cioè l'organizzazione che intende ricevere la prestazione del servizio.
- Erogatrici, cioè le organizzazioni che garantiscono le prestazioni di servizio necessarie alla realizzazione dell'Accordo di Cooperazione.
- Referente, cioè l'organizzazione che si rende responsabile del corretto svolgimento dell'intera operazione concordata con l'organizzazione fruitrice.

E' chiaro quindi come l'organizzazione referente abbia una definizione duale: essa è infatti erogatrice nei confronti dell'organizzazione fruitrice e al tempo stesso fruitrice dei servizi erogati dalle organizzazioni erogatrici.

L'Accordo di Cooperazione definisce quindi una composizione tra servizi. Possiamo ora identificare due tipi di servizi:

- servizi componenti
- servizi composti.

I servizi componenti includono un riferimento all'Accordo di Servizio a cui appartengono. La loro implementazione è la stessa della versione precedente di OpenSPCoop. I servizi composti sono composizione di servizi componenti e sono i diretti responsabili della cooperazione tra i servizi componenti di cui si compongono.

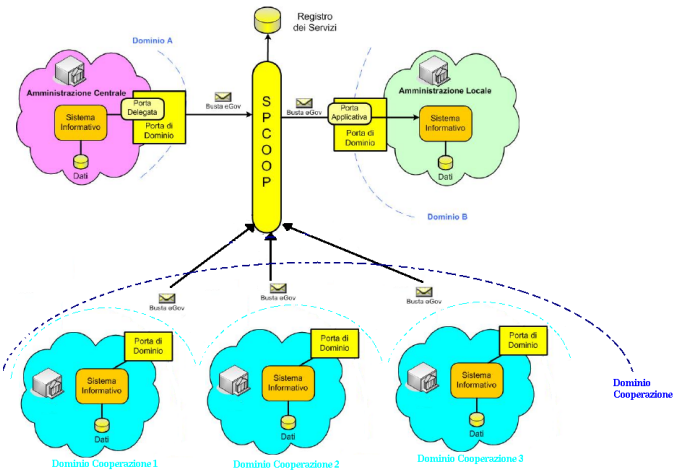

Figura 6.8: Dominio di Cooperazione

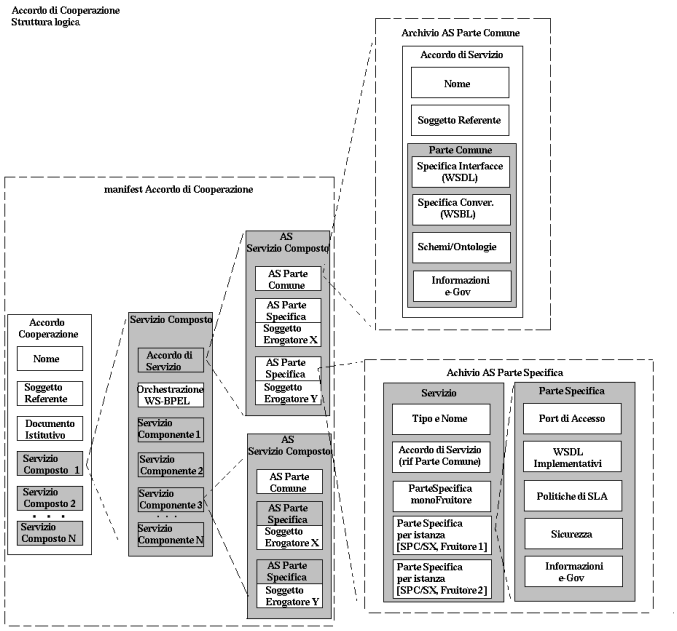

Figura 6.9: Visione Logica dell'Accordo di Cooperazione

Il Registro Servizi di OpenSPCoop è stato quindi arricchito di un nuovo tipo che rappresenti il concetto di servizio composto (*servizio-composto*) e

il suo relativo utilizzo come servizio base per la cooperazione in un Accordo di Cooperazione (*accordo-cooperazione*).

6.3.1 Accordo di Cooperazione nel Registro Servizi

L'Accordo di Cooperazione permette la definizione di un nome, la descrizione dell'Accordo di Servizio e la definizione delle URL ai WSDL che descrivono l'interfaccia del servizio.

```
<xsd:element name='accordo-cooperazione'>
 <xsd:complexType>
  <xsd :sequence>
       <xsd:element name='soggetto-referente' maxOccurs='1' minOccurs='1'>
       <xsd:complexType>
       <xsd:attribute name='tipo' type='xsd:string' use='required'/>
       <xsd:attribute name='nome' type='xsd:string' use='required'/>
       </xsd:complexType>
       </xsd:element>
       <xsd:element name='servizio-composto'
       maxOccurs='unbounded' minOccurs='1'>
       <xsd:complexType>
       <xsd:attribute name='nome-accordo-servizio'
       type='xsd:string' use='required'/>
       </xsd:complexType>
       </xsd:element>
  </xsd:sequence>
  <xsd:attribute name='nome' type='xsd:string' use='required'/>
  <xsd:attribute name='descrizione' type='xsd:string'/>
  <xsd:attribute name='documenti-istitutivo' type='xsd:string'/>
 </xsd:complexType>
</xsd:element>
```

- Attributi:

  - *nome* rappresenta il nome associato all'Accordo di Cooperazione
  - *descrizione* informazione puramente descrittiva contenente una descrizione funzionale associata all'Accordo di Cooperazione
  - *documenti-istitutivo* riferimento ad eventuali documenti relativi a questo Accordo di Cooperazione

- Nested Element

  - *soggetto-referente*. Questo nuovo tipo, specifico per l'Accordo di Cooperazione, e' ricavato rielaborando il soggetto-spcoop gia' presente nel Registro Servizi di OpenSPCoop e in quanto tale necessita delle informazioni per poter essere univocamente identificato (attributi *tipo* e *nome*). Il soggetto referente identifica sul Registro Servizi l'organizzazione referente, cioe' quella che garantisce il corretto svolgimento delle operazioni.
  - *servizio-composto* identifica un servizio composto (specificato in dettaglio più avanti). L'attributo *nome-accordo-servizio* identifica l'Accordo di Servizio per l'erogazione/fruizione di questo Accordo di Cooperazione. E' necessario almeno un servizio composto per qualunque Accordo di Cooperazione.

L'Accordo di Cooperazione, come appena visto, prevede una sequenza di servizi-composti. Questo è un nuovo elemento del Registro Servizi che è riproposto di seguito.

```
<!- Servizio Composto ->
<xsd:element name='servizio-composto' maxOccurs='1' minOccurs='0'>
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element name='servizio-componente' maxOccurs='unbounded' minOccurs='2'>
   <xsd:complexType>
   <xsd:attribute name='tipo-soggetto' type='xsd:string' use='required'/>
```

```
    <xsd:attribute name='nome-soggetto' type='xsd:string' use='required'/>
    <xsd:attribute name='tipo' type='xsd:string' use='required'/>
    <xsd:attribute name='nome' type='xsd:string' use='required'/>
    <xsd:attribute name='azione' type='xsd:string'/>
   </xsd:complexType>
   </xsd:element>
   </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Per il tipo servizio-composto sottolineiamo la definizione di *servizio-componente*. Questo nuovo tipo rappresenta l'adattamento del servizio-spcoop già presente nel Registro Servizi di OpenSPCoop, cioè il soggetto responsabile del servizio e il servizio stesso (attributi *tipo* e *nome*). Si noti anche il vincolo sul numero minimo dei servizi componenti: per parlare di servizio composto ce ne vogliono almeno due.

6.3.2 Esempio di Configurazione del Registro Servizi con un Accordo di Cooperazione

Vediamo ora un caso di configurazione pratica reale del Registro dei Servizi in OpenSPCoop per un Accordo di Cooperazione. Questo accordo prevede il coinvolgimento di quattro soggetti:

- MinisteroFruitore, cioè il soggetto dell'organizzazione che usufruisce del servizio composto
- MinisteroErogatore, cioè il soggetto dell'organizzazione che eroga il servizio composto
- Ministero1 e Ministero2, cioè i soggetti delle organizzazioni che partecipano all'Accordo di Cooperazione per erogare i servizi componenti

Questa divisione rispecchia la nozione astratta di Accordo di Cooperazione come è stata definita nei paragrafi precedenti. La presenza di un Accordo

di Servizio è fondamentale anche se si tratta di un Accordo di Cooperazione: l'Accordo di Servizio infatti regola la prestazione tra l'ente fruitore e l'ente erogatore.

Il codice per il Registro in versione XML è presentato di seguito.

```
<!- Accordi di Cooperazione ->
<accordo-cooperazione
 nome='ServiziUtenti' descrizione='Esempio di un Accordo di Cooperazione' >
 <soggetto-referente tipo='SPC' nome='MinisteroErogatore' />
 <servizio-composto nome-accordo-servizio='ASVerificaUtente' />
</accordo-cooperazione>
```

Per configurare un Accordo di Cooperazione è necessario fornire il nome del soggetto referente e il servizio composto: in sintesi cosa fa l'organizzazione responsabile.

```
<!- Accordo di Servizio del servizio composto ->
<accordo-servizio nome='ASVerificaUtente'
 profilo-collaborazione='sincrono' utilizzo-senza-azione='true' >
 <servizio-composto ws-bpel='http://...PathDelWSPBEL'>
  <servizio-componente accordo-servizio='ASAutenticazioneUtente' />
  <servizio-componente accordo-servizio='ASAutorizzazioneUtente' />
 </servizio-composto>
</accordo-servizio>
```

La configurazione continua con un Accordo di Servizio che serve a indentificare la relazione uno a uno tra erogatore del servizio composto e fruitore dello stesso. Nel servizio composto identifichiamo il riferimento al documento WS-BPEL che contiene la logica di orchestrazione dei servizi composti.

```
<!- Accordo di Servizio del servizio componente 1 ->
<accordo-servizio nome='ASAutenticazioneUtente'
```

```
        profilo-collaborazione='sincrono' utilizzo-senza-azione='true' />
<!- Accordo di Servizio del servizio componente 2 ->
<accordo-servizio nome='ASAutorizzazioneUtente'
        profilo-collaborazione='sincrono' utilizzo-senza-azione='true' />
```

Gli Accordi di Servizio rimangono fondamentali per la cooperazione: essi infatti stabiliscono la singole relazioni necessarie per la realizzazione delle operazioni di invocazione previste dal servizio composto.

