

ARCHITETTURA TECNICA

Sottosistema di Cooperazione

Indice

1. Premessa.....	3
2. Il modello cooperativo e di interoperabilità.....	5
3. Modello architetturale di riferimento	11
4. Il sistema di gestione del canale di interscambio e cooperazione	14
4.1. Architettura Funzionale.....	14
4.1.1. Porta di Dominio: Funzionalità.....	14
4.1.2. Gestore Eventi: Funzionalità.....	19
4.1.3. Gestore Registro Servizi: Funzionalità	20
4.1.4. Infrastruttura di Gestione: Funzionalità	21
4.2. Architettura Applicativa.....	21
4.2.1. Struttura e Tipologia dei Messaggi	22
4.2.2. Porta di Dominio: Moduli Software.....	26
4.2.3. Gestore Registro Servizi: Moduli Software	32
4.2.4. Gestore Eventi: Moduli Software	33
4.2.5. Infrastruttura di Gestione: Moduli Software	34
4.3. Architettura Esecutiva.....	36
4.3.1. Standard tecnologici utilizzati.....	378
4.3.2. Implementazione Gestore Eventi	43
4.3.3. Implementazione Gestore Registro Servizi.....	44
4.3.4. Piattaforma di Esecuzione.....	47
5. Il modello infrastrutturale della sicurezza.....	50
5.1. Introduzione	50
5.2. Sicurezza dei web services: le problematiche	50
5.3. Sicurezza dei Web Services: gli standard	51
5.4. Architettura Applicativa dell'Infrastruttura di Sicurezza.....	57
5.5. Architettura Esecutiva dell'Infrastruttura di Sicurezza.....	60

1. Premessa

Il Piano d'Azione per l'e-government emanato dal Governo italiano il 23 giugno del 2000 pone un forte accento sulla necessità di *interoperabilità* fra i sistemi informativi delle amministrazioni e degli enti, con l'obiettivo di elevare l'efficienza complessiva dei servizi erogati ai cittadini.

Alcune affermazioni contenute nel Piano d'Azione come

tutte le amministrazioni e gli enti devono essere dotati di un sistema informativo progettato non solo per l'automazione delle funzioni e delle procedure interne della amministrazione e per l'erogazione di servizi ai propri utenti, ma anche per l'erogazione di servizi direttamente ai sistemi informatici delle altre amministrazioni

danno forte enfasi all' *interoperabilità fra sistemi informativi appartenenti ad enti diversi*, il cui scopo è far **cooperare** i diversi sistemi coinvolti nell'espletamento di una richiesta.

Con la denominazione di "Cooperazione Applicativa" si suole identificare il *corpus* di documenti di indirizzo, emessi da diversi enti (AIPA, Centro Tecnico RUPA), che si prefiggono l'obiettivo di standardizzare l'interoperabilità fra i sistemi informativi della Pubblica Amministrazione sia centrale che locale.

Tale corpus documentale si concentra principalmente sugli aspetti organizzativi, funzionali e di architettura logica inerenti l'interoperabilità, lasciando ai diversi enti interessati e agli operatori dell'ICT l'individuazione delle soluzioni tecniche più adatte.

Si è giunti a definire il *modello logico-architeturale della Cooperazione Applicativa*, oggi universalmente accettato, e con esso un insieme di specifiche che l'architettura tecnologica, che intende implementare tale modello, deve soddisfare.

Parallelamente vi è stata un'evoluzione del panorama tecnologico generale, sotto la spinta della diffusione del Web e delle tecnologie Internet, le cui risultanze hanno ed avranno notevoli ricadute sui progetti esecutivi di Cooperazione Applicativa.

La maturità e amplissima diffusione dei protocolli e dei linguaggi nati nell'ambito del Web possono essere considerati come cardini tecnologici di cui tutte le soluzioni applicative con architettura distribuita non possono non tenere conto.

Non meno importanti devono essere considerati gli scenari di cooperazione allargata che si sono venuti a concretizzare nell'ambito della Pubblica Amministrazione. Si suole classificare tali scenari in:

- *Government-to-Citizen (G2C)*: interazione tra cittadino ed amministrazioni, in questo scenario la comunicazione assume la forma di un dialogo interattivo con utenti eterogenei (cittadini, operatori di un dato dominio);
- *Government-to-Government (G2G)*: interazione tra due o più amministrazioni, in questo scenario la comunicazione assume la forma di interscambio di dati tra agenti automatici (applicazioni), operanti all'interno di sistemi informatici con connotati tecnologici e architetture diversi.

-
- *Government-to-Business* (G2B): interazione tra amministrazioni ed imprese, in questo scenario la comunicazione può assumere entrambe le forme precedenti: dialogo interattivo con utenti specializzati facenti parte di un'organizzazione, interscambio di dati fra applicazioni operanti rispettivamente nei sistemi informativi dell'azienda e dell'amministrazione

La “Cooperazione Applicativa”, in senso stretto, è stata studiata per modellare lo scenario G2G. Senza perdere di generalità, si può affermare che il modello definito, con le opportune estensioni, può essere allargato a coprire anche le necessità dello scenario G2B.

2. Il modello cooperativo e di interoperabilità

I documenti redatti sul tema della cooperazione applicativa nell'ambito della Pubblica Amministrazione hanno avuto principalmente l'obiettivo di definire ciò che possiamo denominare il Modello Logico della Cooperazione Applicativa. In particolare, tale modello:

- definisce un insieme di concetti
- assegna i ruoli ai diversi attori
- delimita l'ambito di responsabilità delle singole entità coinvolte
- definisce le modalità attraverso le quali i diversi attori possono cooperare.

Illustriamo di seguito i punti appena elencati.

Il Modello Logico della cooperazione applicativa identifica "l'insieme delle risorse hardware, di comunicazione e software (procedure, dati e servizi) che ricadono sotto la giurisdizione di una determinata organizzazione o ente" con il termine **Dominio**.

Traslando questa definizione dai sistemi informatici all'ambito più strettamente organizzativo, il concetto di Dominio ha una duplice valenza:

1. garantire autonomia alle singole amministrazioni, lasciando inalterato il loro patrimonio informativo interno;
2. marcare il confine di responsabilità amministrativa e di sicurezza di un'organizzazione, in particolare, per quel che riguarda le politiche che definiscono il suo sistema informativo.

Identificate in tal modo le entità coinvolte (Domini), è necessario definire in che modo esse possono cooperare.

Per identificare le modalità di cooperazione è utile osservare che, anche in assenza di un'infrastruttura di cooperazione, ciascun dominio per motivi giuridici (partecipazione all'esecuzione di un procedimento amministrativo normato), per accordi specifici (scambi di informazioni utili a più Amministrazioni), per motivi organizzativi (competenze distribuite tra più amministrazioni) ha un insieme di relazioni già esistenti con altri domini. Per cui, questo insieme di relazioni più o meno formalizzate e di domini interrelati va a costituire una *comunità*. La comunità, spesso, preesiste all'introduzione di un'infrastruttura di cooperazione.

Il contesto appena delineato (comunità) trova sicuro giovamento dall'utilizzo di un'infrastruttura di rete su cui si possono attuare le relazioni appena menzionate, mantenendo ferme le prerogative e l'autonomia di ciascun dominio appartenente alla comunità. In sostanza, l'infrastruttura di cooperazione applicativa non ha come primario scopo la creazione di una comunità, bensì quello di facilitare le interrelazioni fra gli appartenenti ad una comunità e a rendere facile l'adesione alla stessa.

Il Modello Logico della cooperazione applicativa introduce, a questo scopo, uno strato che ha il compito di disaccoppiare le specificità del sistema informativo dei singoli Domini dall'infrastruttura di interscambio sottostante; tale strato è denominato **Porta di Dominio** e ha il duplice compito di:

- Esporre un'interfaccia che permetta di richiamare le funzioni messe a disposizione dal Dominio, senza dover conoscere le specificità implementative delle stesse;
- Mettere a disposizione un insieme di funzioni che permettono ai sistemi facenti parte del Dominio di richiamare servizi esterni rendendo trasparente le specificità dello strato di comunicazione;

Il primo compito è svolto dalla cosiddetta *Porta Applicativa*; i servizi che un sistema informativo di un Dominio mette a disposizione degli altri domini sono accessibili esclusivamente attraverso la Porta Applicativa.

Il secondo compito è svolto dalla cosiddetta *Porta Delegata*; attraverso la Porta Delegata il sistema informativo di un dominio può richiedere servizi e inviare messaggi ad altri domini.

Per completare il quadro è necessario identificare le modalità di interscambio attuabili tra le Porte di Dominio. L'interazione tra Porte di Dominio avviene sempre attraverso lo scambio di messaggi, nel Modello Logico della Cooperazione Applicativa tali modalità costituiscono i cosiddetti *profili di cooperazione*, che sono: **Richiesta di Servizio** e **Notifica Evento**.

Richiesta di Servizio si sostanzia in un *messaggio* inviato da un'applicazione appartenente ad un Dominio (denominato Richiedente) ad un'altra applicazione appartenente ad un altro Dominio (denominato Servente). Il messaggio inviato provoca l'invocazione di un'applicazione sul dominio Servente; al termine dell'esecuzione di questa applicazione viene inviata una risposta al dominio Richiedente. Considerando gli effetti applicativi prodotti, il profilo di collaborazione *Richiesta di Servizio* viene, ulteriormente distinto in:

- *Interrogazione*: una richiesta di servizio che restituisce una informazione del dominio Servente senza modificare lo stato di tale dominio.
- *Transazione*: una richiesta di servizio che produce una variazione permanente dello stato del dominio Servente.

Dal punto di vista della *modalità di comunicazione*, il Modello Logico di Cooperazione classifica la Richiesta di Servizio in:

- *Richiesta di servizio sincrona*: il dominio Richiedente invia al dominio Servente un messaggio e rimane in attesa della risposta;
- *Richiesta di servizio asincrona simmetrica*: il dominio Richiedente invia al dominio Servente un messaggio e rimane in attesa di una risposta che contiene la conferma di ricezione del messaggio. In un tempo differito, il dominio Servente comunicherà la risposta effettivamente correlata alla specifica richiesta di servizio;

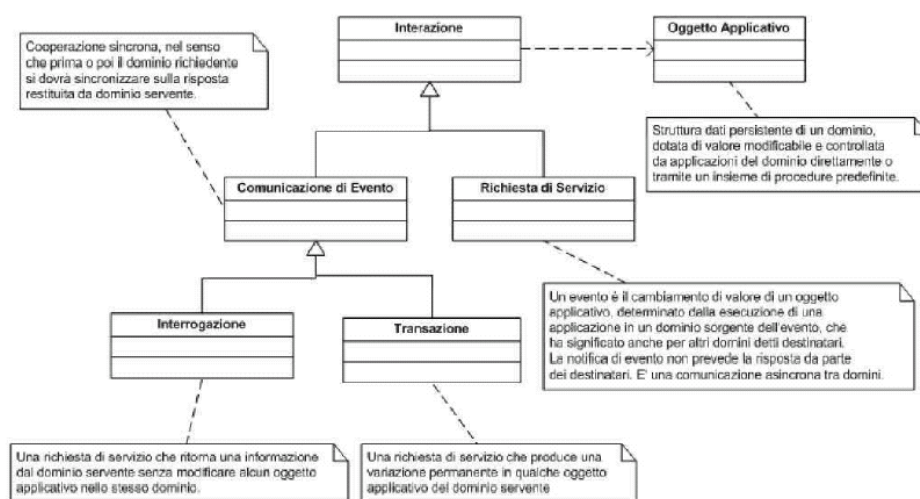
- *Richiesta di servizio asincrona asimmetrica*: il dominio Richiedente invia al dominio Servente un messaggio e rimane in attesa di una risposta che contiene la conferma di ricezione del messaggio. In un tempo differito, il dominio Richiedente contatterà il dominio Servente per ottenere la risposta effettiva del servizio richiesto.

In sintesi, nel caso della richiesta di servizio asincrona simmetrica il “punto di sincronizzazione” è a carico del dominio servente, nel caso di servizio asincrona asimmetrica il “punto di sincronizzazione” è a carico del dominio richiedente.

Infine, le modalità di comunicazione considerate prevedono lo scambio di messaggi in modalità **sincrona** o **asincrona**, per convenzione, il sincronismo viene considerato dal punto di vista della Porta di Dominio presso la quale ha inizio ogni profilo di cooperazione.

Notifica di Evento, prevede che il cambiamento di stato (evento) di un Dominio venga comunicato ad un insieme di domini per i quali tale cambiamento di stato è significativo. Il dominio in cui si verifica l’evento è detto Dominio Sorgente, i domini che sono interessati a ricevere la segnalazione dell’evento sono detti Domini Destinatari.

La discriminante fondamentale tra i due *profili di cooperazione* sopra menzionati consiste nel fatto che nella prima (Richiesta di Servizio) i domini cooperanti dialogano direttamente e hanno contezza reciproca, nel secondo (Notifica Evento) i domini cooperanti non dialogano direttamente e il dominio sorgente dell’evento non ha necessità di conoscere quali siano i domini destinatari dell’evento stesso.



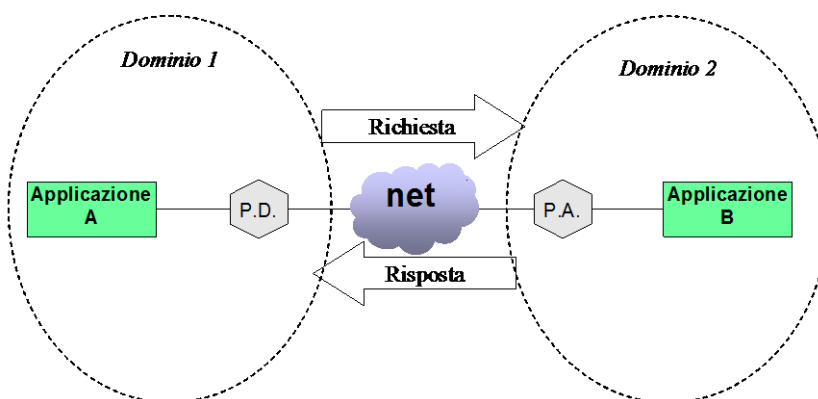
La classificazione dei profili di cooperazione appena illustrata ha razionalizzato l'insieme di possibili comunicazioni potenzialmente attuabili tra i Domini; le caratteristiche dei profili di cooperazione individuati costituiscono la base concettuale da cui si derivano le modalità tecniche di comunicazione necessarie per implementare i suddetti profili.