```
<!-
MinisteroFruitore:  fruitore del servizio composto
->
<soggetto-spcoop tipo='SPC' nome='MinisteroFruitore'>
         <connettore tipo='http' nome='PdDMinisteroFruitore'>
          <property nome='location'
                    valore='http://localhost:8080/openspcoop/PA' />
         </connettore>
</soggetto-spcoop>
```

Per configurare il soggetto che fruisce del servizio composto è necessario configurare l'indirizzo della porta applicativa utilizzata dal fruitore per le comunicazioni con le altre porte di dominio (*connettore*).

```
<!-
MinisteroErogatore:
- erogatore del servizio composto
- fruitore dei servizi componenti
->
<soggetto-spcoop tipo='SPC' nome='MinisteroErogatore'>
 <connettore tipo='http' nome='PdDMinisteroErogatore'>
  <property nome='location' valore='http://localhost:8080/openspcoop/PA' />
 </connettore>
 <!- Servizio composto ->
```

```
 <servizio tipo='SPC' nome='VerificaUtente' accordo-servizio='ASVerificaUtente'
  <servizio-composto>
   <servizio-componente
    tipo-soggetto='SPC' nome-soggetto='Ministero1'
    tipo='SPC' nome='AutenticazioneUtente' />
   <servizio-componente
    tipo-soggetto='SPC' nome-soggetto='Ministero2'
    tipo='SPC' nome='AutorizzazioneUtente' />
  </servizio-composto>
  <fruitore tipo='SPC' nome='MinisteroFruitore' />
 </servizio>
</soggetto-spcoop>
```

Per configurare il soggetto che eroga il servizio composto è necessario configurare l'indirizzo della porta applicativa del soggetto che eroga il servizio (*connettore*). Oltre a questa operazione bisogna fornire i dettagli relativi ad identificare ogni servizio componente che verrà utilizzato da questo servizio composto nel Registro Servizi. Si identifica l'organizzazione (*tipo-soggetto* e *(*nome-soggetto)) e il servizio richiesto (*tipo* e *(*nome)). Essendo il soggetto erogatore un intermediario esso è al tempo stesso erogatore del servizio composto e fruitore dei singoli servizi componenti. La dualità di questo soggetto si riflette nella configurazione del soggetto fruitore.

```
<!-
Ministero1:  erogatore del servizio componente 1
->
<soggetto-spcoop tipo='SPC' nome='Ministero1'>
 <connettore tipo='http' nome='PdDMinistero1'>
  <property nome='location' valore='http://localhost:8080/openspcoop/PA' />
 </connettore>
<!- Servizio componente 1 ->
 <servizio tipo='SPC'
  nome='AutenticazioneUtente' accordo-servizio='ASAutenticazioneUtente'>
```