Le modalità tecniche di comunicazione sono: *Request-Reply*, *Publish&Subscribe* e *Point-to-Point*.

Nella descrizione delle modalità tecniche di comunicazione l'entità concettuale Dominio assume la forma tecnica di *Applicazione*.

La modalità di comunicazione *Request-Reply* prevede che vi siano due attori.

Applicazione Richiedente che invia una richiesta e si mette in attesa della risposta;
Applicazione Servente che riceve la richiesta.

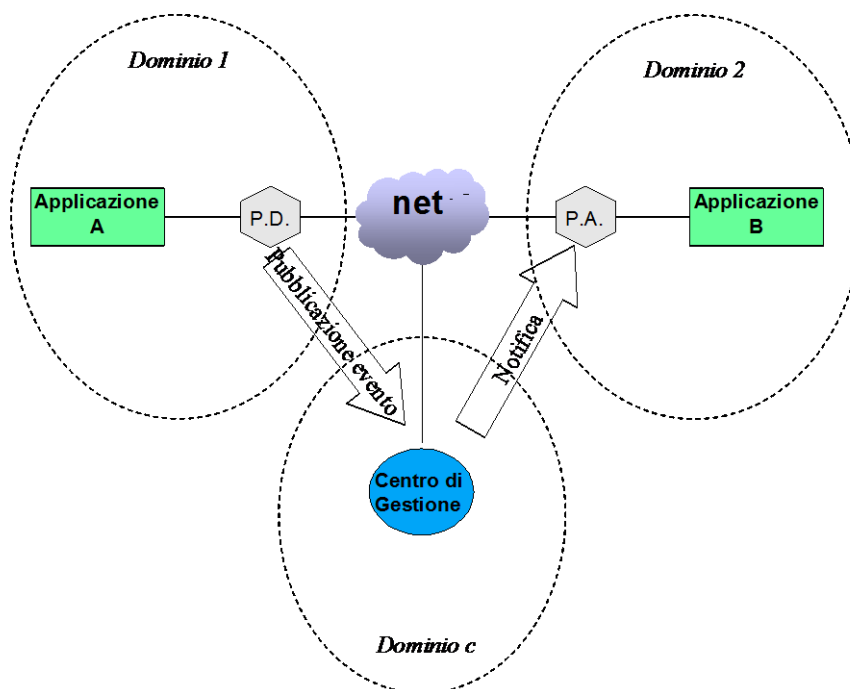


La modalità di comunicazione *Publish & Subscribe* (nel seguito P&S) prevede che vi siano i seguenti attori:

Applicazione Pubblicante: le applicazioni che comunicano la loro intenzione di inviare messaggi, al verificarsi di specifici eventi.

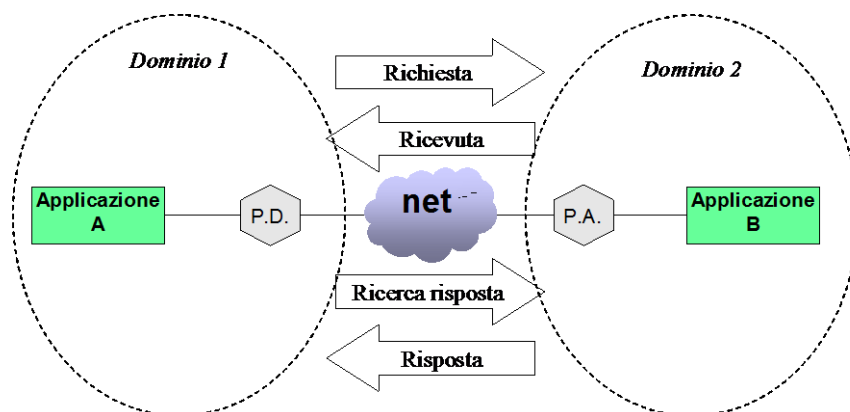
Applicazione Sottoscrittore: le applicazioni che comunicano l'interesse a ricevere la notifica di tutti gli eventi di un certo tipo.

Gestore del Servizio P&S: ha la responsabilità del funzionamento di questa modalità di comunicazione.



La modalità di comunicazione *Point-to-Point* prevede che vi siano i seguenti attori:

- *Applicazione Richiedente* che crea un messaggio e lo deposita in una specifica coda
- *Applicazione Destinatario* che ha accesso alla coda di sua pertinenza e preleva i messaggi in essa contenuti
- *Gestore delle Code* che gestisce tutte le code; esso mantiene i messaggi nelle code fino al momento in cui non vengono prelevati dal destinatario o fino alla scadenza di un timeout. Il Gestore delle Code fa sì che il Richiedente ed il Destinatario non siano direttamente connessi.



Riportando i concetti generali del Modello Logico della Cooperazione sopra illustrati nel contesto di SISN si ha che:

- SISN, dal punto di vista della cooperazione applicativa, può essere scomposto in Domini.

-
- Le applicazioni di ciascun Dominio possono invocare i servizi delle applicazioni di altri Domini utilizzando le interfacce esposte dall'infrastruttura di cooperazione applicativa, le **Porte di Dominio**;
 - Le Porte di Dominio consentono la comunicazione e l'interazione tra le applicazioni di SISN in modo trasparente.

3. Modello architetturale di riferimento

Per definire il modello di riferimento vanno delineate le caratteristiche peculiari del contesto.

L'indicazione principale è che *ciascun ente deve poter scegliere di aderire all'infrastruttura di cooperazione coerentemente con il proprio grado di autonomia e nel rispetto del proprio modello organizzativo e tecnologico; in particolare ciascun ente potrà scegliere liberamente i servizi che intende mettere a disposizione degli altri enti.*

Da ciò deriva che il modello di riferimento deve:

- supportare il decentramento dei servizi;
- garantire alle organizzazioni che lo utilizzano totale indipendenza tecnologica, organizzativa e normativa;

Nello scenario delineato, il Ministero della Salute è visto come il “primus inter pares” e, come tale, rappresenta uno dei nodi logici dell'intero sistema; pur avendo funzionalmente una rilevanza nazionale in quanto organo di governo di massimo livello, da un punto di vista architetturale è uno dei sistemi della rete.

Di conseguenza il modello di riferimento deve permettere che:

- *tutti gli enti cooperanti siano sullo stesso piano*
- *tutti gli enti partecipanti siano aperti alla cooperazione con gli altri.*

Al fine di rispondere ai requisiti esposti, il modello architetturale è ottenuto dalla coniugazione dei modelli: Service Oriented Architecture (SOA) [GARTNER] e Event-Driven Architecture (EDA) [GARTNER]. Entrambi i modelli SOA e EDA si pongono l'obiettivo di definire un design pattern architetturale di riferimento per progettare architetture distribuite, hanno, quindi, diverse caratteristiche in comune, ma non sono intercambiabili. Essi sono studiati per rispondere a requisiti diversi.

Il modello SOA si inserisce nel solco tracciato dai modelli di disegno per architetture distribuite: client/server, object-oriented e component-based e eredita e specializza da questi ultimi diversi concetti.

I principi del modello SOA devono essere ottemperati sia nelle fasi di disegno che di sviluppo e di deployment.

Nella fase di disegno, un servizio è visto come una componente software incapsulata che è costituita da una coppia di elementi definiti separatamente: *service interface* e *service implementation*.

Dalla tassonomia delle componenti appena data discende che, nel modello SOA, *un servizio viene sviluppato per essere richiamato, attraverso un'interfaccia programmatica (service interface), da parte di un'altra componente software (service consumer).*

Sulla scorta di questa affermazione si deriva che la *service interface*, nel modello SOA, ha una importanza fondamentale. Infatti, l'utilizzo del modello SOA si concretizza essenzialmente nella definizione di *service interfaces* e delle relazioni tra le *service interfaces* e ciascun *service consumer*.

La *service interface* definisce il “contratto” di accesso programmatico del servizio. Tale contratto stabilisce l’identità del servizio e le regole di invocazione dello stesso. La *service interface* di ciascun servizio deve contenere le informazioni per rendere il servizio identificabile ed utilizzabile, senza che il *service consumer* sia condizionato dalla implementazione del servizio.

Ottemperare ai principi del modello architetturale SOA nella fase di sviluppo significa:

- implementare le *service implementation* senza fare alcuna assunzione sui *service consumers* che utilizzeranno il servizio specifico
- per ciascuna componente sviluppare prima la *service interface* e, poi, a seguire *service implementation* e i *service consumer*;
- considerare che una *service implementation* può essere invocata in contesti diversi; l’implementatore deve tener conto che l’essenza di un servizio è la sua capacità di essere riusato.

Nell’ambito dei requisiti definiti per l’infrastruttura di cooperazione il modello architetturale SOA soddisfa i seguenti:

- abilita alla cooperazione tra i sistemi senza vincolare l’evoluzione interna degli stessi; garantendo alle organizzazioni che partecipano alla cooperazione totale indipendenza tecnologica, organizzativa e normativa per i loro sistemi
- supporta compiutamente il profilo di cooperazione Richiesta Servizio
- permette l’adesione di un numero infinito di domini, consentendo l’inserimento di nuove entità e l’eliminazione di entità non più aderenti;
- permette l’evoluzione progressiva della rete di cooperazione

Un sistema informativo costruito in accordo al modello SOA risulta essere un’aggregazione di *services* e *services consumers* con accoppiamento lasco. Ottemperare a questo principio nella fase di progettazione, come prima spiegato, significa che i servizi devono essere progettati in maniera *totalmente indipendente* dalle caratteristiche dei potenziali *service consumer*.

Dall’altro lato, il *service consumer* è dipendente dal servizio che richiama, poiché deve contemplare il riferimento alla *service interface* del servizio che intende richiamare, per questo motivo il modello architetturale SOA dà luogo a sistemi informativi ad *accoppiamento lasco* e non *integralmente disaccoppiati*.

Il profilo di cooperazione Notifica Eventi, previsto nel Modello Logico, necessita che i sistemi informativi coinvolti siano *integralmente disaccoppiati*.

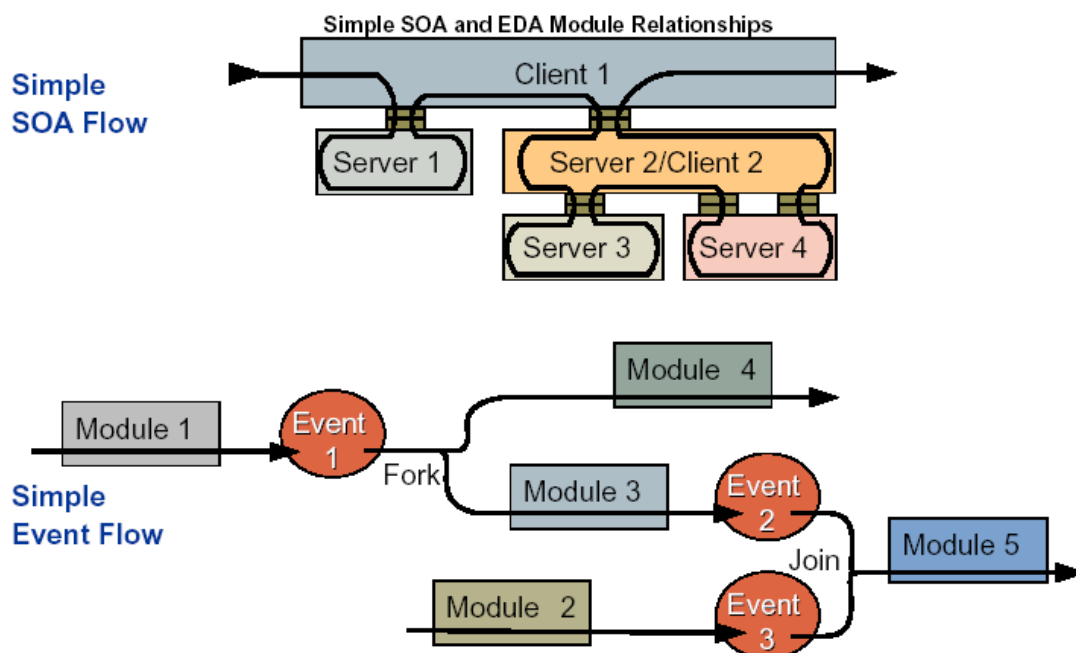
Infatti, il Dominio che emette l’evento potrebbe non conoscere quali sono i domini interessati a ricevere segnalazione dell’evento stesso.

Il modello architetturale Event-Driven Architecture (EDA) è più adatto a soddisfare tale requisito, perché:

- instaura tra le componenti relazioni many-to-many
- le componenti sono o emettitori di eventi ("sources"), o ricevitori di eventi ("sinks") o entrambi; per cui una componente in tale modello non è tecnicamente un servizio, perché non instaura nessuna connessione con un client e non agisce col ruolo di server.
- una componente può pubblicare eventi senza dover conoscere i destinatari di essi e ricevere eventi senza dover conoscere i mittenti. In EDA, quindi, le relazioni tra le componenti si creano dinamicamente a runtime in base alla tipologia e al contenuto dell'evento emesso.

Un sistema informativo conforme al modello EDA è *integralmente disaccoppiato*.

La figura sotto (fonte Gartner Research) schematizza il flusso di controllo che si viene ad attuare nei due modelli sopra illustrati.



Source: Gartner Research (June 2003)

In conclusione il modello architetturale a cui si conforma l'infrastruttura di cooperazione applicativa descritta sarà costituito dalla coniugazione delle caratteristiche dei modelli SOA e EDA.