```
   <fruitore tipo='SPC' nome='MinisteroErogatore' />
  </servizio>
</soggetto-spcoop>
<!-
Ministero2:  erogatore del servizio componente 2
->
<soggetto-spcoop tipo='SPC' nome='Ministero2'>
  <connettore tipo='http' nome='PdDMinistero2'>
   <property nome='location' valore='http://localhost:8080/openspcoop/PA' />
  </connettore>
  <!- Servizio componente 2 ->
  <servizio tipo='SPC'
   nome='AutorizzazioneUtente' accordo-servizio='ASAutorizzazioneUtente'>
   <fruitore tipo='SPC' nome='MinisteroErogatore' />
</servizio>
</soggetto-spcoop>
```

La forma dei singoli soggetti che erogano i servizi componenti coincide con la forma dei soggetti per realizzare gli Accordi di Servizio.

Capitolo 7

Implementazione dei casi d'uso

In questo capitolo introduciamo il framework scelto per l'orchestrazione dei servizi complessi JBPM (Java Business Process Management). La trattazione include anche una descrizione sommaria di JBossAS, l'application server necessario al funzionamento di JBPM. Saranno inoltre chiariti concetti implementativi pratici riguardo l'uso del linguaggio BPEL e della sua relativa implementazione nell'engine JBPM-BPEL. In questo capitolo viene espressamente mostrato il codice necessario allo sviluppo di un processo in BPEL a partire da zero. Sarà in ultimo approfondita l'implementazione dei casi d'uso realizzati durante questo lavoro di tesi.

7.1 JBoss Application Server

JBossAS [JBoss] è un application server J2EE sviluppato da JBoss (http://www.JBoss.org). E' un application server ad alte prestazioni per piattaforme di classe enterprise per la creazione e lo sviluppo di applicazioni e-business. Combinando una architettura robusta e al tempo stesso flessibile con una licenza no-cost per software open source, JBossAS è rapidamente diventato il più popolare

sistema middleware per sviluppatori, piccole e grandi imprese. Questo application server è largamente conosciuto per la sua potenza e la sua semplicità grazie al suo supporto per il modello di programmazione Enterprise Java Bean 3.0 [EJB]. EJB3 riduce considerevolmente il modello di programmazione Java Enterprise espandendo la potenza dei servizi per la piattaforma Java Enterprise Edition per semplificare gli oggetti Java tramite le annotazioni standard Java. Grazie a queste caratteristiche, JBossAS permette alle organizzazioni dell'Information Technology di raggiungere grossi risultati in minor tempo. JBoss è quindi un application server per architetture orientate al servizio (vedi secondo capitolo) che ha i suoi punti di forza grazie anche alle seguenti caratteristiche:

- supporto per Java Management Extensions (JMX) per la configurazione dei servizi via console.
- supporto per Java Server Faces, un framework standard per applicazioni Web.
- funzialità di caching e clustering per applicazioni complesse.
- supporto totale per web Service, dallo sviluppo alla sicurezza avanzata.

7.2 JBPM (java Business Process Management)

In questa sezione descriviamo JBPM, un framework per linguaggi a processi. Il prodotto è disponibile presso il sito di JBoss (http://www.JBoss.com).

7.2.1 JBPM

JBoss JBPM (java Business Process Management) è un framework estensibile e flessibile per linguaggi a processi. jPDL (java Process Description Language) è un linguaggio a processi che è costruito sopra il comune framework JBPM. jPDL è intuitivo e permette di esprimere processi produttivi

in maniera grafica in termini di operazioni, stati d'attesa per comunicazioni asincrone, timers, azioni automatizzate ecc... Per collegare queste operazioni insieme, jPDL ha un meccanismo potente ed estensibile per il controllo del flusso. jPDL ha dipendenze minime e può essere usato tanto facilmente quanto una libreria Java (vedi figura 7.2) ma può essere anche usata in ambienti dove sono previste computazioni ad alto throughput impiegandolo in cluster di application server J2EE (vedi figura 7.1). Il framework JBPM si dimostra flessibile perchè definisce una tecnologia di base per tutti i tipi di linguaggi a processi.

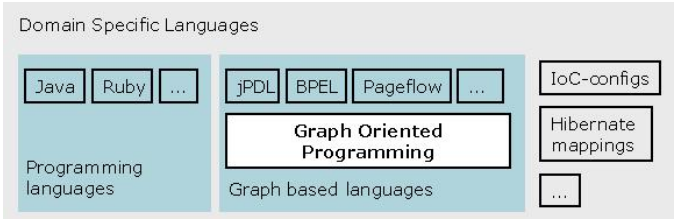

Figura 7.1: Il linguaggio a processi e la sua collocazione

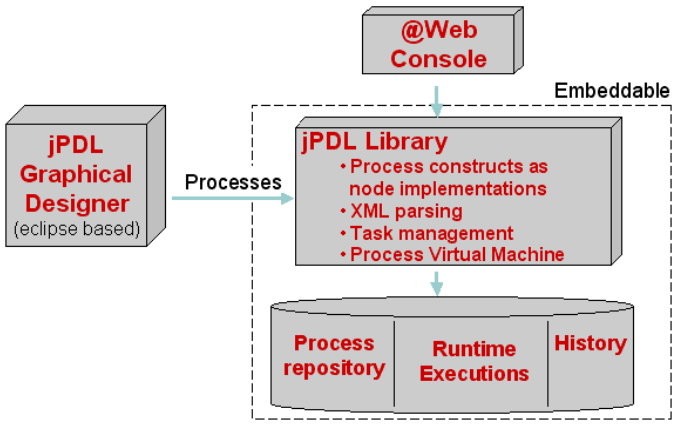

Figura 7.2: Panoramica sui componenti jPDL

7.2.2 JBPM per la gestione dei processi produttivi

L'obiettivo di BPM è di rendere un'organizzazione più efficiente nell'esecuzione dei suoi compiti. Il primo passo è descrivere come il lavoro deve essere fatto in una organizzazione. Definiamo un processo produttivo come una descrizione di come le persone e il sistema lavorano insieme per compiere un determinato compito. Una volta che il processo produttivo è stato definito, si effettuano eventuali ottimizzazioni (Business Process Reengineering, BPR), con meccanismi di controllo e statistiche.
Un altro modo di migliorare l'efficienza può essere automatizzare una parte o l'intero processo produttivo utilizzando l'information technology.
Quindi automatizzare e ottimizzare i processi produttivi sono le due principali tecniche per migliorare l'efficenza di una organizzazione.
L'adozione di processi produttivi ottimizzati è fondamentale per le aziende: l'esecuzione di un processo produttivo efficiente permette loro di risparmiare tempo e denaro, sia in termini di macchinari che in termini di forza lavoro impiegati. Lo sforzo e il costo addizionale richiesti ad una azienda per analizzare, automatizzare e ottimizzare i processi produttivi porta necessariamente ad un guadagno a breve e a lungo termine.
Un altro obiettivo principale dei sistemi BPM è di facilitare l'automazione dei processi produttivi. Nella costruzione del software per processi produttivi si possono distinguere due ruoli: lo sviluppatore e l'analista del processo. In piccoli team, questi lavori possono essere realizzati dalla stessa persona. L'analista di processo studia e descrive il processo produttivo e specifica i requisiti software, mentre lo sviluppatore crea software eseguibile.
Le suite tradizionali di BPM cercano di partire dal modello dell'analista dei processi e si muovono verso il software eseguibile. In seguito cercano di minimizzare le necessità di abilità tecniche in maniera tale che l'analista di processi possa produrre facilmente il software eseguibile a partire dal modello. Questo scenario è descritto in figura 7.3.

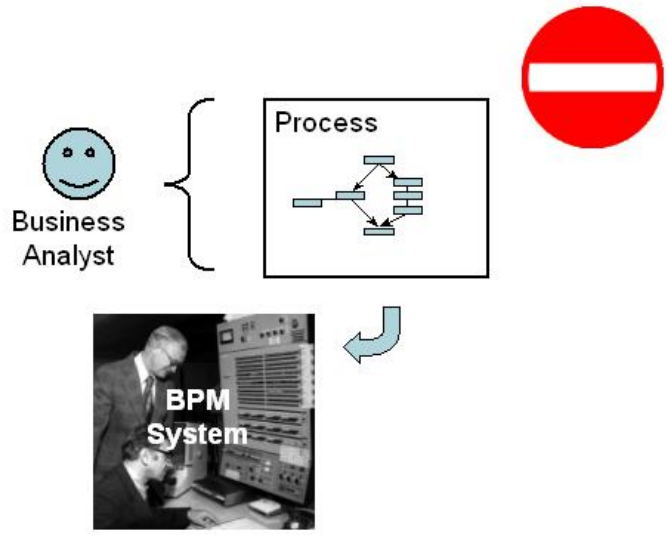

Figura 7.3: La separazione dei ruoli nei processi produttivi

Invece, nella visione di JBPM, l'idea centrale è che l'analista di processi e lo sviluppatore comunicano in un linguaggio condiviso con l'aiuto della visione grafica del processo. Le abilità tecniche saranno sempre necessarie per lo sviluppo del software. L'analista software è responsabile per la rappresentazione grafica e non dovrebbe essere forzato a tener conto degli aspetti tecnici che saranno necessari per rendere il processo eseguibile. Gli aspetti tecnici, comunque, non dovrebbero richiedere cambi alla rappresentazione grafica del processo.

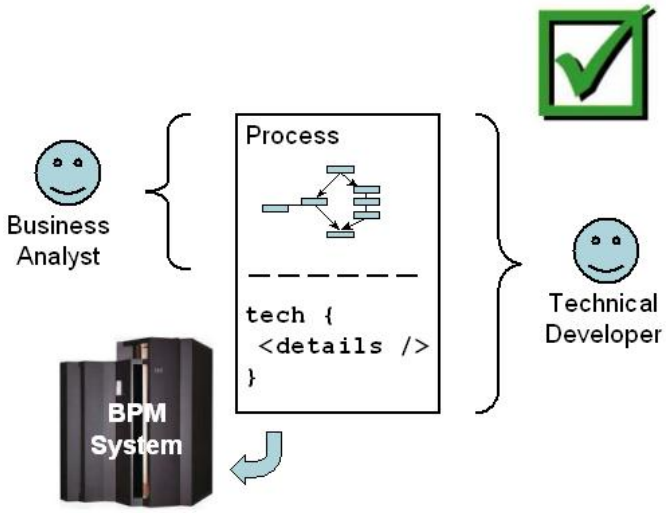

Figura 7.4: I ruoli nell'implementazione e gestione dei workflow

7.3 Realizzare workflow in JBPM-BPEL

7.3.1 Struttura dei File per un Servizio Workflow

In questa sezione presentiamo i passi necessari per realizzare un servizio di tipo worfklow completo a partire da zero. E' importante specificare che un workflow scritto in WS-BPEL viene visto dall'esterno come un comune Web Service che è possibile invocare con un comune client. La trattazione seguente mostrerà la struttura delle directory e dei file da utilizzare insieme ai task Ant scritti per ottenere un processo WS-BPEL correttamente funzionante sotto JBoss.
La descrizione di un workflow sarà costituita dalle directory e dai file seguenti:

- *src*: in questa cartella si mantiene il codice java associato al Web Service del workflow.
- *deploy*: in questa cartella si mantengono in maniera ben separata il

codice relativo al servizio di tipo workflow e altre informazioni collegate al deploy

- file *build.xml e local_env.xml*: essi contengono codice Ant riusabile per qualunque servizio scritto rispettando questa struttura.

La directory deploy contiene:

- Directory *bpel*: in questa directory sono contenuti il file WS-BPEL che implementa la logica di orchestrazione, il documento WSDL che presenta il processo di tipo workflow come un comune Web-Service e i documenti WSDL che il workflow deve orchestrare. Questi ultimi WSDL possono essere non completamente specificati, cioè possono mancare delle informazioni relative al deploy dei servizi orchestrati (ad esempio, indirizzo).
- Directory *resources*: questa directory contiene i file necessari a:
  - Creare le interfacce java e i tipi degli oggetti associati a questo servizio a partire dal documento WSDL associato al workflow (file *wscompile.xml*).
  - Fornire all'engine JBPM-BPEL il mapping tra il workflow e le classi java che lo implementano(cioè un file che mostra come un oggetto XML viene convertito in un oggetto java e viceversa) e gli indirizzi dei servizi che il workflow contatterà a run-time (file *bpelapplication.xml* in Web/classes).
  - Fornire ad JBoss le informazioni per mappare il Web Service associato al workflow su una servlet come per qualunque altro Web Service di cui si vuole effettuare il deploy (file Web.xml e WebService.xml in Web)

7.3.2 Deploy del Servizio Workflow

Quando le directory e i file sono stati organizzati come specificato nel paragrafo precedente, bisogna assicurarsi che sotto JBoss sia già attivo l'engine JBPM-BPEL. In caso affermativo è possibile utilizzare gli script Ant scritti durante questo lavoro di tesi per velocizzare il più possibile l'operazione di *deploy* del servizio che implementa questo workflow. In questo paragrafo verrà descritto ogni singolo task Ant a livello funzionale mentre il loro utilizzo verrà mostrato in un esempio pratico che è possibile comprendere nel paragrafo successivo.

- ant deploy-definition: Questo task crea un file compresso a partire dal processo scritto in BPEL, dal relativo WSDL descrittivo e dai WSDL dei servizi coinvolti e verifica la sintassi e la semantica del processo inviando tale archivio all'engine BPEL. L'engine BPEL che è in esecuzione sotto JBoss come un qualunque altro servizio Web risponde a tale operazione con un codice HTTP, 200 se è accettato, 505 se sono occorse delle eccezioni durante l'analisi dell'archivio. L'archivio viene accettato se il compilatore WS-BPEL e il parser WSDL hanno accettato i rispettivi file. Questa operazione di fatto registra la definizione di questo progetto sull'engine JBPM-BPEL, cui manca solo il Web Service associato (file WAR). I seguenti passi spiegano come ottenere questo altro archivio.

- ant generate-Service: Questo task analizza il WSDL descrittivo associato al processo e provvede alla creazione del relativo servizio in maniera concreta, iniziando cioè a mostrare quali siano le modalità per contattare il Web Service che rappresenta il processo. Il risultato di questo task è una serie di documenti WSDL che includono i servizi usati dal processo e i WSDL che definiscono il processo stesso. L'informazione relativa all'indirizzo è fornita in maniera parametrica

tramite una particolare URL (*REPLACE_WITH_ACTUAL_URI*) che indica che l'indirizzo del servizio sarà definito a seconda del punto in cui verrà effettuato il deploy.

- ant generate-artifacts: Questo task utilizza i documenti WSDL prodotti dal task generate-Service e ne analizza gli eventuali tipi XML coinvolti. Questo task richiede l'utilizzo di *Java Web Service Developer Pack*, una raccolta di tool per chi crea e sviluppa Web Service. Tra i tool a disposizione viene usato *WsCompile* che effettua l'analisi di un documento WSDL e produce le classi associate al servizio. Il tool provvede a creare una interfaccia Java da implementare per la logica del servizio oltre alla conversione in tipi Java dei tipi XML utilizzati dai vari documenti WSDL. Il tool infine crea un file per collegare i tipi XML ai tipi Java per le operazioni RPC, in una parola le informazioni di binding e mapping. Tra le classi generate è bene ricordare quella fondamentale, che ha lo stesso nome dell'operazione del servizio e che va implementata per la completezza del servizio stesso.

- ant deploy: Questo task utilizza tutti i documenti prodotti dai task generate-Service e generate-artifacts per creare una struttura di directory e di file ammessa dall'Application Server JBoss e che corrisponda al formato del servizio per cui l'engine BPEL aveva accettato le definizioni con il task deploy-definition. Il risultato di questo task è la generazione di un archivio Web (WAR) e il suo relativo deploy in JBoss. Al contrario di quanto accade per il task deploy-definition non viene fornita alcuna risposta e il riscontro sul corretto deploy deve essere ricercato analizzando il file di log di JBoss. Un corretto deploy in genere è identificato dal sottoservizio ServiceEndPointManager di JBoss che comunica la URL dove è possibile invocare il servizio.

7.4 I Casi di Prova in BPEL

Prima di passare all'implementazione dei casi d'uso per l'Accordo di Cooperazione sono stati realizzati vari casi di prova in BPEL. Lo scopo funzionale di questi casi d'uso è stato familiarizzare con il linguaggio e la sua implementazione scelta oltre a mostrare capacità avanzate di elaborazione del linguaggio stesso. Per il caso d'uso ElevaQuadrato e solo per questo caso viene mostrato il contenuto dei file coinvolti nella realizzazione e come utilizzare gli script Ant.

7.4.1 Il Caso d'Uso ElevaQuadrato

Questo caso d'uso è un caso minimale ma permette di familiarizzare con il linguaggio e con i relativi concetti base. Presentiamo questo caso d'uso in maniera totale, specificando anche il ruolo di ogni file contenuto nella directory del progetto e il significato semantico degli elementi all'interno di ogni singolo file. L'organizzazione dei file rispetta quella descritta nei

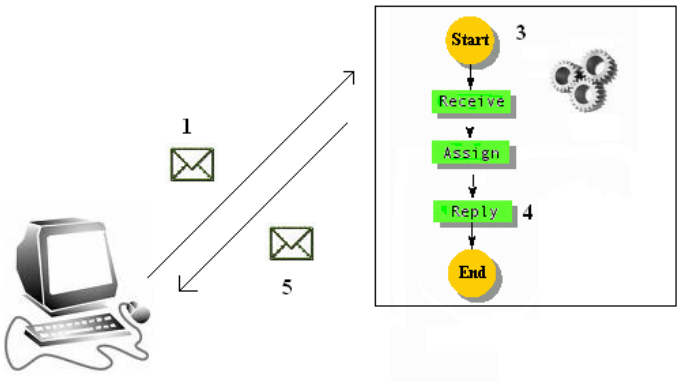

Figura 7.5: ElevaQuadrato in BPEL

paragrafi precedenti ed è la seguente:

```
build.xml
local_env.xml
src/
  Quadrato_Impl.java
deploy/
  bpel/
   quadrato.bpel
   quadrato.wsdl
   META-INF/
     bpel-definition.xml
  resources/
    wscompile.xml
    web/
     web.xml
     webservice.xml
     classes/
       bpel-application.xml
```

Iniziamo la descrizione del workflow ElevaQuadrato mostrando prima la logica di orchestrazione che in questo caso non coinvolge altri servizi (il file *quadrato.bpel*) e quindi il documento WSDL per la definizione astratta di questo workflow come Web Service (il file *quadrato.wsdl*).

Il seguente codice WS-BPEL per il processo ElevaQuadrato è contenuto in *quadrato.bpel*.