4. Il sistema di gestione del canale di interscambio e cooperazione

La descrizione della soluzione architeturale coinvolge diversi aspetti:

- *l'aspetto funzionale* che inerisce le funzioni che l'Infrastruttura di Cooperazione Applicativa mette a disposizione e l'organizzazione delle stesse;
- *l'aspetto applicativo* che riguarda i moduli software realizzati
- *l'aspetto tecnologico* che riguarda le caratteristiche delle piattaforme tecnologiche su cui i moduli sono implementati ed eseguiti

4.1. Architettura Funzionale

L'obiettivo di questo capitolo è descrivere le funzioni che l'infrastruttura di cooperazione applicativa fornisce. Si possono identificare quattro macrocomponenti funzionali:

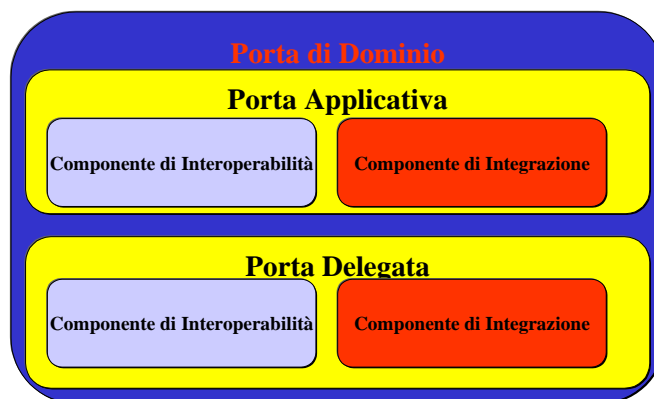
- ***Porta di Dominio***
- ***Gestore Eventi***
- ***Gestore Registro Servizi***
- ***Infrastruttura di Gestione***

4.1.1. Porta di Dominio: Funzionalità

Nel Modello Logico della Cooperazione Applicativa è stato definito il ruolo della Porta di Dominio: essa concentra tutte le funzioni che permettono ai Domini di dialogare tra loro. Per fare ciò la Porta di Dominio deve interfacciare l'infrastruttura che supporta la comunicazione verso i Domini e il sistema informativo interno al singolo dominio di cui fa parte.

Sinteticamente possiamo affermare che: la Porta di Dominio *interopera* con gli altri Domini e *integra* il sistema informativo di un dato Dominio.

Sulla base di queste considerazioni possiamo schematizzare l'architettura funzionale di una Porta di Dominio



Come si evince dalla figura, le funzioni espletate da una Porta di Dominio sono suddivise in due macro-raggruppamenti: *Porta Applicativa* e *Porta Delegata*, nella quali sono previsti i seguenti due strati logici:

- *Componente di Integrazione* che implementa il colloquio con il sistema informativo del Dominio; la componente di integrazione fa da interfaccia tra il mondo esterno ed il sistema informativo del Dominio rendendo trasparenti le peculiarità funzionali, implementative e tecnologiche di quest'ultimo.
- *Componente di Interoperabilità* che implementa le funzioni di comunicazione con gli altri domini. La Componente di Interoperabilità implementa i servizi previsti dalla cooperazione applicativa e dialoga con la Componente di Integrazione per comunicare con il sistema informativo del Dominio.

La suddivisione in strati ha diversi vantaggi:

- marca chiaramente i confini degli strati indipendenti che costituiscono i due macroraggruppamenti funzionali;
- favorisce la suddivisione della progettazione in modo indipendente dei singoli strati;
- riduce la complessità di realizzazione dell'architettura distribuita.

Vediamo, ora, il ruolo che ciascuno strato svolge nell'architettura funzionale.

La Componente di Interoperabilità della Porta di Dominio costituisce l'unico punto attraverso il quale i Domini colloquiano: *ciascun Dominio attraverso la propria Componente di Interoperabilità*

veicola le richieste di servizio verso gli altri Domini cooperanti e, simmetricamente, riceve tutte le richieste di servizio provenienti dagli altri Domini.

Sulla scorta di quanto appena descritto, possiamo, ora, individuare le funzionalità dettagliate che la Componente di Interoperabilità della Porta di Dominio implementa.

Suddividiamo tali funzionalità in due livelli distinti: *Livello Comunicazione e Livello Trasformazione*.

Nel *Livello Comunicazione* sono collocate le funzioni che gestiscono la comunicazione tra i domini. Le funzionalità fondamentali di questo livello sono: *Gestione della Comunicazione* e *Gestione della Sicurezza*.

Gestione della Comunicazione implementa l'intero insieme di funzioni necessario al supporto dei profili di cooperazione definiti dal Modello Logico di Cooperazione Applicativa.

Ne discende che le funzioni facenti parte di *Gestione della comunicazione* sono:

- *pubblicazione di servizio* (service publish): il Modello Logico di Cooperazione Applicativa prevede che ciascun Dominio metta a disposizione i servizi che possono essere richiesti dagli altri Domini, appartenenti alla rete di cooperazione. Affinché ciò sia possibile il Dominio deve poter rendere rintracciabili i servizi da esso messi a disposizione, la funzione di *pubblicazione di servizio* è finalizzata a tale scopo. Attraverso tale funzione il Dominio comunica la disponibilità di un suo servizio, tale comunicazione deve contemplare due principali informazioni: la *collocazione del servizio* e le *modalità attraverso le quali può essere invocato*.
- *rintracciamento di servizio* (service retrieve): i servizi pubblicati dai Domini attraverso la funzione *pubblicazione di servizio*, prima illustrata, devono essere rintracciati dai Domini che intendono fruirne. La funzione *rintracciamento di servizio* ha lo scopo di ricercare la collocazione di un dato servizio e di determinarne le modalità di invocazione.
- *pubblicazione evento* (event publish): il Modello Logico di Cooperazione prevede il profilo di cooperazione Notifica Evento, nel quale viene comunicato alla rete di cooperazione il cambiamento di stato avvenuto all'interno di un Dominio e al quale altri Domini sono interessati. La funzione *pubblicazione evento* consente al Dominio in cui si è verificato il cambiamento di stato di comunicare alla rete di cooperazione tale evento. Il Dominio che pubblica l'evento non ha conoscenza di quali siano i Domini destinatari di tale comunicazione e, quindi, non deve supportare il rintracciamento di tali Domini e l'inoltro ad essi dell'evento.
- *sottoscrizione evento* (event subscribe): gli eventi pubblicati da un Dominio attraverso la funzione *pubblicazione evento* sono destinati a tutti i Domini che hanno interesse a riceverli. La funzione *sottoscrizione evento* al Dominio di dichiarare il proprio interesse a ricevere uno specificato evento. Il Dominio che dichiara l'interesse a ricevere un evento non ha

conoscenza di quali siano i Domini che emettono il dato evento e, quindi, non dialoga direttamente con tali Domini.

- *interfacciamento protocollo di trasporto* (protocol interface): le funzioni descritte in precedenza si occupano di implementare i profili di cooperazione stabiliti dal modello logico, descritto nel capitolo 3. L'esecuzione di tali funzioni richiede la presenza di uno strato comune sottostante alle componenti funzionali viste finora, che comprende i servizi necessari al trasporto dei messaggi tra i Domini. Tali funzioni interfacciano i vari strati del protocollo di comunicazione: le modalità di comunicazione (sincrona, asincrona), l'attivazione delle connessioni, il recapito garantito dei messaggi, il recovery di trasmissioni fallite. Si assume che tali funzioni di base siano messe a disposizione dall'infrastruttura di comunicazione e non sono, quindi, da implementare. La funzione interfacciamento protocollo di trasporto ha il compito di rendere trasparenti le peculiarità di tali funzioni agli altri livelli della Componente di Interoperabilità.
- *Gestione della Sicurezza*: implementa le funzioni che rendono sicura l'interoperabilità tra i Domini.

La sicurezza è un requisito fondamentale per un'infrastruttura di cooperazione applicativa: ciascun Dominio espone servizi che consentono l'accesso al sistema informativo del Dominio stesso, tale sistema informativo ha proprie policies di protezione dei dati e delle funzioni e queste devono essere rispettate anche dai Domini esterni che richiedono i servizi esposti. I sistemi interoperabili, sono sistemi, per loro stessa natura, potenzialmente insicuri: il decentramento sia nell'architettura che nell'amministrazione, l'eterogeneità nelle tecnologie di realizzazione, la distribuzione rappresentano una sfida alla sicurezza generale del sistema e necessitano di opportune strategie di gestione. Gestione della Sicurezza è deputata a tale scopo. In particolare essa implementa le seguenti funzioni:

- *Identificazione ed autenticazione* dei domini che richiedono l'accesso ai servizi esposti dalla propria Porta di Dominio e componenti del sistema;
- *Verifica delle autorizzazioni* all'utilizzo dei servizi richiesti
- *Tracciatura* delle azioni eseguite da ciascun richiedente al fine di poter ricondurre inequivocabilmente ad un richiedente l'esecuzione di una data azione
- *Controllo* delle azioni eseguite per poter prevenire eventuali richieste di servizio potenzialmente portatrici di effetti dannosi (virus, denial-of-services ecc.)
- *Protezione dei dati scambiati* per assicurare la riservatezza delle informazioni.

Le funzioni sopra elencate sono implementate interfacciando l'infrastruttura di sicurezza di ciascun Dominio, in modo da garantire la piena autonomia dei Domini nello stabilire le policies di sicurezza sui servizi di propria pertinenza.

Nel *Livello Trasformazione* in cui sono collocate le funzioni che provvedono al trattamento del messaggio ricevuto o da spedire.

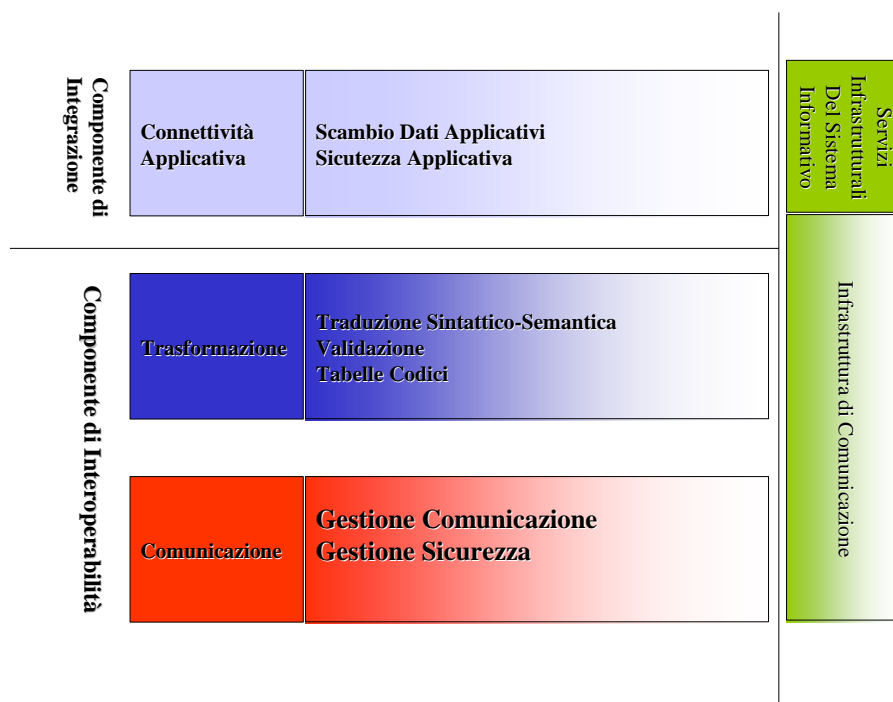
Nel dettaglio le funzionalità fondamentali di questo livello sono:

- *Traduzione sintattica/semantica*: queste funzionalità si occupano di trasformare i dati scambiati sia per quanto riguarda la struttura (data mapping) che per quel che riguarda il contenuto (superamento del semantic mismatch).
- *Validazione*: queste funzionalità verificano la correttezza dei dati scambiati.
- *Tabelle di codici*: queste funzionalità implementano la codifica/decodifica di codici contenuti nei dati scambiati semplificando e velocizzando la traduzione.

Per quanto riguarda la Componente di Integrazione essa implementa il *Livello Connettività Applicativa* in cui sono collocate tutte le funzioni che implementano l'interfaccia verso il sistema informativo del Dominio. In questo livello sono realizzate le funzionalità:

- *Interfaccia Applicativa*: queste funzionalità supportano la comunicazione bidirezionale con la specifica applicazione del Dominio: ricevono dalle applicazioni i dati da trasmettere attraverso la Porta Delegata e inviano alle applicazioni specifiche i dati ricevuti attraverso la Porta Applicativa. Tali funzionalità hanno il compito di rendere trasparente agli altri livelli le peculiarità tecnologiche delle applicazioni del Dominio.
- *Sicurezza Applicativa*: sono le funzionalità di interfaccia con l'infrastruttura di sicurezza: autenticazione, autorizzazione, e altre funzionalità di sicurezza applicativa.

La figura sotto sintetizza quanto sopra descritto.



Sul lato sinistro della figura sono rappresentati due blocchi che rappresentano componenti che la Porta di Dominio interfaccia per eseguire le funzioni di propria pertinenza:

Infrastruttura di Comunicazione che supporta la trasmissione dei messaggi attraverso il protocollo di comunicazione;

Servizi Infrastrutturali del Sistema Informativo che rappresenta la piattaforma di esecuzione su cui opera lo specifico sistema informativo del Dominio e con la quale la Porta di Dominio deve dialogare per invocare le applicazioni del dominio, trasmettere ad essa i dati, ricevere da essa i dati da trasmettere ad altri Domini.

4.1.2. Gestore Eventi: Funzionalità

Il profilo di cooperazione Notifica Evento prevede che il cambiamento di stato verificatosi in un Dominio possa essere notificato a tutti i Domini per i quali tale cambiamento di stato può essere significativo. Inoltre, il Dominio in cui si verifica il cambiamento di stato non può conoscere a priori quali sono i Domini che potenzialmente sono interessati a tale cambiamento di stato.

Per semplificare l'esposizione che segue denominiamo con il termine *evento* il cambiamento di stato all'interno di un Dominio, con l'aggettivo *Sorgente* il Dominio in cui si verifica l'evento a seguito del cambiamento di stato, con l'aggettivo *Destinatario* il Dominio che ha interesse a ricevere l'evento.