```
<process name='ElevaQuadrato'
 targetNamespace='http://jbpm.org/examples/quadrato'
 xmlns='http://schemas.xmlsoap.org/ws/2003/03/business-process/'
 xmlns:tns='http://jbpm.org/examples/quadrato'
 xmlns:bpel='http://schemas.xmlsoap.org/ws/2003/03/business-process/'>
 <partnerLinks>
```

```
  <!- establishes the relationship with the caller agent ->
  <partnerLink name='caller' partnerLinkType='tns:Quadrato-Caller'
  myRole='Quadrato' />
 </partnerLinks>
 <variables>
  <!- holds the incoming message ->
  <variable name='request' messageType='tns:rootMessage' />
  <!- holds the outgoing message ->
  <variable name='response' messageType='tns:squareMessage' />
 </variables>
 <sequence>
  <!- receive the root ->
  <receive operation='quadrato' partnerLink='caller' portType='tns:Quadrato'
  variable='request' createInstance='yes' />
  <!- multiplies the root integer ->
  <assign>
   <copy>
    <from expression='
    bpel:getVariableData('request', 'root') *
    bpel:getVariableData('request', 'root')' />
    <to variable='response' part='square' />
   </copy>
  </assign>
  <!- reply with the square ->
  <reply operation='quadrato' partnerLink='caller' portType='tns:Quadrato'
  variable='response' />
 </sequence>
</process>
```

Il processo (elemento <process>) è definito attraverso un nome (attributo *name*) e identifica uno spazio di nomi (attributi *targetNamespace* e *tns*) con connotazione identica a quella dei documenti di tipo WSDL. All'interno del processo è definito un solo elemento partnerLink di nome *caller*, il cui ti-

po (*tns:Quadrato-Caller*) è stato già definito nel documento quadrato.wsdl. Il partnerlink come già visto nel capitolo su BPEL identifica il collegamento tra il processo e uno dei punti di entrata per l'invocazione dei servizi. In questo caso il partnerLink definisce per mezzo di quale operazione del servizio associato a questo processo viene effettuata la creazione dell'istanza del processo. La definizione delle variabili globali segue quella dei partnerLink: queste variabili sono usate per memorizzare i messaggi scambiati tra client, processo e servizi e realizzano di fatto *lo stato del processo*. Le operazioni contenute nella sequenza principali prevedono:

- (*receive*). La ricezione di un messaggio sul partnerLink identifica anche la creazione del processo stesso (attributo *createInstance*)
- (*assign*). Il messaggio ricevuto viene processato: se ne estrapola il campo 'root' e lo si moltiplica per se stesso. Tale risultato viene copiato nel messaggio finale.
- (*reply*). La risposta appena costruita viene rimandata indietro al client.

Il file *quadrato.wsdl* contiene il seguente codice: si tratta di una descrizione di un Web Service come tutti gli altri, con la sola eccezione che definisce il tipo del partnerLink che il processo usa.

```
<definitions targetNamespace='http://jbpm.org/examples/quadrato'
 xmlns='http://schemas.xmlsoap.org/wsdl/'
 xmlns:tns='http://jbpm.org/examples/quadrato'
 xmlns:xsd='http://www.w3.org/2001/XMLSchema'
 xmlns:plt='http://schemas.xmlsoap.org/ws/2003/05/partner-link/'>
 <!- characterizes the relationship between the Quadrato and its caller ->
 <plt:partnerLinkType name='Quadrato-Caller'>
  <plt:role name='Quadrato'>
   <plt:portType name='tns:Quadrato' />
  </plt:role>
```

```
  <!- the Caller does not provide Services to the Quadrato,
  this is why we omit the 'Caller' role ->
 </plt:partnerLinkType>
 <!- carries the integer value ->
 <message name='rootMessage'>
  <part name='root' type='xsd:int' />
 </message>
 <!- carries the result ->
 <message name='squareMessage'>
  <part name='square' type='xsd:int' />
 </message>
 <!- describes the interface presented to callers ->
 <portType name='Quadrato'>
  <operation name='quadrato'>
   <input message='tns:rootMessage' />
   <output message='tns:squareMessage' />
  </operation>
 </portType>
</definitions>
```

Il file *bpel-definition.xml* è richiesto dall'engine BPEL e serve a fornire le istruzioni per comprendere la struttura dell'archivio compresso che verrà generato col prossimo comando Ant.

```
<bpelDefinition location='quadrato.bpel' xmlns='http://jbpm.org/bpel'>
 <!- makes WSDL interface elements available to the process ->
 <imports>
  <wsdl location='quadrato.wsdl' />
 </imports>
</bpelDefinition>
```

Dati questi tre file è possibile eseguire il comando

```
ant deploy-definition
```

per effettuare la registrazione di questo processo nell'engine. La corretta esecuzione viene presentata come

```
   INFO [/jbpm-bpel] processDeployServlet:  deploying process definition:
file=file:/.../quadrato-process.zip
INFO [BpelReader] read wsdl definitions:  quadrato.wsdl

INFO [BpelReader] read bpel process:  quadrato.bpel

INFO [/jbpm-bpel] processDeployServlet:  deployed process definition:
ElevaQuadrato
```

L'esecuzione del comando

```
ant generate-Service
```

effuttua la generazione in una directory locale dei documenti WSDL che rendono il documento quadrato.wsdl la definizione completa di un Web Service. I passi che seguono servono a costruire un Web Service in un formato comprensibile per JBoss.

Il file *wscompile.xml* è associato all'utilizzo di WSCompile, una libreria richiesta per la costruzione di WAR per workflow in JBPM-BPEL. Una delle operazioni di questa libreria è di creare classi java a partire da documenti WSDL per ogni oggetto trovato nel documento WSDL e interfacce java per qualunque gruppo di operazioni trovate (i.e. *portType*). La configurazione dell'operazione è semplice: basta specificare per quale file bisogna generarne le interfacce java (valore dell'attributo *location*) e a quale package appartengono queste interfacce (valore dell'attributo *packageName*).

```
<configuration xmlns='http://java.sun.com/xml/ns/jax-rpc/ri/config'>
 <wsdl location='target/resources/Web/wsdl/quadrato-Service.wsdl'
  packageName='org.jbpm.bpel.ElevaQuadrato' />
</configuration>
```

Configurato appropriatamente questo file, è possibile eseguire il comando

```
ant generate-artifacts
```

per creare le interfacce java. La classe che implementa il servizio (qua-

drato_Impl.java) deve implementare l'interfaccia associata al portType di questo servizio. La logica di questa classe non viene mai utilizzata ma la sua definizione è necessaria allo sviluppo di un Web Service sotto JBoss.

il file *Web.xml* definisce il collegamento tra la definizione WSDL del processo e la classe che ne implementa la logica. Si crea quindi un servlet, cioè una applicazione che esegue codice java per Web Service. Si identifica quindi il nome della servlet e per questo nome si specifica:

- Quale classe ne realizza la logica (*servlet-class*).
- Quale sia l'indirizzo per contattarlo (*url-pattern*).

```
<Web-app version='2.4' xmlns='http://java.sun.com/xml/ns/j2ee'
 xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
 xsi:schemaLocation='http://java.sun.com/xml/ns/j2ee
 http://java.sun.com/xml/ns/j2ee/Web-app_2_4.xsd'>
 <servlet>
  <servlet-name>quadratoServlet</servlet-name>
  <servlet-class>org.jbpm.bpel.ElevaQuadrato.Quadrato_Impl</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>quadratoServlet</servlet-name>
  <url-pattern>/quadrato</url-pattern>
 </servlet-mapping>
</Web-app>
```

Il file *WebServices.xml* fornisce le informazioni per collegare i tipi definiti nel documento WSDL quadrato.wsdl e le classi generate da WSCompile. Ulteriori informazioni specificano come la parte concreta di quadrato.wsdl (i *port*) sia collegata alle interfacce create e quale sia il servlet che gestisce questo Web Service. In questo file si trovano informazioni per l'engine JBPM-BPEL (elemento *init-param*) che specificano quale sia il partnerLink che provoca la generazione del processo nell'engine.

```
<WebServices version='1.1' xmlns='http://java.sun.com/xml/ns/j2ee'
```

```
 xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
 xsi:schemaLocation='http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/j2ee_Web_Services_1_1.xsd'>
 <WebService-description>
  <!- descriptive name for the Service ->
  <WebService-description-name>Eleva al Quadrato</WebService-description-name>
  <!- WSDL Service file ->
  <wsdl-file>WEB-INF/wsdl/quadrato-Service.wsdl</wsdl-file>
  <!- Java to XML mapping file ->
  <jaxrpc-mapping-file>WEB-INF/jaxrpc-mapping.xml</jaxrpc-mapping-file>
  <port-component>
   <!- logical name for the port (unique within the module) ->
   <port-component-name>QuadratoPort</port-component-name>
   <!- WSDL port element (in Service.wsdl) ->
   <wsdl-port xmlns:portNS='http://jbpm.org/examples/quadrato'>
   portNS:QuadratoPort
   </wsdl-port>
   <!- Service endpoint interface class ->
   <Service-endpoint-interface>
   org.jbpm.bpel.ElevaQuadrato.Quadrato
   </Service-endpoint-interface>
   <!- associated servlet (in Web-app.xml) ->
   <Service-impl-bean>
   <servlet-link>quadratoServlet</servlet-link>
   </Service-impl-bean>
   <handler>
   <init-param>
   <description>
   name of the partner link served by this port
   </description>
   <param-name>partnerLinkHandle</param-name>
   <param-value>caller</param-value>
   </init-param>
```

```
    </handler>
   </port-component>
 </WebService-description> </WebServices>
```

Nel file *bpelApplication.xml* si specificano le URL usate dall'engine a runtime per contattare i servizi. Questo esempio, non coinvolgendo altri servizi, non richide una configurazione di questo file.

```
<bpelApplication name='ElevaQuadrato' xmlns='http://jbpm.org/bpel'
 xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
 xsi:schemaLocation='http://jbpm.org/bpel
 http://jbpm.org/bpel/bpel_application_1_0.xsd' />
```

L'esecuzione del comando

```
ant deploy
```

realizza un archivio Web (WAR) per un Web Service sotto JBoss. Il corretto deploy viene evidenziato da tracce nel log del server

```
[TomcatDeployer] deploy, ctxPath=/quadrato, warUrl=...
[IntegrationConfigurator] Message reception enabled for process:
ElevaQuadrato
[WSDLFilePublisher] WSDL published to:  file:/.../quadrato-Service.wsdl
[ServiceEndpointManager] WebService started:  http://.../quadrato/quadrato
```

Il servizio è ora pronto per poter essere utilizzato da un qualsiasi client Web Service correttamente configurato.

7.4.2 Routing Semplice e Routing Elaborato

A livello di logica di orchestrazione, questo caso d'uso è leggermente più elaborato del precedente. In questo caso d'uso, il processo deve contattare due servizi, inoltrando loro il messaggio ricevuto dal client e fornire come risposta al client entrambe le risposte ricevute. Questo caso d'uso mette in eviden-

za le caratteristiche di BPEL in merito agli assegnamenti automatizzati di messaggi SOAP.

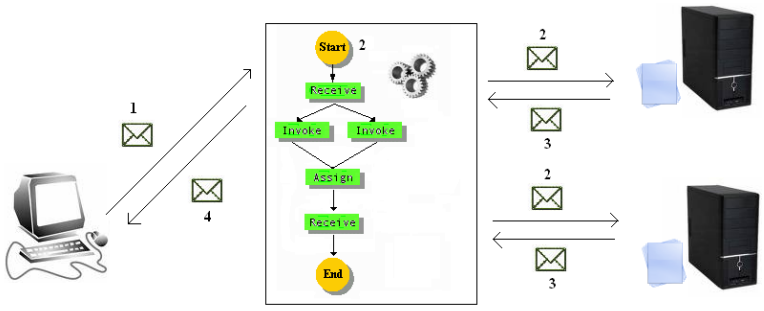

Figura 7.6: Routing in BPEL

I passi che si analizzano in figura 7.6 sono i seguenti:

1. Il client contatta il processo Route per inviare il suo messaggio (es. nome)

2. Il processo viene attivato e inoltra il messaggio ricevuto dal client ai due servizi.

3. Ricevute le risposte dai servizi, provvede a ricavarne il contenuto (es. 'Servizio 1 nome OK') e a ricomporle per creare la risposta per il client.

4. Il processo invia la risposta al client, chiudendo la comunicazione.

Nel caso d'uso di routing elaborato, il messaggio di risposta del primo servizio viene usato come input del secondo servizio. L'esecuzione prosegue come già visto per il routing semplice. A differenza del caso d'uso ElevaQuadrato, in questo caso d'uso il processo contatta Web Service esterni, prepara ed elabora messaggi SOAP sia per il rispondere al client sia per formulare le richieste ai Web Service. L'elaborazione dei messaggi è estremamente facilitata dalla gestione automatica dei namespace nell'engine JBPM-BPEL.

7.4.3 Uso di Tipi XML e Query XPath 1.0

BPEL è un linguaggio basato su XML e i tool messi a disposizione con l'engine permettono di effettuare un controllo sintattico e semantico del codice BPEL che viene scritto. Il validatore XML fornito realizza controlli a livello di messaggi SOAP ma non relativamente all'intero contenuto del messaggio. In altre parole, l'utilizzo di un tipo XML non default in un documento WSDL usato da BPEL viene riconosciuto solo a livello di nome e non a livello di contenuto.

Per utilizzare quindi i campi contenuti all'interno di un tipo XML bisogna ricorrere a query Xpath. Supponiamo di avere un tipo XML di nome 'Utente' definito come segue:

```
<complexType name='Utente'>
 <sequence>
  <element name='nome' type='xsd:string'/>
  <element name='cognome' type='xsd:string' />
 </sequence>
</complexType>
```

Una tipica espressione XPath per leggere il nome è '/Utente/nome'. Questo caso d'uso pone quindi l'accento sui tipi XML e sulle query XPath nel linguaggio BPEL. L'utilizzo di questo caso d'uso è stato utilizzato nel caso d'uso Login, a cui si rimanda per una descrizione più completa.

7.4.4 Indirizzamento Dinamico dei Servizi

L'implementazione di BPEL nell'engine JBPM-BPEL risolve l'indirizzo dei servizi da contattare tramite un registro XML di nome bpel-application che contiene delle URL esplicite di servizi espressi sotto forma di documenti WSDL. L'engine quindi ottiene dinamicamente tutte le informazioni necessarie a contattare i servizi con una data interfaccia prelevando i WSDL associati grazie alle URL nel registro e controllando che il servizio implementi le operazioni richieste in maniera concreta. L'implementazione attuale dell'en-

gine, però, impone la conoscenza statica, cioè a tempo di design, di questi indirizzi. In caso di compatibilità di più WSDL tra quelli con URL nota, viene scelto sempre il primo tra quelli registrati nella registro bpel-application. Un esempio di file *bpelapplication.xml* che definisce la struttura del registro per le URL è fatto in questo modo:

```
<bpelApplication name='AtmFrontEnd' xmlns='http://jbpm.org/bpel'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:schemaLocation='http://jbpm.org/bpel
http://jbpm.org/bpel/bpel_application_1_0.xsd'>
<ServiceCatalogs>
<urlCatalog contextUrl='http://localhost:8080/'>
<wsdl location='ticket/ticketIssuer?wsdl' />
<wsdl location='account/accountSystem?wsdl' />
</urlCatalog>
</ServiceCatalogs>
</bpelApplication>
```

Questo tipo di implementazione è:

- poco flessibile, perchè impone che l'indirizzo dei servizi da contattare sia sempre già noto a tempo di design
- in alcune situazioni, il comportamento del codice non è quello descritto a causa della scelta basata su URL nel catalogo.

Per superare questa difficoltà, in BPEL esiste il tipo EndpointReference che permette di definire l'indirizzo di un servizio e il relativo Service. Quando questo oggetto viene assegnato ad un partnerlink, la ricerca nel URLCatalog viene annullata e viene direttamente contattato il servizio all'indirizzo indicato. In questo caso d'uso il processo deve contattare il servizio previa ricezione dell'indirizzo da un altro servizio di tipo 'Registro' che ha la logica semplice di ritornare l'indirizzo del servizio richiesto in parametro. I passi rilevanti sono i seguenti:

1. Il client contatta il servizio Dinamico con un tipo XML contenente due stringhe, un 'nomeutente' e un 'nomeservizio'.

2. Il processo si attiva e, rilevato il 'nomeservizio', contatta il Registro per ottenere la URL.

3. Se la URL è 'NO_URL', una risposta con esito negativo viene ritornata al client.

4. Se la URL non è 'NO_ URL' si contatta il servizio all'indirizzo richiesto e la relativa risposta viene raccolta.

5. Il processo risponde al client e pone fine alla comunicazione.

7.5 Implementazione dei Casi d'Uso per il Formato degli Accordi di Cooperazione

L'implementazione dei casi d'uso di prova utilizzati nelle prime fasi della tesi erano volti a prendere confidenza con la programmazione orientata ai grafi, il funzionamento del linguaggio BPEL e la relativa implementazione nell'engine JBPM-BPEL. La realizzazione di queste prime applicazioni è stata effettuata usando codice simile agli esempi di riferimento contenuti all'interno della distribuzione utilizzata dell'engine. I Web Service scritti durante questa fase sono stati realizzati utilizzando una libreria dell'engine, WsCompile, che genera delle classi in stile stub/skeleton per tutte le procedure di tipo RPC. Inoltre, i client realizzati erano anch'essi dei Web Service JBoss che implementano l'interfaccia TestCase presente nel framework JUnit di Sun per la realizzazione di test.
Per realizzare, invece, i casi d'uso per OpenSPCoop e i relativi client di prova sono stati utilizzati appositi strumenti appartenenti al SOAP Engine Axis, nella fattispecie Java2WSDL e WSDL2Java. Questo per mantenere la linea scelta con le precedenti distribuzioni di OpenSPCoop. Questo cambio

di strumenti ha creato la necessità di analizzare i messaggi SOAP scambiati tra processo e servizi per analizzare condizioni di errore inattese. E' stato utilizzato un analizzatore di pacchetti TCP disponibile in ambiente Linux di nome WireShark che ha permesso di rilevare il corretto formato dei messaggi scambiati tra i processi e i servizi. Parte della composizione automatizzata di messaggi in BPEL si è persa e questo ha allungato i tempi di sviluppo dei casi d'uso stessi. Come già descritto nel capitolo precedente, i casi d'uso individuati sono tre e riguardano in ordine:

- la verifica del funzionamento dell'Accordo di Cooperazione e dei mezzi necessari a realizzare una orchestrazione tra servizi.
- l'utilizzo di tipi XML, la composizione e l'inoltro di messaggi.
- l'invocazione di una famiglia di servizi con interfaccia comune con indirizzo non noto.

7.5.1 La porta di dominio OpenSPCoop come proxy SOAP

La porta di dominio OpenSPCoop può essere anche utilizzata come proxy SOAP. Ne viene qui riproposto il funzionamento in quanto basilare per il meccanismo di invocazione di Web Service da parte dei workflow. Per utilizzare la porta di dominio in questo modo occorre integrare i servizi applicativi del proprio dominio attraverso la modalità d'integrazione *trasparente*. Questa modalità prevede che il servizio applicativo utilizzi (in caso di porta delegata) o esponga (in caso di porta applicativa) le interfacce applicative native dei servizi, così come registrate negli accordi di servizio; in tal caso la Porta di Dominio agisce come un proxy SOAP trasparente con funzionalità di imbustamento e sbustamento eGov dei messaggi applicativi; utilizzando questa modalità, gli applicativi potranno continuare ad operare esattamente come se stessero interagendo direttamente con il servizio applicativo dell'altro Ente. L'invocazione della porta delegata in modalità trasparente può

essere realizzata tramite gli strumenti del linguaggio di programmazione nativo del servizio applicativo, utilizzando ad esempio stub creati tramite il proprio ambiente di sviluppo Web Services (ad esempio WSDL2JAVA in Axis), facendo riferimento direttamente al WSDL del servizio destinazione. In questo caso la principale modifica rispetto all'invocazione dell'effettivo servizio destinazione sarà la URL utilizzata per l'invocazione http, che dovrà essere quella corrispondente alla porta delegata del servizio esposta dalla PdD.

7.5.2 Caso d'Uso Hello

Il caso d'uso Hello permette di verificare la corretta configurazione dell'engine JBPM-BPEL e l'utilizzo di Accordi di Cooperazione in OpenSPCoop. Questo caso d'uso coinvolge quattro soggetti, uno che richiede il servizio (MinisteroFruitore), uno che regola l'intero processo (MinisteroErogatore) e i due soggetti che erogano i servizi da coordinare (Ministero1 e Ministero2). Il funzionamento del processo associato a questo Accordo di Cooperazione può essere riassunto nei seguenti passi:

- Il MinisteroFruitore contatta il MinisteroErogatore inviando il proprio nome.
- Il MinisteroErogatore contatta il Ministero1 per ottenere la stringa Hello seguita dal nome ricevuto in precedenza.
- Il MinisteroErogatore contatta il Ministero2 per confermare che la precedente operazione è andata a buon fine e chiederne il tracciamento.
- Il MinisteroErogatore conclude le comunicazione inviando la stringa Hello seguita dal nome ricevuto dal MinisteroErogatore.

I passi da effettuare coinvolgono:

- configurazione:

- configurazione del Registro Servizi di OpenSPCoop per la registrazione dell'Accordo di Cooperazione definito tra i due soggetti, registrazione dei soggetti SPCoop e del servizio che istanzia l'Accordo di Cooperazione.
- configurazione del dominio MinisteroErogatoreSPCoopIT nella porta di dominio OpenSPCoop: registrazione di una porta applicativa per il servizio SPC/Hello
- configurazione del dominio MinisteroFruitoreSPCoopIT nella porta di dominio OpenSPCoop: registrazione della porta delegata utilizzata dal soggetto SPC/MinisteroFruitore per invocare il servizio SPC/Hello.
- configurazione del dominio Ministero1SPCoopIT e Ministero1SPCoopIT nella porta di dominio OpenSPCoop: registrazione degli accordi di servizio.

Solo per questo caso d'uso proponiamo qui di seguito una configurazione reale del Registro Servizi. Per gli altri casi d'uso la configurazione varia nei nomi e nelle URL ma rimane sostanzialmente la stessa.