Per supportare quanto richiesto dal profilo di cooperazione Notifica Evento è necessaria una componente funzionale dell'architettura che:

- consenta al Dominio Sorgente di comunicare il verificarsi di un evento
- consenta ai Domini Destinatari di comunicare il loro interesse riguardo il verificarsi di una determinata tipologia di eventi

La componente **Gestore degli Eventi** implementa tali funzioni. In particolare, tale componente prevede le seguenti funzioni:

- *pubblicatore eventi*: consente ad un Dominio di pubblicare un evento affinché sia propagato ai Domini interessati a riceverlo. Il Dominio comunica con il Gestore Eventi con la funzione della Porta di Dominio *pubblicazione evento*, descritta nel capitolo precedente
- *sottoscrittore eventi*: consente ad un Dominio di dichiarare il proprio interesse a ricevere un evento fornendo i parametri che lo caratterizzano (tipo di evento, protocollo per la notifica). Il Dominio comunica con il Gestore Eventi della funzione della Porta di Dominio *sottoscrizione evento*, descritta nel capitolo precedente.
- *notificatore evento*: ogni evento pubblicato deve essere notificato a tutti i Domini che al momento della pubblicazione hanno registrato il proprio interesse alla data tipologia di evento.

Oltre alle funzioni appena descritte che caratterizzano il ruolo precipuo del Gestore Eventi sono necessarie altre funzioni di supporto indispensabili al corretto funzionamento della componente:

memorizzazione degli eventi: gli eventi trasmessi dai Domini sorgenti devono essere conservati dal Gestore Eventi fino al momento della notifica ai Domini destinatari, per fare ciò è necessario che il Gestore Eventi sia dotato di un *sistema di memorizzazione permanente* con le relative funzioni di interfacciamento: *memorizzazione eventi* identifica l'insieme di queste componenti.

gestione del tempo limite: diverse funzioni del Gestore Eventi possono avere come parametro un intervallo di tempo:

- il Dominio che emette un evento può indicare un tempo limite entro la quale il Gestore degli Eventi deve consegnare tale evento a tutti i Domini interessati;
- i Domini interessati ad un determinato evento possono condizionare tale interessamento al verificarsi del dato evento in un determinato intervallo di tempo

La funzione *gestione tempo limite* gestisce tutti i parametri legati ad un intervallo di tempo.

Gestore Eventi nell'infrastruttura di Cooperazione applicativa è un *well-known service*, in altri termini un servizio che tutti i Domini devono essere in grado di contattare direttamente senza effettuare operazioni di rintracciamento preliminare.

4.1.3. Gestore Registro Servizi: Funzionalità

Affinché i Domini partecipanti ad una infrastruttura di cooperazione possano effettivamente interoperare è necessario mettere a disposizione:

- una modalità attraverso la quale un Dominio possa dichiarare la disponibilità di un servizio;
- una modalità per verificare l'esistenza di un servizio e come tale servizio possa essere invocato

Il cardine sul quale poggiano le suddette modalità è il *Registro dei Servizi*.

Il Registro dei Servizi mantiene memoria, in modo permanente, dei servizi disponibili nell'infrastruttura di cooperazione, per cui:

- un Dominio dichiara la disponibilità di un nuovo servizio aggiornando il Registro dei Servizi;
- un Dominio che cerca un determinato servizio, interroga il Registro dei Servizi.

La componente funzionale **Gestore Registro Servizi** contempla l'insieme di funzioni necessarie a far colloquiare i Domini con il Registro dei Servizi.

Il Registro dei Servizi deve mantenere un insieme di informazioni che permette di rintracciare ed invocare i servizi, tali informazioni sono:

- tipologia del servizio e sue varianti, ruoli abilitati all'esecuzione.
- collocazione del servizio e modalità di accesso, formato di codifica dei messaggi, parametri di input

Il Gestore Registro Servizi nell'infrastruttura di Cooperazione applicativa è un *well-known service*, in altri termini un servizio che tutti i Domini devono essere in grado di contattare direttamente senza effettuare operazioni di rintracciamento preliminare.

4.1.4. Infrastruttura di Gestione: Funzionalità

Ai raggruppamenti funzionali sopra illustrati vanno aggiunti un insieme di servizi infrastrutturali che devono essere previsti su ciascun Dominio. Tali servizi vanno sotto il nome di Infrastruttura di Gestione e sono:

- *Gestione errori*: implementa l'insieme di funzioni da eseguire nel caso in cui si verifichi un'anomalia nel funzionamento delle componenti della Porta di Dominio.
- *Gestione configurazione*: implementa l'insieme di funzioni che permettono la modifica e lettura dei parametri di configurazione di una Porta di Dominio.
- *Servizio di gestione delle componenti*: implementa l'insieme di funzioni per l'attivazione, la disattivazione delle diverse componenti di una Porta di Dominio
- *Monitoraggio*: implementa le funzioni che permettono il controllo costante del comportamento della Porta di Dominio.
- *Servizio di schedulazione*: tale servizio consente la schedulazione di attività batch.
- *Servizio di logging and tracing*: servizi che effettuano il logging ed il tracing degli eventi e dei messaggi.

4.2. Architettura Applicativa

Nel capitolo precedente, si sono identificati quattro macrocomponenti funzionali (Porta di Dominio, Gestore Eventi, Gestore Registro Servizi, Infrastruttura di Gestione) che l'infrastruttura di cooperazione applicativa deve prevedere; partendo da ciò in questo capitolo si illustrerà l'architettura applicativa di ciascuna delle macrocomponenti, tale architettura identifica i moduli software e le loro interrelazioni.

4.2.1. Struttura e Tipologia dei Messaggi

In termini generali, la Cooperazione Applicativa intende modellare ed implementare in modo standardizzato le interrelazioni esistenti tra entità eterogenee che si esplicano sotto forma di interscambio di dati. L'esistenza (o la potenziale esistenza) di interscambio di informazioni tra diverse entità *costituisce il prerequisito fondamentale all'implementazione di un sistema di Cooperazione Applicativa*.

L'interscambio di informazioni si concretizza nello scambio bidirezionale di messaggi.

Prima di procedere con la descrizione dei moduli software che costituiscono l'infrastruttura di cooperazione applicativa del livello nazionale di SISN è necessario:

- definire la struttura dei dati scambiati che sia condivisa tra tutti i sistemi che devono comunicare. La definizione di tale struttura facilita l'elaborazione dei dati scambiati. Una struttura dati condivisa permette di implementare funzionalità comuni per verificare la correttezza sintattica dei dati scambiati e la loro interpretazione.
- determinare tutte le tipologie di messaggi che possono essere scambiati.

Per definire la struttura comune per i dati scambiati si deve partire dall'analisi dell'interscambio esistente, identificando nello specifico contesto:

- *le entità coinvolte nello scambio;*
- *i dati scambiati e le modalità attraverso le quali viene attuato lo scambio.*

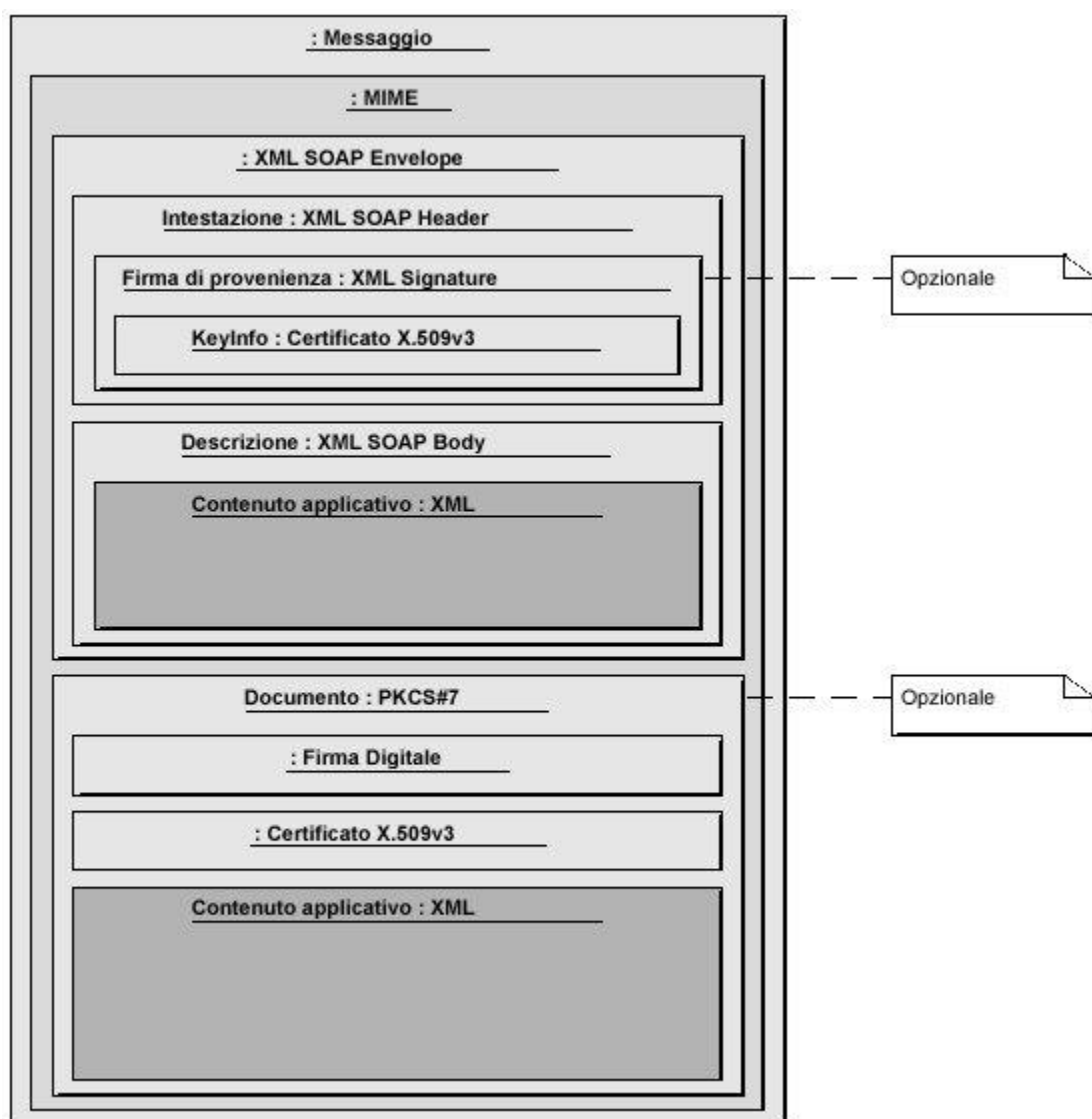
La definizione della struttura va fatta a due livelli:

- *il livello trasmissivo*: la struttura del messaggio trasmesso, ciò riguarda il protocollo di comunicazione utilizzato ed è indipendente dalle caratteristiche dello specifico contesto in cui la cooperazione si attua;
- *il livello contenuto*: la codifica della sintassi e della semantica del contenuto del messaggio trasmesso, tale codifica dipende dalle caratteristiche dello specifico contesto.

Per quanto riguarda il livello trasmissivo, nello specifico contesto, le entità che devono comunicare sono applicazioni informatiche che attraverso le Porte di Dominio si scambiano messaggi strutturati. Di conseguenza, il contenuto dei messaggi deve essere totalmente interpretabile in modo automatico.

Le tecnologie legate allo standard XML si prestano molto bene a questo scopo, in particolare l'utilizzo del protocollo *SOAP* consente di rendere trasparente i diversi meccanismi di trasporto. Basandosi su tale protocollo è stata definita la struttura standard dei messaggi scambiati attraverso un'architettura di cooperazione applicativa tra enti della Pubblica Amministrazione. Tale struttura standard è denominata ***busta di e-government***.

La figura sotto illustra le componenti della busta di e-government.



Il contenuto del messaggio è strutturato in più blocchi distinti di informazione sulla base della specifica standard *SOAP 1.1 with attachments*. Tale standard prevede l'aggregazione in un unico messaggio di più blocchi distinti di informazioni anche eterogenee. Per fare ciò si utilizza il protocollo standard *MIME*.

La struttura più esterna della 'busta' è un messaggio *MIME* concepito in modo tale da risultare indipendente dal contenuto.

La prima parte del messaggio *MIME* contiene un messaggio *SOAP* il cui *envelope* è costituito da:

- una *intestazione (SOAP Header)* contenente le informazioni di indirizzamento e opzionalmente la firma digitale del messaggio in formato *XML Signature*;
- un *corpo del messaggio (SOAP Body)* contenente i dati applicativi del messaggio *SOAP*.

La seconda parte opzionale del messaggio *MIME* contiene un documento in formato *PKCS#7* così composto:

- *firma digitale* del documento applicativo;
- *certificato X.509v3* del mittente con il quale è stato firmato il documento;
- *documento applicativo* in formato *XML* firmato.

La *busta di e-government* rappresenta il contenitore dei dati scambiati e, quindi, definisce la sintassi e la semantica della struttura che trasporta i dati.

La sintassi e la semantica del contenuto della *busta di e-government* è strettamente correlata al dominio applicativo in cui si opera poiché modella concetti dello stesso.

Pertanto, la definizione della sintassi e semantica a *livello contenuto* richiede l'analisi dettagliata delle tipologie di dati scambiati in ciascun ambito di interscambio tra gli enti coinvolti nello specifico dominio applicativo.

Definita la struttura del messaggio di trasporto, è possibile, ora, elencare le tipologie di messaggi che possono essere scambiati tra i Domini attraverso l'infrastruttura di cooperazione applicativa.

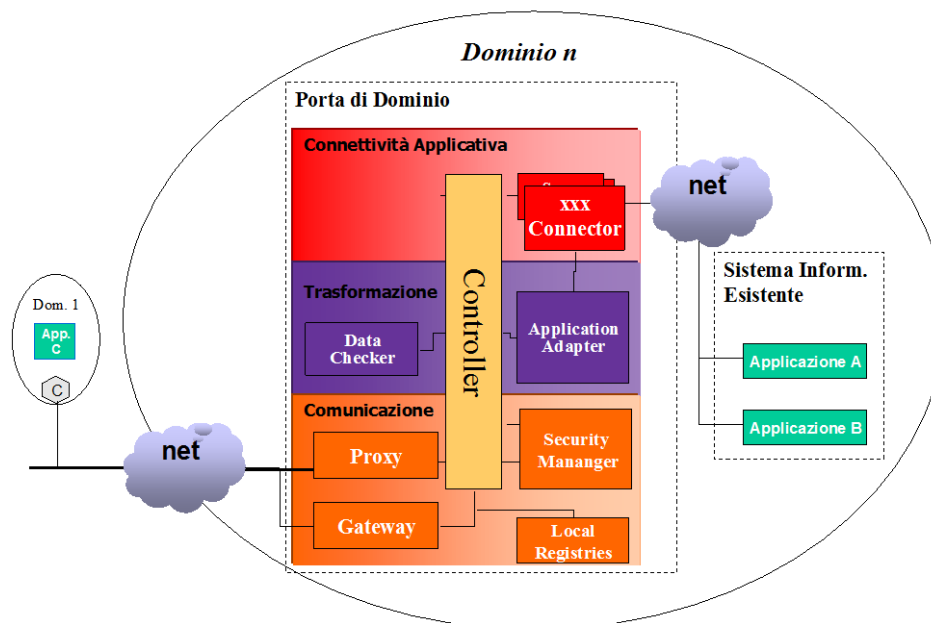
Tali tipologie sono:

- **Messaggio di pubblicazione evento** con il quale un Dominio comunica al Gestore Eventi un evento da pubblicare.
- **Messaggio di sottoscrizione evento** con il quale un Dominio comunica al Gestore Eventi il suo interesse a ricevere la notifica del verificarsi di un dato evento.
- **Messaggio di pubblicazione di servizio** con il quale un Dominio richiede al Gestore Registro Servizi di memorizzare nel registro la disponibilità di un nuovo servizio.

-
- **Messaggio di ricerca di servizio** con il quale un Dominio richiede al Gestore Registro Servizi la collocazione e le modalità di invocazione di un determinato servizio.
 - **Messaggio di sincronizzazione registro** con il quale viene richiesto ai Domini di aggiornare i propri registri locali dei servizi.
 - **Messaggio di avvenuto evento** con il quale si notifica a tutti i domini che hanno dichiarato interesse verso un dato evento che tale evento si è verificato.
 - **Messaggio di richiesta di servizio** con il quale un Dominio richiede ad un altro Dominio l'invocazione di un dato servizio. Questo messaggio può essere inviato solo dopo che la collocazione del servizio è stata determinata attraverso un messaggio di ricerca di servizio.

4.2.2. Porta di Dominio: Moduli Software

La figura data sotto illustra lo schema a blocchi dei moduli software della Porta di Dominio e le loro interrelazioni.



Per ciascuno dei moduli rappresentati nello schema dato sopra viene descritto il ruolo e la forma scelta per l'implementazione.

Proxy è il modulo software che si occupa dei messaggi in ingresso al Dominio. I messaggi sono strutturati sotto forma di busta di e-government, il Proxy si occupa dell'estrazione dei dati dalla busta di e-government dopo la ricezione del messaggio. Il modulo Proxy riceve le seguenti tipologie di messaggi:

- Messaggio di sincronizzazione registro;
- Messaggio di avvenuto evento;
- Messaggio di richiesta di servizio.

Il modulo Proxy è implementato sotto forma di Web Service che riceve messaggi SOAP di tipo Document.

Gateway è il modulo software che si occupa di inoltrare i messaggi in uscita dal Dominio. I messaggi sono strutturati sotto forma di busta di e-government, il Gateway si occupa di costruire la busta di e-government prima della spedizione del messaggio. Il modulo Gateway inoltra le seguenti tipologie di messaggi:

- Messaggio di pubblicazione evento;
- Messaggio di sottoscrizione evento;
- Messaggio di pubblicazione di servizio;
- Messaggio di ricerca di servizio;
- Messaggio di richiesta di servizio

Il modulo Gateway è implementato sotto forma di Web Service Client che invia messaggi SOAP di tipo Document.

Controller il processo di trattamento di un messaggio da inviare o ricevuto è costituito da un insieme di passi che devono essere eseguiti in maniera controllata e coordinata, ciascun passo è a carico di uno dei moduli che costituiscono la Porta di Dominio. Controller coordina l'esecuzione per processo di trattamento invocando i moduli corretti, gestendo la transazionalità delle operazioni, mantenendo lo stato del processo.

Data Checker implementa il controllo sintattico e semantico dei dati da inviare e ricevuti. La validazione si basa su un insieme di regole che può essere dinamicamente modificato: le regole possono riguardare la presenza o meno di un dato oppure la validità, in termini di intervalli di un determinato dato ecc..

La validazione preliminare all'invio rende più efficiente l'infrastruttura di cooperazione in quanto evita che siano inviate delle richieste non corrette ai Domini.

La validazione dei dati ricevuti è necessaria perché non tutti i Domini prevedono la verifica di correttezza dei dati prima dell'inoltro, pertanto i dati ricevuti potrebbero essere non corretti sintatticamente e/o semanticamente e, in tal caso, non devono essere inoltrati alle applicazioni del Dominio.

Il Data Checker fa uso di un *Dizionario dei Metadati* e di *Tabelle di Codifica* nelle attività di verifica. Si veda dis eguito la descrizione della componente Local Registries per i dettagli.

Il Data Checker è implementato sotto forma di un insieme di classi Java che riceve in input un XML Document e ne verifica la correttezza in funzioni di un insieme di regole mantenute in un repository apposito. Nel caso di dati ricevuti l'XML Document viene estratto dal Proxy dalla busta di e-government, nel caso di dati da inviare l'XML Document è preparato dal modulo Application Adapter.

Application Adapter implementa la trasformazione bidirezionale dei dati. Application Adapter trasforma i dati da inviare dal formato dell'applicazione mittente in un XML Document in accordo ad un DTD predefinito e i dati ricevuti dal formato XML nel formato riconosciuto dall'applicazione destinataria. Application Adapter deve effettuare:

- trasformazioni isomorfe: la sintassi e la semantica del costrutto da tradurre è simile sia nel linguaggio di partenza che nel linguaggio di arrivo per cui deve essere effettuata soltanto una trasformazione lessicografica
- trasformazioni non isomorfe: la semantica nel linguaggio di partenza del costrutto da tradurre non ha una corrispondenza diretta nel linguaggio di arrivo (semantic mismatch), per cui Application Adapter deve effettuare delle trasformazioni complesse sui dati, utilizzando anche informazioni esterne al costrutto che contiene i dati di partenza.

Application Adapter è implementato sotto forma di un insieme di classi Java ed opera in due direzioni:

- da uno specifico Connector (si veda dopo) riceve uno stream di byte strutturato in conformità alla sintassi utilizzata dall'applicazione che ha emesso i dati stessi. Application Adapter, in accordo alle regole di trasformazione predefinite, opera le trasformazioni conseguenti per ottenere un XML Document.
- da Data Checker riceve un XML Document e, in accordo alle regole di trasformazione predefinite, opera le trasformazioni conseguenti per ottenere uno stream di byte strutturato in modo tale che sia recepibile dall'applicazione destinataria.

Le regole di trasformazione sono configurabili dinamicamente e sono mantenute in un repository apposito. Application Adapter fa uso dei Local Registries, in particolare del Dizionario dei Metadati, per effettuare le trasformazioni.

Local Registries implementa la gestione dei databases locali necessari alle attività della Porta di Dominio. Tali databases sono: *Registro Locale dei Servizi*, *Dizionario dei Metadati* e *Tabelle di Codifica*.

Il *Registro Locale dei Servizi* mantiene tutte le informazioni sui servizi esportati dallo specifico Dominio e tutte le informazioni per il rintracciamento dei servizi offerti da altri Domini che sono di interesse dello specifico dominio.

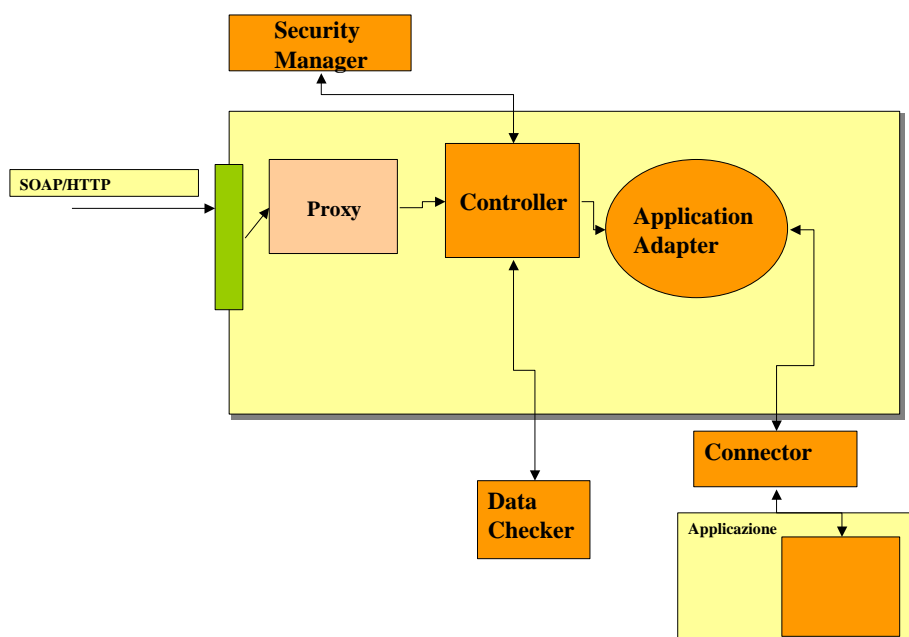
Il Dizionario dei Metadati contiene le regole che definiscono la sintassi e la semantica statica dei costrutti del linguaggio utilizzato per lo scambio dei dati. La sintassi del linguaggio di scambio definisce i costrutti che strutturano il *livello contenuto* del messaggio trasportato dalla busta di e-government, la semantica statica definisce il significato associato a tali costrutti. In considerazione di ciò, il Dizionario dei Metadati manterrà regole lessicografiche e grammaticali che permettono al Data Checker:

- al Data Checker di verificare che il costrutto utilizzato esista e sia scritto correttamente (controllo lessicografico) e di verificare che il costrutto sia utilizzato in un modo sintatticamente corretto (controllo sintattico);
- all'Application Adapter di effettuare la corretta traduzione fra il linguaggio di strutturazione dei dati utilizzato dalla specifica applicazione interna e il linguaggio di scambio dati, utilizzato nell'infrastruttura di cooperazione.

Le *Tabelle di Codifica* hanno l'obiettivo di facilitare la gestione di dati codificati che hanno bassa dinamicità.

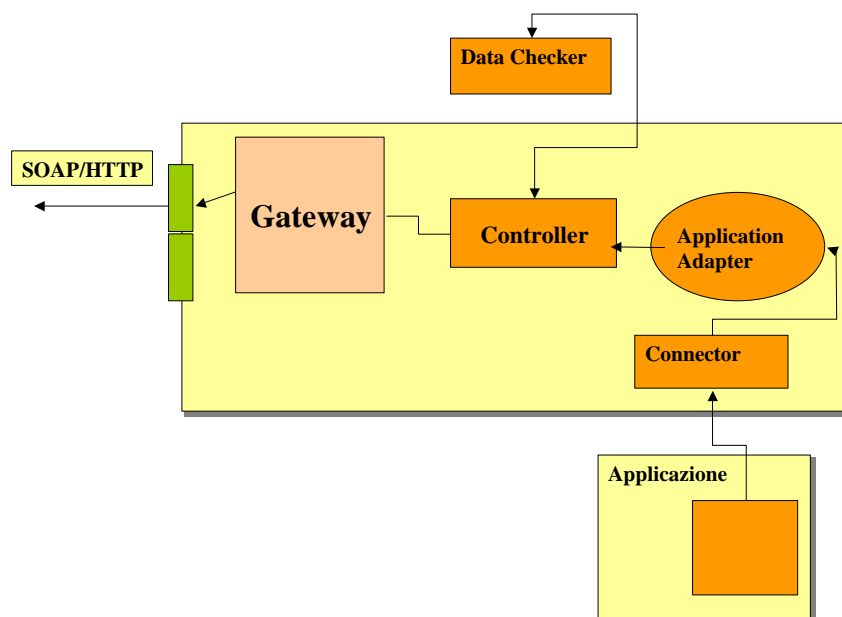
Connectors implementano le modalità tecniche di dialogo con le applicazioni del Dominio; le applicazioni del Dominio possono essere tecnologicamente eterogenee, tale eterogeneità si estrinseca in diverse modalità tecniche di comunicazione (protocolli di comunicazione, modalità di invocazione, strutturazione dei parametri, policies di autenticazione ecc.) che devono essere gestite. A tale fine nella Porta di Dominio possono essere diversi Connector, ciascuno specializzato per interfacciare un particolare protocollo o una particolare applicazione. L'implementazione dei Connector dipende dalle caratteristiche tecniche delle applicazioni del sistema informativo che devono dialogare con la Porta di Dominio.

La figura che segue schematizza le interrelazioni tra i moduli software sopradescritti a fronte della ricezione di un messaggio.



Il messaggio SOAP/http viene ricevuto dal Proxy, che verifica le credenziali che accompagnano il messaggio attraverso il Security Manager. Se il controllo ha esito positivo, viene estratto il contenuto dalla busta di e-government e viene passato al Controller che attiva il processo di trattamento. A questo punto il Controller attiva il Data Checker per la verifica di correttezza dei dati ricevuti. Dopo la verifica, il Controller attiva l'Application Adapter dell'applicazione specifica a cui è indirizzato il messaggio. L'Application Adapter richiama, poi, la funzione dell'applicazione attraverso lo specifico Connector.

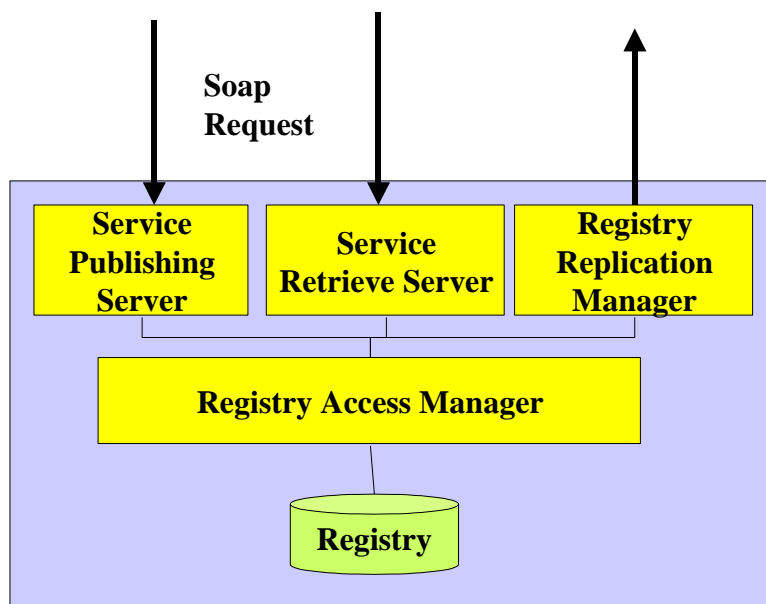
La figura che segue schematizza le interrelazioni tra i moduli software sopradescritti a fronte dell'invio di un messaggio.