```
<!- Accordi di Cooperazione ->
 <accordo-cooperazione nome='Hello'
  descrizione='Esempio di un Accordo di Cooperazione' >
  <soggetto-referente tipo='SPC' nome='MinisteroErogatore' />
  <servizio-composto nome-accordo-servizio='ASHello' />
 </accordo-cooperazione>
<!- Accordo di Servizio del servizio composto ->
 <accordo-servizio nome='ASHello'
  profilo-collaborazione='sincrono' utilizzo-senza-azione='true' >
  <servizio-composto ws-bpel='http://...PathDelWSPBEL'>
   <servizio-componente accordo-servizio='ASHelloUtente' />
   <servizio-componente accordo-servizio='ASTracciaUtente' />
  </servizio-composto>
```

```
</accordo-servizio>
<!- Accordo di Servizio del servizio componente 1 ->
<accordo-servizio nome='ASHelloUtente'
 profilo-collaborazione='sincrono' utilizzo-senza-azione='true' />
<!- Accordo di Servizio del servizio componente 2 ->
<accordo-servizio nome='ASTracciaUtente'
 profilo-collaborazione='sincrono' utilizzo-senza-azione='true' />
<!-
 MinisteroFruitore:  fruitore del servizio composto
->
<soggetto-spcoop tipo='SPC' nome='MinisteroFruitore'>
 <connettore tipo='http' nome='PdDMinisteroFruitore'>
  <property nome='location' valore='http://localhost:8080/openspcoop/PA' />
 </connettore>
</soggetto-spcoop>
<!-
 MinisteroErogatore:
 - erogatore del servizio composto
 - fruitore dei servizi componenti
->
<soggetto-spcoop tipo='SPC' nome='MinisteroErogatore'>
 <connettore tipo='http' nome='PdDMinisteroErogatore'>
  <property nome='location' valore='http://localhost:8080/openspcoop/PA' />
 </connettore>
 <!- Servizio composto ->
 <servizio tipo='SPC' nome='Hello' accordo-servizio='ASHello'>
  <servizio-composto>
   <servizio-componente tipo-soggetto='SPC' nome-soggetto='Ministero1'
    tipo='SPC' nome='HelloUtente' />
   <servizio-componente tipo-soggetto='SPC' nome-soggetto='Ministero2'
    tipo='SPC' nome='TracciaUtente' />
  </servizio-composto>
  <fruitore tipo='SPC' nome='MinisteroFruitore' />
```

```
 </servizio>
</soggetto-spcoop>
<!-
 Ministero1:  erogatore del servizio componente 1
->
<soggetto-spcoop tipo='SPC' nome='Ministero1'>
 <connettore tipo='http' nome='PdDMinistero1'>
  <property nome='location' valore='http://localhost:9080/openspcoop/PA' />
 </connettore>
 <!- Servizio componente 1 ->
 <servizio tipo='SPC' nome='HelloUtente'
  accordo-servizio='ASHelloUtente'>
  <fruitore tipo='SPC' nome='MinisteroErogatore' />
</servizio>
</soggetto-spcoop>
<!-
 Ministero2:  erogatore del servizio componente 2
->
<soggetto-spcoop tipo='SPC' nome='Ministero2'>
 <connettore tipo='http' nome='PdDMinistero2'>
  <property nome='location' valore='http://localhost:7080/openspcoop/PA' />
 </connettore>
 <!- Servizio componente 2 ->
 <servizio tipo='SPC' nome='TracciaUtente'
  accordo-servizio='ASTracciaUtente'>
  <fruitore tipo='SPC' nome='MinisteroErogatore' />
 </servizio>
</soggetto-spcoop>
```

In sequenza evidenziamo:

- La configurazione dell'Accordo di Cooperazione. Questa prevede la specifica di un nome e di una descrizione. L'elemento soggetto-referente

identifica quale soggetto-spcoop è responsabile per l'orchestrazione mentre il servizio composto specifica solo un Accordo di Servizio: questo permette l'identificazione dell'erogatore e del fruitore del servizio composto.

- La configurazione dell'Accordo di Servizio. L'Accordo di Servizio configura il servizio composto identificando il documento WS-BPEL per l'orchestazione dei servizi e i servizi componenti che partecipano a questa orchestrazione. Ogni servizio componente riferisce un Accordo di Servizio.

- La configurazione degli Accordi di Servizio associati ad ogni servizio componente.

- La configurazione dei soggetti coinvolti nell'intera orchestrazione. Servono quattro soggetti per l'intera operazione:

  - Fruitore (*MinisteroFruitore*), cioè quello che usufruisce dell'intera operazione.
  - Erogatore (*MinisteroErogatore*), cioè quello che eroga il servizio composto.
  - Due componenti (*Ministero1* e *Ministero2*), cioè i soggetti che erogano i singoli servizi componenti.

  La configurazione più interessante è quella del soggetto erogatore, dal momento che le altre sono identiche a quelle della versione del Registro Servizi di OpenSPCoop senza Accordi di Cooperazione. Rispetto ad un comune soggetto bisogna specificare l'Accordo di Servizio associato, oltre ai singoli servizi componenti.

- Erogazione/fruizione del servizio

1. l'organizzazione fruitore MinisteroFruitoreSPCoopIT effettua l'invocazione del servizio interessato semplicemente interagendo attraverso la sua porta di dominio
2. il messaggio SOAP viene ricevuto, controllato e viene identificato il servizio relativo a questa richiesta; conclusa questa operazione viene preparata una busta eGov e si procede all'invio tramite la URL letta nel Registro Servizi. La porta di dominio rimane in attesa della risposta
3. ricevuta la busta eGov, si procede al controllo della busta stessa e in caso di controlli andati a buon fine si procede con la consegna del messaggio SOAP all'interno della busta eGov.
4. riconosciuta la modalità di consegna WSDL si procede all'invocazione del Web Service.
   (a) L'invocazione del servizio provoca la creazione di un processo workflow associato al Web Service Hello nell'engine JBPM-BPEL.
   (b) La logica del processo è definita come segue:
      – ricevuto il messaggio SOAP si preleva il campo 'nome'.
      – si invoca il Web Service HelloUtente che provvede ad aggiungere 'Hello' al 'nome' ricevuto nella richiesta inoltrata dal processo
      – si invoca il Web Service TracciaUtente, con il quale si traccia la richiesta di questo servizio
      – si prepara una risposta SOAP con il contenuto del messaggio del Web Service HelloUtente e lo si invia
      – tale risposta viene inviata al client di questo processo, cioè la porta di dominio stessa e l'istanza di processo associata a questo Web Service viene distrutta.

– Si prepara la busta eGov e la si invia alla porta di dominio che sbloccata conclude la comunicazione.

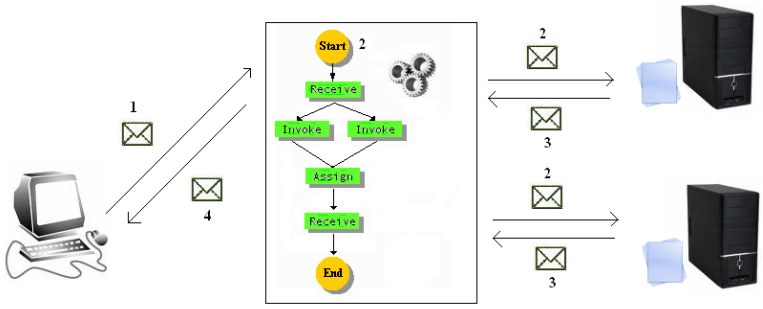

Figura 7.7: Il caso d'uso Hello in BPEL

Illustriamo per questo caso d'uso il comportamento delle porte di dominio che partecipano all'Accordo di Cooperazione Hello, senza riproporlo per i casi d'uso che verranno presentati in seguito.

I profili di collaborazione tra tutti i soggetti in gioco sono del tipo *request-response*, cioè il mittente contatta il destinatario e rimane in attesa della risposta. La comunicazione tra tutte le porte di dominio avviene tramite buste eGov. Mostriamo in sequenza quali azioni scatena lo scambio di buste eGov, che sono numerate in figura 7.7.

La busta eGov 1 avvia l'Accordo di Servizio relativo a questo Accordo di Cooperazione. Quando la porta di dominio del soggetto erogatore riceve la busta eGov ne preleva la busta SOAP e la inoltra al Web Service che fa da interfaccia al workflow. Questa azione causa la creazione dell'istanza di processo secondo le modalità già descritte all'inizio del capitolo. Il processo deve quindi contattare il servizio HelloUtente attraverso la URL nota nel suo ServiceCatalog. In questo caso, la porta di dominio viene usata come proxy SOAP, cioè serve a inviare messaggi SOAP in buste eGov. La busta eGov 2 arriva alla porta di dominio dov'è disponibile il servizio HelloUtente che una volta contattato produce la risposta (inviata con la busta eGov 3)

usando anche questa porta di dominio come proxy SOAP. La busta eGov 3 continua l'avanzamento del workflow che genera un nuovo messaggio in uscita per il secondo servizio TracciaUtente, imbustato nella eGov 4. La busta eGov 4, a pari della 2, causa la generazione della risposta del servizio TracciaUtente che viaggia nella busta eGov 5. La busta eGov 5 conclude le comunicazioni del workflow con i soggetti per la cooperazione. Il messaggio finale viene inviato con la busta eGov 6. La busta eGov 6 contiene la risposta per il soggetto fruitore e comporta la distruzione dell'istanza di processo all'interno della porta di dominio.

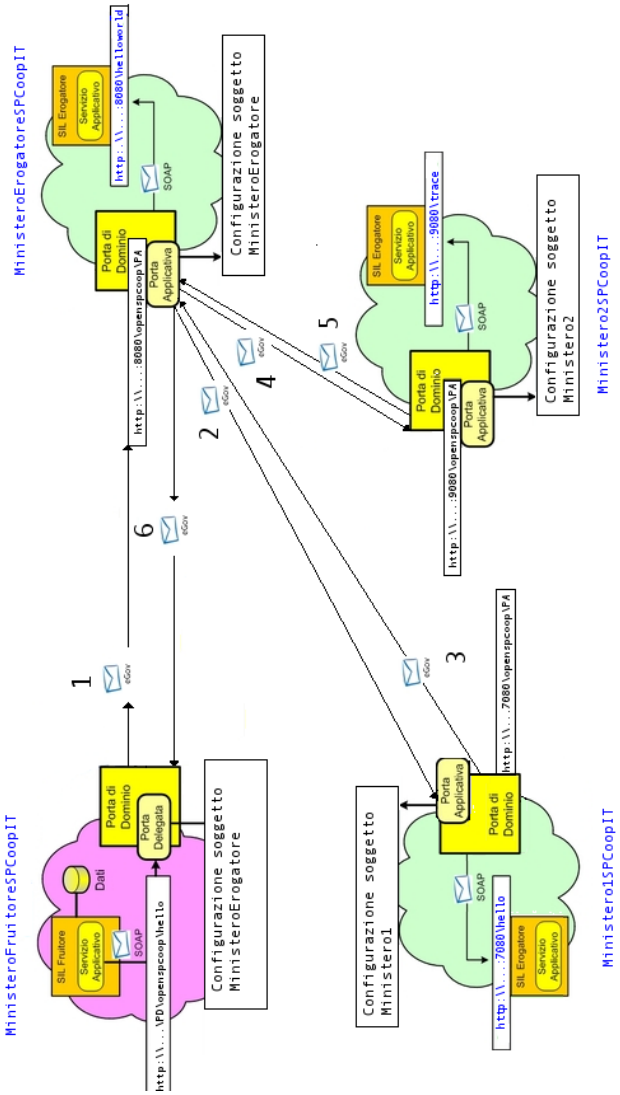

Figura 7.8: Hello World in OpenSPCoop: Lo scambio delle buste eGov.

7.5.3 Caso d'Uso Transazione

Il caso d'uso Transazione è il primo caso d'uso ad utilizzare due servizi fondamentali per la realizzazione di una operazione completa. Infatti nell'esempio precedente si poteva notare che il servizio TracciaUtente non svolge una azione fondamentale al completamento dell'operazione e la sua aggiunta si deve alla costruzione sintattica dell'Accordo di Cooperazione che prevede

la presenza di almeno due servizi. Nel caso d'uso Transazione sono necessari due servizi: il ControlloAutenticazione e il ControlloAutorizzazione. Questo caso d'uso permette di verificare l'utilizzo di oggetti XML nelle richieste e nelle risposte durante le comunicazioni con i Web Service. La logica del ControlloAutenticazione è semplice e intuibile: viene verificata l'esatta corrispondenza di nome utente e password in un apposito file di proprietà. Un file di proprietà è un file costituito da righe ASCII che rispettano la sintassi $nomeProprieta = valoreProprieta$. Un file fatto in questo modo viene letto grazie ad una classe di java.util, la classe *Properties*, che tramite il metodo load(), provvede a costruire una Map, cioè una tabella hash con chiavi $nomeProprieta$ e valori $valoreProprieta$. In caso di verifica corretta il nome dell'utente viene usato come chiave in un altro file di proprietà per recuperare l'identificativo associato a questo nome utente. Questo identificativo costituisce la risposta del servizio chiamato con questa operazione. Oltre a questa operazione, il servizio dispone anche di metodi per aggiungere, modificare e rimuovere utenti e valori all'interno di questi file.
La logica del Controllo di Autorizzazione è simile a quella del Controllo di Autenticazione: si verifica la presenza dell'identificativo ricevuto e in caso di presenza viene ritornata una lista di operazioni che l'utente autenticato può invocare. In caso di mancata presenza, la lista ritorna un codice di $identificativoNonTrovato$. Così come il servizio precedente, anche questo servizio ha ulteriori operazioni per aggiungere modificare e rimuovere identificativi e rispettive liste di operazioni autorizzate. Le definizioni dei tipi degli oggetti XML sono definiti in un file XSD (types.xsd) che viene condiviso sia dai WSDL dei Web Service utilizzati dal processo sia dal processo stesso. Il tipo Credenziali è usato dal ControlloAutenticazione mentre il tipo EsitoAutorizzazione dal ControlloAutorizzazione.