Un messaggio originato da un'applicazione del Dominio viene consegnato allo specifico Connector che lo passa all'Application Adapter che procede alla traduzione in XML del messaggio. Il messaggio così strutturato viene passato al Controller che attiva il Data Checker per verificarne la correttezza e poi il Gateway che costruisce la busta di e-government e la inoltra al destinatario.

4.2.3. Gestore Registro Servizi: Moduli Software

La figura data sotto illustra lo schema a blocchi dei moduli software del Gestore Registro Servizi



Per ciascuno dei moduli rappresentati nello schema dato sopra viene descritto il ruolo e la forma scelta per l'implementazione.

Service Publishing Server è il modulo software che riceve i messaggi di pubblicazione di servizio con il quale un Dominio richiede al Gestore Registro Servizi di memorizzare nel registro la disponibilità di un nuovo servizio. Il modulo è implementato sotto forma di Web Service che riceve messaggi SOAP.

Service Retrieve Server è il modulo software che riceve i messaggi di ricerca di servizio con il quale un Dominio richiede al Gestore Registro Servizi la collocazione e le modalità di invocazione di un determinato servizio. Il modulo è implementato sotto forma di Web Service che riceve messaggi SOAP e riceve un XML Document in formato WSDL (si veda il capitolo 5.3 per i dettagli di questo formato)

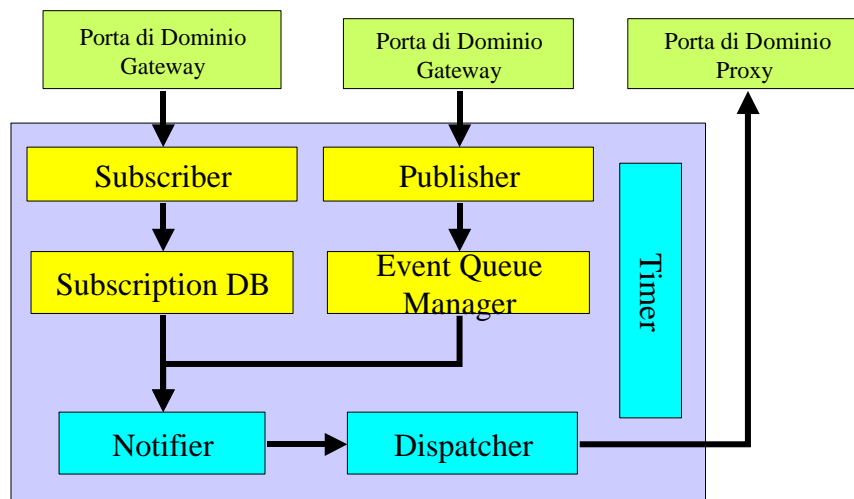
Registry Replication Manager è il modulo software che gestisce l'invio dei messaggi di sincronizzazione dei registri con il quale viene richiesto ai Domini di aggiornare i propri registri locali dei servizi.

Registry Access Manager è l'interfaccia programmatica che gli altri moduli devono chiamare per accedere al Registry

Registry è il database in cui viene mantenuto l'elenco dei servizi.

4.2.4. Gestore Eventi: Moduli Software

La figura data sotto illustra lo schema a blocchi dei moduli software del Gestore Eventi.



Per ciascuno dei moduli rappresentati nello schema dato sopra viene descritto il ruolo e la forma scelta per l'implementazione.

Subscriber riceve i *messaggi di sottoscrizione evento* con i quali un Dominio comunica al Gestore Eventi il suo interesse a ricevere la notifica del verificarsi di un dato evento. Il modulo Subscriber memorizza nel **Subscription DB** le sottoscrizioni ricevute. Il modulo è implementato sotto forma di Web Service e riceve messaggi SOAP, utilizza un DBMS per memorizzare le sottoscrizioni.

Publisher riceve i *messaggi di avvenuto evento* con il quale un Dominio comunica al Gestore Eventi che si è verificato un determinato evento. Gli eventi ricevuti vengono passati al modulo

Event Queue Manager (vedi sotto). Il modulo è implementato sotto forma di Web Service e riceve messaggi SOAP.

Event Queue Manager gestisce la memorizzazione permanente degli eventi ricevuti. Il modulo utilizza il meccanismo delle code per gestire la memorizzazione e il prelevamento degli eventi da notificare. Tali azioni sono eseguite in accordo al paradigma FIFO (First-in First-out). Il modulo è implementato sotto forma di classi Java che interfacciano un repository (coda) dove vengono mantenuti gli eventi.

Notifier legge gli eventi presenti nella coda gestita da Event Queue Manager e verifica, per ciascun nuovo evento rilevato, i sottoscrittori a cui deve essere inviato ciascun evento consultando il database dei sottoscrittori. Gli eventi estratti dalla coda con i rispettivi destinatari vengono passati al modulo Dispatcher. Il modulo è implementato sotto forma di classi Java.

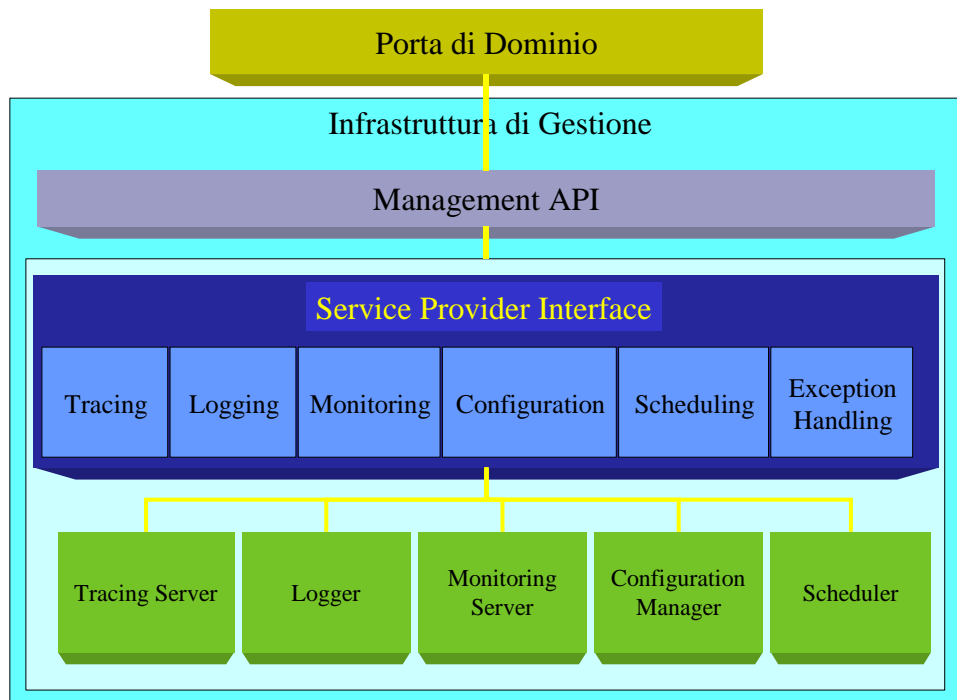
Dispatcher invia l'evento rilevato da Notifier a ciascun sottoscrittore richiamando le funzioni esposte a tale scopo dalle corrispondenti Porte di Dominio. Il modulo è implementato sotto forma di Web Service Client che invia messaggi SOAP.

Timer gestisce le scadenze temporali a cui può essere condizionata la notifica di specifici eventi. Il modulo è implementato sotto forma di *daemon* che verifica continuamente le diverse scadenze temporali definite.

4.2.5. Infrastruttura di Gestione: Moduli Software

La Porta di Dominio, il Gestore Registro Servizi e il Gestore Eventi sono le componenti architetture espressamente deputate all'esecuzione di funzioni proprie dell'infrastruttura di Cooperazione Applicativa. L'insieme di tali componenti costituisce un'architettura complessa che, come tale, deve essere amministrata, configurata e controllata. Nell'ambito dell'architettura funzionale si è individuato l'insieme di funzioni deputato a questo fine, raggruppate nel macrocomponente denominato Infrastruttura di Gestione.

La figura data sotto illustra lo schema a blocchi dei moduli software della componente Infrastruttura di Gestione.



Lo schema a blocchi descritto nella figura è suddiviso in tre strati:

- *Management API* che implementano il colloquio tra le funzioni di gestione e gli altri moduli della Porta di Dominio.
- *Service Provider Interface* (SPI) che espongono i servizi dei singoli sistemi che implementano le diverse funzioni relative alla gestione.
- *Providers* che sono prodotti/soluzioni che implementano le funzioni necessarie al supporto della gestione.

La comunicazione tra moduli software e l'Infrastruttura di Gestione è bidirezionale: i moduli software della Porta di Dominio comunicano informazioni all'Infrastruttura di Gestione e quest'ultima invia messaggi ai moduli software per cambiare il loro stato.

Le principali Management API che implementano la comunicazione tra i moduli software e l'Infrastruttura di Gestione sono:

- Registrare le applicazioni, con tale API un modulo software comunica alla infrastruttura di gestione la sua esistenza;
- Ricevere messaggi di errore, con tale API un modulo software comunica alla infrastruttura di gestione il verificarsi di un errore

-
- Ricevere messaggi di tracing, con tale API i moduli software possono comunicare all'infrastruttura di gestione le informazioni necessarie a tracciare le loro attività
 - Ricevere messaggi di log, con tale API i moduli software possono comunicare all'infrastruttura di gestione informazioni di stato che consentiranno di recuperare eventuali stati inconsistenti.

I principali messaggi che attraverso le Management API vengono inviati dalla Infrastruttura di Gestione ai moduli software della Porta di Dominio sono:

- attivazione e disattivazione di un modulo
- abilitazione di un livello di tracing
- abilitazione di un livello di logging
- impostazione di specifici attributi di configurazione del modulo
- segnalazione di eccezioni di sistema

Nello schema a blocchi dato nella figura sono illustrate anche le tipologie di Providers che si ritiene necessari nell'architettura completa, essi sono:

- Tracing Server
- Logger
- Monitoring Server
- Configuration Manager
- Scheduler

4.3. Architettura Esecutiva

I moduli software individuati nei capitoli precedenti risiedono su sistemi elaborativi in grado di eseguirli, per *sistema elaborativo* s'intende l'insieme di tecnologie software di base e hardware.

L'insieme costituito da sistema elaborativo e moduli software su di esso collocati viene denominato **Nodo Applicativo** dell'Infrastruttura di Cooperazione.

L'Infrastruttura di Cooperazione Applicativa dal punto di vista dell'Architettura Esecutiva risulta essere **una federazione di Nodi Applicativi** dove ciascun nodo ospita tutti i moduli software necessari ad esportare i servizi di uno specifico Dominio; ad un Nodo Applicativo corrisponde uno ed un solo Dominio.

L'Architettura Esecutiva si occupa di descrivere le caratteristiche e la struttura dei Nodi Applicativi, in termini di tecnologie di base e di distribuzione dei moduli software.

4.3.1. Standard tecnologici utilizzati

Di seguito saranno illustrati gli standard tecnologici che sono stati scelti come riferimento per l'implementazione dell'Architettura Esecutiva.

Tali standard sono: **Web Services e J2EE**.

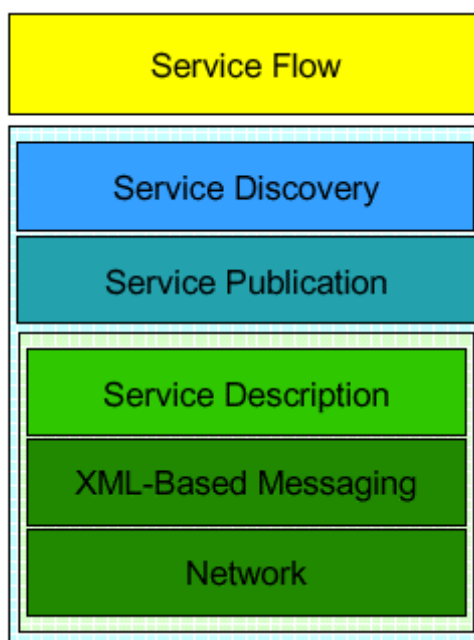
Web Services: Caratteristiche e utilizzo nell'Infrastruttura di Cooperazione Applicativa

I **Web Services** costituiscono le fondamenta tecnologiche su cui è implementata l'Infrastruttura di Cooperazione Applicativa.

Sinteticamente si può affermare che *un Web Service è un'interfaccia che descrive un insieme di operazioni che sono rese accessibili via rete, attraverso l'utilizzo di messaggi strutturati in un linguaggio standard (XML).*

Un Web Service viene descritto mediante una struttura dati standardizzata denominata *service description* che contiene tutti i dettagli necessari a interagire con il servizio di cui il Web Service è interfaccia: il formato dei messaggi ricevibili, il protocollo di trasporto, la locazione del servizio. L'interfaccia nasconde i dettagli implementativi del servizio, permettendone l'utilizzo, indipendentemente dalla piattaforma hardware e software sulla quale il servizio stesso è implementato.

La figura sotto mostra lo stack concettuale delle componenti che fanno parte dell'implementazione di un'architettura basata sui Web Services.



Il layer più basso dello stack è *Network*, questo perché tutte le operazioni che vengono eseguite con i Web Services devono essere effettuate attraverso una rete:

- la pubblicazione di un servizio e il rintracciamento della sua service description vengono realizzate mediante una comunicazione tra sistemi connessi in rete.
- l'invocazione del servizio viene realizzata mediante una comunicazione tra sistemi connessi in rete.

Esistono diversi protocolli di rete pubblici che possono essere utilizzati per implementare il layer Network dello stack dei Web Services, tra questi SMTP (Simple Mail Transfer Protocol), generalmente utilizzato per il trasferimento della posta elettronica, e FTP (File Transfer Protocol), utilizzato per il trasferimento di file; in ambito Internet, però, il protocollo più diffuso è HTTP/HTTPS, e per l'ambito SISN, si utilizza tale protocollo.