```
<xsd:complexType name='Credenziali'>
     <xsd:sequence>
```

```
            <xsd:element name='nome'
            type='xsd:string'></xsd:element>
            <xsd:element name='password'
            type='xsd:string'></xsd:element>
      </xsd:sequence>
</xsd:complexType>
<xsd:complexType name='EsitoAutenticazione'>
<xsd:sequence>
 <xsd:element name='IDUtenteAutenticato'
 type='xsd:string'></xsd:element>
 <xsd:element name='eventualeMotivoErrore'
 type='xsd:string'></xsd:element>
</xsd:sequence>
</xsd:complexType>
```

I passi da effettuare coinvolgono:

- configurazione: la configurazione è del tutto simile a quella del caso d'uso Hello, tranne naturalmente per i nomi e le descrizioni dei servizi utilizzati.

- Erogazione/fruizione del servizio

    1. come nel caso d'uso Hello.
    2. come nel caso d'uso Hello.
    3. come nel caso d'uso Hello.
    4. riconosciuta la modalità di consegna WSDL si procede all'invocazione del Web Service.
        (a) L'invocazione del servizio provoca la creazione di un processo workflow associato al Web Service Transazione nell'engine JBPM-BPEL.
        (b) La logica del processo è definita come segue:

- ricevuto il messaggio SOAP si preleva l'oggetto XML di tipo Credenziali che diventa parte del messaggio SOAP in uscita per il servizio ControlloAutenticazione.
- si invoca il Web Service ControlloAutenticazione che provvede a ritornare l'identificativo di questo nome utente in caso di corretta autenticazione, mentre un messaggio di errore ('NOK') e il relativo motivo di errore (es 'Utente non trovato' o 'Utente/Password errate') accompagnano la risposta per autenticazione incorretta.
- in base alla risposta del punto precedente, il processo
  - in caso di esito negativo conclude la comunicazione, preparando la risposta per il client e include la precedente risposta nel messaggio per il client.
  - in caso positivo si preleva l'identificativo che diventa parametro del messaggio SOAP in uscita per il servizio ControlloAutorizzazione.
    - si preleva la risposta del servizio, un oggetto XML EsitoAutorizzazione che diventa parte della risposta per il client.
    - la risposta precedentemente preparata viene inviata al client di questo processo, cioè la porta di dominio stessa e l'istanza di processo associata a questo Web Service viene distrutta.

5. come nel caso d'uso Hello.

In questo caso d'uso non sono stati considerati gli aspetti della sicurezza relativi a informazioni confidenziali (ad esempio il tipo *Credenziali*) perchè non rientra tra gli obiettivi di questo caso d'uso che è invece quello di mostrare l'utilizzo complesso di tipi XML in un Accordo di Cooperazione.

7.5.4 Caso d'Uso Sportello

Il caso d'uso Sportello permette di rappresentare in OpenSPCoop e con gli Accordi di Cooperazione uno tra gli scenari più comuni nelle organizzazioni con Architetture Orientate ai Servizi. Lo scenario che viene realizzato è l'aggiornamento di una informazione su una famiglia di servizi con la medesima interfaccia. Per mostrare come sia comune un servizio del genere in una organizzazione, specie se appartenente alla Pubblica Amministrazione, mostriamo un esempio.

Un utente ha cambiato residenza e quindi decide di informare l'Ufficio dell'Anagrafe al fine di aggiornare questa informazione. L'utente quindi si presenterà allo Sportello con un documento di identificazione (carta d'identità) e i documenti che attestano il cambio di residenza (es. contratto d'acquisto dell'immobile). In assenza di un mezzo di coordinazione tra gli uffici dello stesso comune, l'utente dovrebbe recarsi in tutti gli uffici per i quali è necessario il suo indirizzo di residenza, ad esempio l'Ufficio delle Imposte e altri uffici che regolano erogazioni di servizi della vita di tutti i giorni per quell'utente, come ad esempio la corrente elettrica, l'acqua corrente ecc... In presenza di uno strumento unificato di aggiornamento per il cambio di residenza, l'utente deve solo recarsi all'Ufficio dell'Anagrafe per cambiare in un solo passo la residenza presso tutti gli uffici convenzionati. Risulta ovvio come un tale esempio sia estendibile ad una vasta gamma di scenari molto vicini alla vita di tutti i giorni.

Il comportamento di base dell'engine JBPM-BPEL permette l'invocazione dei servizi offerti secondo l'utilizzo di un Service Catalog: con questo si intende un file XML con le URL dei servizi da invocare. Come già descritto nella sezione 7.4.4 (indirizzamento dinamico dei servizi), all'atto di una invoke in BPEL, il processo utilizza tale file per cercare tra gli URL noti il primo che implementa il servizio che sta per invocare. E' chiaro come una implementazione fatta in questo modo sia scomoda per invocare diversi servizi

con una stessa interfaccia. I problemi che si incontrano sono i seguenti:

- imposizione della conoscenza statica degli indirizzi di tutti i servizi. Sebbene questo non sia un vincolo progettuale difficoltoso, questo approccio di fatto richiede manipolazioni del file Service Catalog diretta per gestire l'iterazione e non permette in maniera semplice di realizzare un aggiornamento parallelo delle interfacce interessate.

- imposizione della conoscenza di tutte le casistiche. Potrebbe accadere che un dato utente non debba aggiornare le sue informazioni per tutte le interfacce note ma solo per quelle per cui è registrato. Questo impone di fatto di conoscere fra tutte le URL dei servizi possibili quelle da utilizzare realmente per l'utente che chiede tale aggiornamento. Generalizzare una logica per tutti gli utenti sarebbe proponibile ma risulterebbe complicata da realizzare direttamente in codice BPEL.

L'implementazione proposta è più flessibile perchè aperta alla conoscenza dinamica delle URL e degli utenti, e utilizza una variabile di tipo EndpointReference con la quale è possibile memorizzare le informazioni circa la parte concreta di un Web Service, cioè i suoi elementi Service e l'indirizzo SOAP.