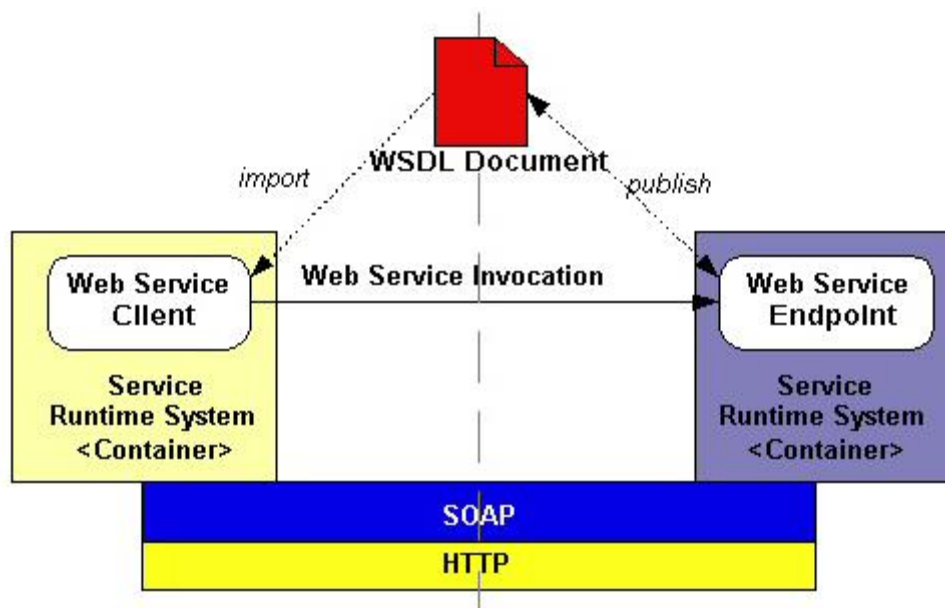
Il layer successivo, denominato *XML-Based Messaging*, stabilisce che i messaggi, scambiati tra i sistemi appartenenti ad un'architettura distribuita basata sui Web Services, devono essere strutturati in accordo al linguaggio standard XML. L'utilizzo di questo standard permette l'interoperabilità delle applicazioni, poiché i dati in un documento XML hanno una rappresentazione in formato testo e sono indipendenti dalla specifica piattaforma di elaborazione che li genera.

Nell'ambito delle architetture distribuite, storicamente, sono attuabili due paradigmi di invocazione:

- Remote Procedure Call (RPC)
- Messaging

Entrambi i paradigmi di invocazione presentano vantaggi e svantaggi, e la scelta di uno dei due dipende dalle caratteristiche del contesto operativo in cui la comunicazione tra i sistemi si deve collocare. Per questo motivo, entrambi i paradigmi di invocazione possono essere implementati utilizzando il protocollo SOAP: l'implementazione del paradigma RPC mediante SOAP prende il nome di *Web Services RPC-style*, l'implementazione del paradigma Messaging mediante SOAP prende il nome di *Web Services Document-style*.

Per quanto riguarda i Web Services RPC-style lo schema di implementazione è dato nella figura sotto:



Nella figura si può notare che sono presenti due stubs: Service Runtime System che si occupano di trasformare i dati da scambiare (marshaling e unmarshaling), di gestire la chiamata al Web Service Endpoint, di gestire la restituzione del risultato al Web Service Client. Tutto ciò è basato sulla *service description* contenuta nel file WSDL, si veda dopo per i dettagli sulla strutturazione del contenuto di questo file.

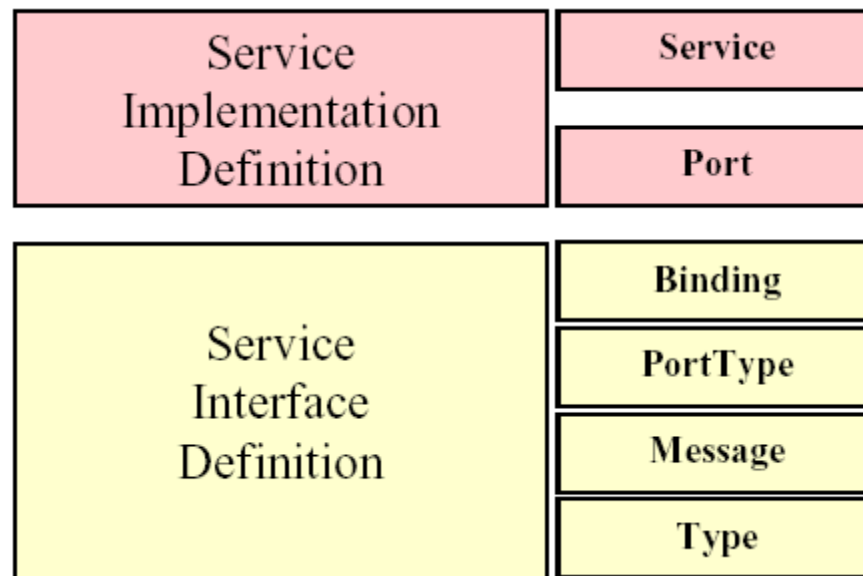
Per quanto riguarda Web Services Document-style si ha che:

- il contenuto del messaggio SOAP (body) è un *XML document* strutturato in accordo ad una sintassi preconcordanata;
- l'*XML document* è l'unico parametro passato all'invocazione via SOAP di un Web Service. La risposta all'invocazione è diversa a seconda della *semantica della comunicazione*.

Il layer successivo dello stack dei Web Services, denominato *Service Description*, stabilisce che le caratteristiche dei servizi invocabili sono descritte utilizzando un linguaggio XML, denominato **WSDL**.

Utilizzando WSDL si descrivono in un XML document (*service descriptor*), in maniera formale, l'interfaccia e tutti i meccanismi mediante i quali è possibile interagire con il servizio.

In definitiva, con WSDL si definisce l'interfaccia e le modalità di invocazione di un servizio. Nella figura sotto viene rappresentata la struttura di WSDL.



WSDL contiene due parti: *service interface* e *service implementation*.

Service Interface contiene:

PortType: definisce la *signature* del Web Service, i messaggi XML di input e di output.

Message: specifica quali XML data types costituiscono le varie parti del messaggio

Types: definisce eventuali tipi dati complessi

Binding: descrive il protocollo, il formato dei dati per una particolare service interface (WSDL: portType).

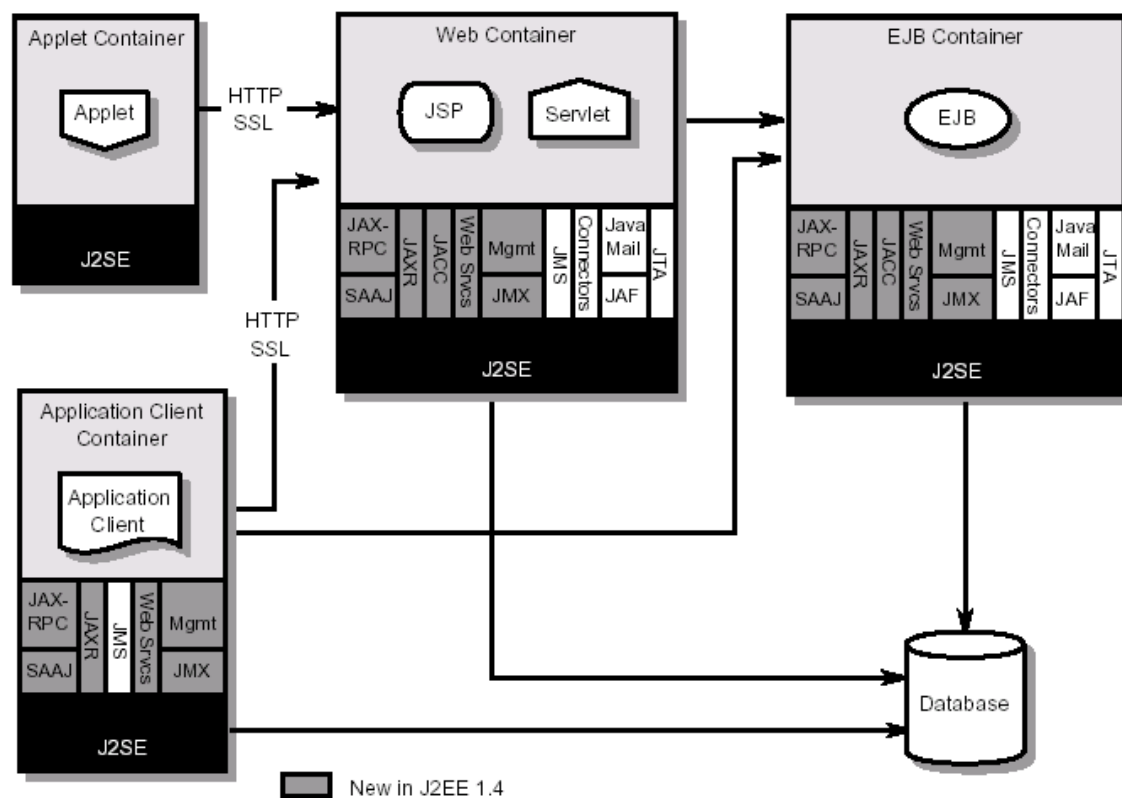
Da tutta la descrizione data sopra deriva che i Web Services hanno caratteristiche tali da soddisfare la maggior parte dei requisiti architetturali e tecnologici posti dall'infrastruttura di cooperazione applicativa:

- separazione fra l'interfaccia di un servizio e la sua implementazione;
- protocollo di comunicazione indipendente dal trasporto;
- accoppiamento lasco tra le componenti che comunicano via SOAP;
- possibilità di implementare le modalità tecniche di comunicazione: Request-Reply, Point-To-Point;
- possibilità di scegliere tra due paradigmi di invocazione (RPC e Messaging)
- possibilità di implementare l'approccio misto per quel che riguarda la pubblicazione e il rintracciamento dei servizi.

J2EE: Caratteristiche e utilizzo nell'Infrastruttura di Cooperazione Applicativa

Java 2 Platform, Enterprise Edition (J2EE) definisce un insieme di standard e mette a disposizione un insieme completo di servizi standardizzati che semplificano la realizzazione di applicazioni in Java.

La figura data sotto schematizza tutte le componenti che costituiscono lo standard J2EE.



In essa sono mostrate le relazioni logiche tra gli elementi che costituiscono la piattaforma J2EE standard, tali relazioni non implicano un partizionamento fisico degli stessi su diversi sistemi elaborativi.

I Containers, denotati da rettangoli separati, sono elementi che a runtime forniscono servizi alle componenti applicative. Ad esempio Application Client Container mette a disposizione il servizio Java Message Service (JMS) attraverso una serie di APIs standard a tutte le applicazioni eseguite nel dato container. Allo stesso modo, tutte le componenti applicative eseguite nei diversi container possono utilizzare, attraverso API standard, tutti i servizi che lo specifico container offre.

Gli application server conformi allo standard J2EE devono implementare tutti i servizi standardizzati definiti in esso. Ciò implica che in qualsiasi application server J2EE-compliant si ritrovano gli stessi servizi con le stesse interfacce.

L'Infrastruttura di Cooperazione Applicativa del livello nazionale di SISN, oggetto di questo studio, è implementata su piattaforma J2EE-compliant.

Tra i servizi standardizzati e le tecnologie disponibili sulle piattaforme J2EE, concentriamo l'attenzione su alcune che riteniamo molto utili per l'implementazione dell'Infrastruttura di Cooperazione Applicativa del livello nazionale di SISN.

Java API for XML Messaging (JAXM) è un Java Community Process project (JSR 67) [J2EE] che definisce una API per supportare lo scambio di XML document. JAXM utilizza SOAP 1.1 with Attachments per interfacciare il protocollo di trasporto. JAXM non si preoccupa del trasporto del messaggio, ma della composizione, invio, ricezione e decomposizione dei messaggi. In sostanza, JAXM facilita l'implementazione di un sistema di scambio messaggi basati su XML document.

La busta di e-government definisce la struttura di un XML document da trasportare mediante il protocollo SOAP 1.1 with Attachments. ***Per tale motivo lo standard JAXM è adatto ad essere utilizzato per supportare la composizione e decomposizione della busta di e-government e si ipotizza di utilizzarlo nell'implementazione dei moduli software Gateway e Proxy della Porta di Dominio.***

Java Message Service (JMS) [J2EE] è la specifica di una API che interfaccia sistemi di messaging. JMS semplifica la realizzazione di applicazioni software che hanno bisogno di scambiare dati ed eventi in modalità asincrona.

Fornisce supporto al messaging in modalità sia message queueing che P&S. Diversi middleware di messaging espongono l'interfaccia JMS, molti application server conformi allo standard J2EE contemplano la *reference implementation* di JMS. ***Lo standard JMS è utilizzato nell'ambito dell'Infrastruttura di Cooperazione Applicativa per implementare le modalità tecniche di comunicazione asincrone (Point-to-Point e P&S).***

Per quanto riguarda l'implementazione del modulo *Infrastruttura di Gestione* si fa uso dello standard **JMX** [J2EE] che è implementato negli application server J2EE più diffusi, ciò ci consente di avere a disposizione molte funzionalità necessarie all'infrastruttura già implementate.

Java Management Extensions (JMX) è una tecnologia standard, che fa parte della famiglia J2EE, per il supporto al *management* e *monitoring* che può essere utilizzata per costruire infrastrutture di gestione di sistemi implementati in Java.

JMX è stata progettata per rispondere ai requisiti posti dalle nuove architetture distribuite, in particolare l'elevata dinamicità di tali architetture.

Infatti, JMX non definisce una nuova semantica per la gestione delle risorse, ma specifica un'infrastruttura di gestione nella quale componenti che si autodescrivono possono essere aggiunte on-the-fly all'architettura senza impattare sull'infrastruttura di gestione.

Il management delle risorse con l'utilizzo di JMX è resa totalmente indipendente dall'infrastruttura tecnologica di gestione soggiacente.