```
<wsa:EndpointReference>
      <wsa:Address>wsa:AttributedURI</Address>
      <wsa:ReferenceProperties>
            wsa:ReferencePropertiesType</ReferenceProperties>
      <wsa:ServiceName PortName='xs:NCName'>
            wsa:ServiceNameType</ServiceName>
      <wsa:PortType>wsa:AttributedQName</PortType>
</wsa:EndpointReference>
```

E' possibile assegnare una variabile EndpointReference a un variabile partnerLink: un partnerLink memorizza la parte astratta (portType) di un servizio a tempo di design e le informazioni per contattare quel servizio a

tempo d'esecuzione. Mentre la parte astratta non è logicamente modificabile, la parte relativa all'indirizzo lo è. Nell'implementazione del partnerLink nell'engine infatti le informazioni relative al servizio vengono lette solo al momento della *invoke* in quanto necessarie a creare il client SOAP per il processo.

I servizi coinvolti in questo caso d'uso sono lo SportelloServizi e una lista di servizi di tipo GenericoServizio non nota a tempo di design. Lo SportelloServizi riceve in input il nome di una operazione e controlla un file di proprietà (vedi caso d'uso Transazione) alla ricerca di una entry per il nome dell'operazione. Se tale operazione è presente allora viene ritornata la lista degli indirizzi da contattare per quella operazione. Gli indirizzi appartengono a servizi di tipo GenericoServizio. Il GenericoServizio aggiorna il suo file di proprietà per il nome utente con l'informazione ricevuta. Il risultato dell'invocazione del servizio con l'operazione di aggiornamento produce un esito (es. 'Utente Mario Rossi ha aggiornato le sue informazioni con via Spade 3').

I passi da effettuare coinvolgono:

- configurazione:
  - come nel caso d'uso Hello
  - configurazione del dominio MinisteroErogatoreSPCoopIT nella porta di dominio OpenSPCoop: registrazione di una porta applicativa per il servizio SPC/Sportello
  - configurazione del dominio MinisteroFruitoreSPCoopIT nella porta di dominio OpenSPCoop: registrazione della porta delegata utilizzata dal soggetto SPC/MinisteroFruitore per invocare il servizio SPC/Sportello.

I servizi di tipo GenericoServizio che verranno invocati non vengono configurati nel Registro Servizi. In questo caso la porta di dominio viene

usato come SOAP Proxy usando la modalità trasparente di integrazione.

- erogazione/fruizione del servizio

  1. come nel caso d'uso Hello.
  2. come nel caso d'uso Hello.
  3. come nel caso d'uso Hello.
  4. riconosciuta la modalità di consegna WSDL si procede all'invocazione del Web Service.
     (a) L'invocazione del servizio provoca la creazione di un processo workflow associato al Web Service Sportello nell'engine JBPM-BPEL.
     (b) La logica del processo è definita come segue:
        - ricevuto il messaggio SOAP si preleva il campo operazione e si prepara il messaggio SOAP in uscita per il servizio SportelloServizi.
        - si invoca il Web Service SportelloServizi che provvede a ritornare la lista delle URL dei servizi, mentre un messaggio di errore ($'NO\_URL'$) identifica la situazione eccezionale di 'nessuna URL per quella operazione'.
        - in base alla risposta del punto precedente, il processo
          * in caso di esito negativo conclude la comunicazione, preparando la risposta per il client e include la precedente risposta nel messaggio per il client.
          * in caso positivo si prelevano le URL. Per ogni URL
            · si preleva la stringa i-esima e si inizializza una variabile EndpointReference.
            · si assegna la variabile al partnerlink per il GenericoServizio

- si memorizza la risposta del servizio
  - al termine del while si prepara una risposta per il client di questo processo, cioè la porta di dominio stessa e l'istanza di processo associata a questo Web Service viene distrutta.

5. come nel caso d'uso Hello.

Capitolo 8

Conclusioni e sviluppi futuri

In questa tesi è stato presentato il processo tramite il quale è stato perseguito l'obiettivo di progettare l'Accordo di Cooperazione secondo le specifiche del CNIPA per OpenSPCoop e la relativa implementazione nel registro Servizi di OpenSPCoop oltre ai casi d'uso che potessero dimostrare le reali potenzialità di questo nuovo elemento.
Passare dalle specifiche del CNIPA alla realizzazione dell'Accordo di Cooperazione in OpenSPCoop ha richiesto uno studio approfondito del Registro Servizi e un principale studio di fattibilità dell'obiettivo prefisso. Conclusa questa prima fase l'attenzione si è concentrata sulla ricerca di un engine per processi produttivi. Sia la nuova tecnologia che la programmazione orientata agli oggetti hanno richiesto uno studio abbastanza approfondito e non sempre facile a causa di una documentazione spesso poco chiara, frammentaria e soprattutto incompleta. La stessa realizzazione dei casi d'uso iniziali è stata per questi motivi parecchio difficoltosa.
Durante questo lavoro di tesi è stata realizzata una documentazione più dettagliata e un tutorial di base messi a disposizione della stessa comunità di sviluppatori in JBPM-BPEL. L'indirizzo di tale documentazione è disponibile in bibliografia [NB]. Lo sviluppo dei casi d'uso è stato reso anche più difficoltoso per la mancanza di strumenti per realizzare le logiche di orche-

strazione in maniera grafica. Le interfacce grafiche che si trovano in rete così come quelle indicate dagli sviluppatori dell'engine stesso presentano lacune che hanno imposto una stesura manuale di tutto il codice di orchestrazione realizzato, prolungandone i tempi necessari. Per rendere quindi utilizzabile in maniera più facile l'Accordo di Cooperazione si suggerisce l'implementazione di una interfaccia grafica che renda possibile realizzare workflow con pochi click. Una feature interessante di questa applicazione è senza dubbio un analizzatore di tipi XML per velocizzare le operazioni che coinvolgono query XPath.

Bibliografia

[AP] Andrea Poli, OpenSPCoop: un'implementazione della Specifica di Cooperazione Applicativa per la Pubblica Amministrazione Italiana,
Tesi di Laurea Specialistica in Tecnologie Informatiche, Università di Pisa, Febbraio 2006.

[BPEL] Business Process Execution Language for Web Services version 1.1
http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[CN1] SPC, 'Sistema pubblico di cooperazione: Architettura, Versione 1.0', CNIPA, 25 Novembre 2004.
http://www.cnipa.gov.it/site/_files/SPCoop-Architettura_v1.0_20041125_.pdf

[CN2] SPC, 'Sistema pubblico di cooperazione: Porta di Dominio, Versione 1.0',CNIPA, 14 Ottobre 2005
http://www.cnipa.gov.it/site/_files/SPCoop-PortaDominio_v1.0_20051014.pdf

[CN3] SPC, 'Specifiche della Busta di e-Government, Edizione 1.0', CNIPA, 21 Aprile 2004

http://www.cnipa.gov.it/site/_files/SPC_Bustae-Gov_v.1_0-21-04-2004.pdf

[CN4] SPC, 'Sistema pubblico di cooperazione: Busta di e-Gov, Versione 1.1',CNIPA, 14 Ottobre 2005 http://www.cnipa.gov.it/site/_files/SPCoop-Busta_e-Gov_v1.1_20051014.pdf

[CN5] SPC, 'Sistema pubblico di cooperazione: Accordo di Servizio, Versione 1.0' CNIPA, 14 Ottobre 2005 http://www.cnipa.gov.it/site/_files/SPCoop-AccordoServizio_v1.0_20051014.pdf

[CN6] SPC, 'Sistema pubblico di cooperazione: Servizi di Registro, Versione 1.0', CNIPA, 14 Ottobre 2005 http://www.cnipa.gov.it/site/_files/SPCoop-ServiziRegistro_v1.0_20051014.pdf

[CNIPA] Sistema Pubblico di Connettività (SPC) http://www.cnipa.gov.it/site/it-it/In_primo_piano/Sistema_Pubblico_di_Connettività_(SPC)

[EJB] Enterprise JavaBeans Technology http://java.sun.com/products/ejb/

[INFOSET] XML Information Set (Second Edition) http://www.w3.org/TR/xml-infoset/

[JBoss] JBoss Application Server http://www.JBoss.org/elqNow/elqRedir.htm?ref=/pdf/JBossASBrochure-Mar2006.pdf

[JBPM] JBoss.com - JBoss JBPM http://labs.JBoss.com/JBossjbpm/docs/index.html

[JBPM-BPEL] JBoss: JBPM BPEL extension 1.1 beta 3
http://docs.JBoss.com/jbpm/bpel/introduction.html

[NB] Developing Netbeans BPEL process with jbpm-bpel
http://www.jboss.com/index.html?module=bb&op=viewtopic&p=4084745

[NS] Namespaces in XML 1.0 (Second Edition)
http://www.w3.org/TR/REC-xml-names/

[OPENSPCOOP] OpenSPCoop - Implementazione Open Source della Specifica SPCoop
http://www.openspcoop.org

[R12] Il progetto CART, Regione Toscana
http://www.rete.toscana.it/etoscana/eprog_cart.php

[R13] Il progetto SOLE, Regione Emilia Romagna
http://www.progetto-sole.it

[RB] Ruggero Barsacchi, Progettazione di un framework Open Source per la
cooperazione applicativa per la Pubblica Amministrazione Italiana, Tesi di Laurea Specialistica in Tecnologie
Informatiche, Università di Pisa, 2005.

[SCHEMA] XML Schema
http://www.w3.org/XML/Schema

[SOAP] SOAP Specifications
http://www.w3.org/TR/soap

[SPC] Sistema Pubblico di Connettività
http://www.cnipa.gov.it/site/_files/5.SPC,Architettura%20SPC,Q,3.0.pdf

[SPCOOP] CNIPA: Servizi di interoperabilità evoluta e cooperazione applicativa
http://www.openspcoop.org/openspcoop/jsp/index.jsp?sel=spcoop

[w3cArch] Web Service Architecture
http://www.w3.org/TR/ws-arch/

[WF] Windows Workflow Foundation
http://msdn2.microsoft.com/en-us/netframework/aa663328.aspx

[XML] Extensible Markup Language (XML)
http://www.w3.org/XML/

[YAWL] YAWL: Yet Another Workflow Language
http://www.yawl-system.com