4.3.2. Implementazione Gestore Eventi

Nell'ambito dell'Architettura Funzionale è stata individuata la macrocomponente funzionale, Gestore Eventi, deputata a gestire il profilo di cooperazione Notifica Evento, previsto dal Modello Logico di Cooperazione Applicativa. L'architettura applicativa del Gestore Eventi prevede i seguenti moduli software: Subscriber, Publisher, Event Queue Manager, Notifier, Dispatcher, Timer.

L'architettura funzionale del Gestore Eventi può essere implementata secondo i seguenti due schemi:

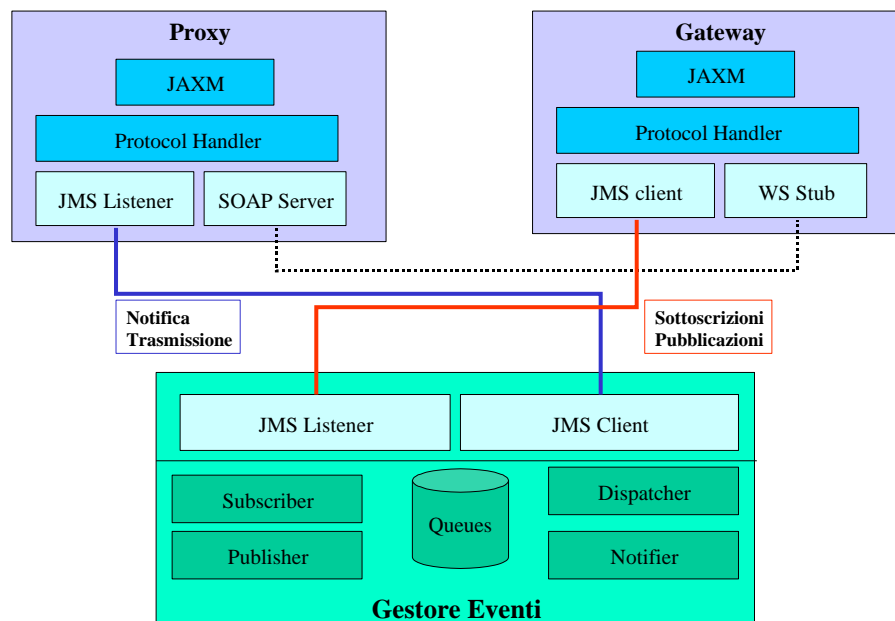
- *store&disptatch*: modalità di propagazione degli eventi, in cui un evento viene inviato ad un *sistema specializzato*, su cui risiede il modulo software Gestore Eventi, che memorizza l'evento (store) e poi lo distribuisce (dispatch) a tutti i destinatari interessati;
- *broadcast*: modalità di propagazione che prevede l'invio dell'evento a tutti i sistemi raggiungibili e, poi, lascia a questi ultimi determinare se l'evento è di proprio interesse.

La modalità store&dispatch richiede la presenza di un sistema specializzato su cui sia collocato il Gestore Eventi. La modalità broadcast non richiede sistemi specializzati, ma è molto meno efficiente della modalità store&dispatch, perché tutti i sistemi ricevono tutti gli eventi e devono analizzarli prima di scartarli.

Per tale ragione è stata implementata la modalità store&dispatch. L'attuazione della modalità store&dispatch richiede il supporto alla modalità tecnica di comunicazione P&S.

Il protocollo di trasporto HTTP non supporta nativamente la modalità tecnica P&S, per cui è stata individuata la soluzione *JMS-based*, che sopperisce a questa carenza del protocollo di trasporto.

Nella soluzione JMS-based si ipotizza di utilizzare l'interfaccia standard JMS, facente parte dello standard J2EE, descritto nella sezione successiva, appoggiandosi alla reference implementaion di tale interfaccia disponibile in diversi application server J2EE-compliant. JMS è una API che interfaccia sistemi di messaging, supportando sia la modalità tecnica PTP che P&S. Sulla base di ciò si può ipotizzare si utilizzare JMS in accordo a quanto schematizzato nella figura sotto.



Nella figura sono evidenziati i moduli Proxy e Gateway che fanno parte della Porta di Dominio. Nella soluzione JMS-based nei moduli Gateway e Proxy è introdotta la componente *Protocol Handler* che ha in carico la gestione dell'istradamento della *busta di e-government*. In particolare, nell'ambito del Gateway, *Protocol Handler*, in base al tipo di destinazione ed al tipo di servizio, provvede ad instradare la busta di e-government verso un *Web Service Stub* per la trasmissione in formato SOAP/http, nel caso si tratti di Richiesta di Servizio, oppure verso il JMS Client, nel caso si tratti di pubblicazione di un evento (modalità P&S).

Nell'ambito del Proxy, *Protocol Handler* riceve un messaggio JMS se si tratta di una Notifica Evento oppure una request SOAP se si tratta di una Richiesta di Servizio. *Protocol Handler* estrae da queste ultime la busta di e-governement.

Come si vede, *Protocol Handler* gestisce le diverse modalità di ricezione e trasmissione e le rende trasparenti agli altri componenti della Porta di Dominio, per fare ciò fa uso della tecnologia standard JAXM (si veda la sezione successiva per i dettagli) che consente di trattare generici messaggi SOAP, la cui trasmissione e ricezione può avvenire attraverso sistemi diversi.

La soluzione JMS-based prevede, quindi, che:

- tutte le comunicazioni sono basate su JMS
- i moduli Publisher, Notifier, Dispatcher, Subscriber vanno implementati sfruttando quanto la reference implementation di JMS mette a disposizione.

4.3.3. Implementazione Gestore Registro Servizi

In generale, la pubblicazione di un servizio è una funzione:

-
- intrinsecamente asincrona;
 - la cui frequenza d'esecuzione è notevolmente inferiore a quella di tutte le altre tipologie di funzioni previste dall'infrastruttura di cooperazione applicativa (richiesta di servizio, pubblicazione e ricezione evento); è ragionevole immaginare che l'attività preponderante delle Porte di Dominio sia dedicata a pubblicare eventi, a richiedere ed eseguire servizi. Se ciò non fosse vero starebbe ad indicare che l'infrastruttura di cooperazione pubblica tanti servizi che vengono, però, scarsamente utilizzati, per cui l'infrastruttura di cooperazione applicativa perderebbe di significato
 - non strettamente transazionale: un servizio esiste nell'ambito dell'infrastruttura di cooperazione applicativa se e solo se è stato pubblicato; il fatto che un Dominio possieda un servizio che, potenzialmente, interessa ad altri Domini, ma non ancora pubblicato non può essere considerato uno stato inconsistente dell'infrastruttura di cooperazione.

Le caratteristiche che discriminano l'insieme di soluzioni possibili derivano da come si risponde alle seguenti domande:

- in che modo pubblicare un servizio
- in che modo mantenere memoria della disponibilità di un servizio
- in che modo rintracciare un servizio disponibile

Gli approcci attuabili sono *completamente distribuito*, *completamente centralizzato* oppure un *ibrido* dei due.

Completamente Distribuito: il Dominio che pubblica il servizio invia un messaggio in broadcast a tutti i Domini attestati sulla rete con la *service description*, contenuta nel file WSDL, del servizio pubblicato. Ciascun Dominio mantiene un Registro Locale in cui memorizza il file WSDL ricevuto. Il rintracciamento del servizio avviene consultando il Registro Locale. La cancellazione di un servizio pubblicato avviene attraverso un messaggio opportunamente strutturato inviato in broadcast a tutti i Domini attestati sulla rete.

Questo approccio è notevolmente efficiente per quanto riguarda il rintracciamento dei servizi pubblicati, poiché si basa sulla consultazione di un registro collocato localmente a ciascun Dominio. L'invio in broadcast dei messaggi di pubblicazione servizio non si ritiene appesantisca il traffico di rete in maniera significativa, dato che, come esposto nella premessa a questo capitolo, tali messaggi sono di numero considerevolmente inferiore a tutte le altre tipologie di messaggi che viaggiano nell'infrastruttura.

Questo approccio ha come controindicazione il fatto che non vi è certezza che tutti i Domini ricevano il messaggio di pubblicazione: tutti i Domini che si connettono all'infrastruttura successivamente all'invio del messaggio di pubblicazione evento non lo riceveranno mai. Per ovviare a questo problema si può attuare la tecnica denominata *cascading discovery*: il Dominio che entra chiede con un messaggio in broadcast l'aggiornamento del proprio registro locale, le risposte a questo messaggio contengono la replica dei registri locali dei Domini che rispondono e una marca temporale che identifica l'ultimo aggiornamento a cui è allineato lo specifico registro.

Completamente Centralizzato: prevede che esista un registro centrale, collocato su un sistema univocamente identificato nell'ambito dell'infrastruttura di cooperazione, nel quale vengono memorizzate tutte le *service description* dei servizi pubblicati dai Domini. Nel momento in cui un Dominio intende richiedere un determinato servizio, interroga il registro centrale e determina le modalità di invocazione da applicare per richiamare il dato servizio. La cancellazione di un servizio pubblicato avviene inviando un messaggio opportunamente strutturato al gestore del registro centrale.

Questo approccio richiede l'individuazione di un sistema che viene elevato a gestore del registro dei servizi e che, in virtù di tale ruolo, diviene, dal punto di vista logico, un single point of failure, dell'intera infrastruttura di cooperazione. Per ovviare a ciò è necessario che il sistema elevato a tale rango abbia un'infrastruttura tecnologica altamente affidabile e ne venga garantita la continuità operativa a tutti i livelli: hardware, software di base e applicativo.

Ibrido: prevede l'esistenza di uno o più *rendezvous point*, che sono entità speciali all'interno dell'infrastruttura di cooperazione applicativa, che mantengono registri in cui vengono memorizzate tutte le *service description* dei servizi pubblicati dai Domini. I rendezvous point si occupano, poi, di aggiornare, attraverso l'invio di messaggi di notifica, i registri locali dei singoli Domini. Nel momento in cui un Dominio intende richiedere un determinato servizio, interroga innanzitutto il proprio registro locale e nel caso in cui il servizio richiesto non fosse in esso presente interroga il registro del rendezvous point.

Questo approccio consente il funzionamento dell'infrastruttura di cooperazione anche nel caso di temporanea indisponibilità del rendezvous point, mitigando così i requisiti di affidabilità e continuità operativa di quest'ultimo. Inoltre, questo approccio ha un'efficienza paragonabile all'approccio Completamente Distribuito perché il rintracciamento di un servizio, nella maggior parte dei casi, si risolve attraverso un'interrogazione del registro locale del Dominio richiedente.

L'approccio Ibrido è il più adeguato ai requisiti dell'infrastruttura di cooperazione applicativa oggetto di questa trattazione poiché in questo modo:

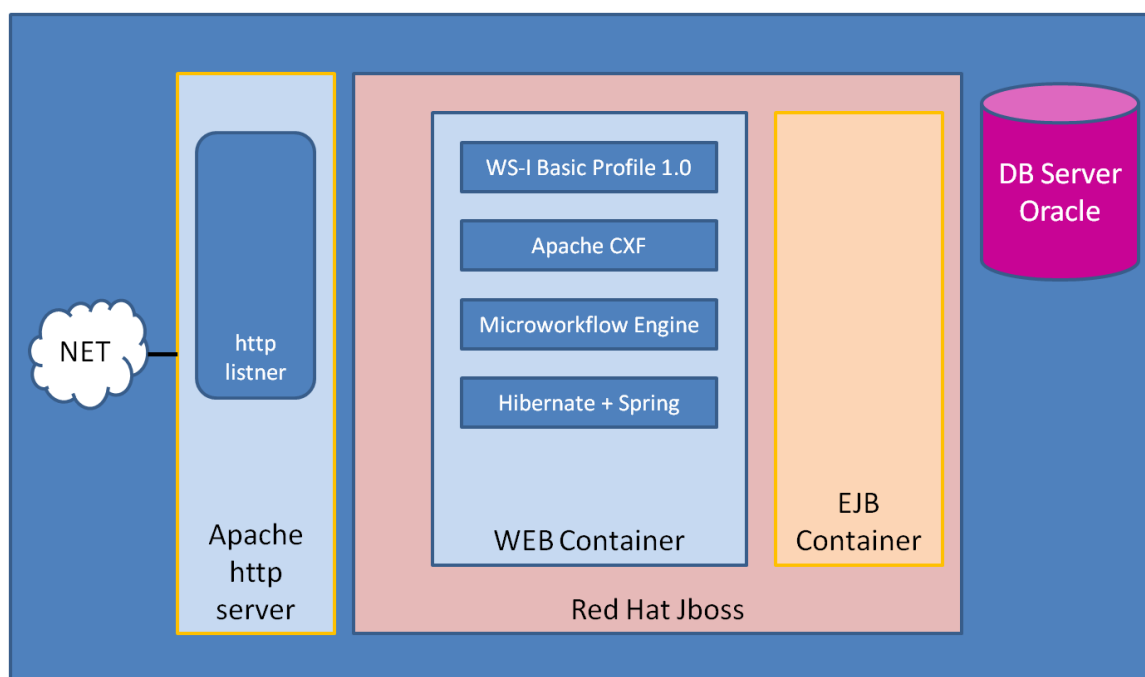
- il livello nazionale di SISN può funzionare in modo autonomo, utilizzando un proprio registro locale che verrà sincronizzato con gli altri attraverso un protocollo concordato con gli enti partecipanti all'infrastruttura di cooperazione applicativa e utilizzando, eventualmente, dei rendezvous point da stabilire
- il livello nazionale di SISN può adeguarsi all'evoluzione della cooperazione applicativa in ambito nazionale senza impatti onerosi su quanto già realizzato, utilizzando eventuali registri centrali messi a disposizione, ad esempio, da futuri Centri di Gestione della Cooperazione Applicativa in ambito nazionale (AgID: ex-DigitPA ex-CNIPA)

4.3.4. Piattaforma di Esecuzione

I criteri che sono stati alla base della selezione delle tecnologie adottate derivano:

- dalle specifiche funzionali
- dai requisiti non funzionali, correlati al carico elaborativo che l'Infrastruttura di Cooperazione Applicativa deve essere in grado di soddisfare, e dai requisiti architetturali, correlati alle caratteristiche organizzative dello specifico contesto. Tali requisiti sono:
 - Flessibilità: poter gestire le evoluzioni organizzative, di processo e normative che accompagneranno il sistema nel corso degli anni, per gestire le interazioni con soggetti istituzionali nazionali ed internazionali in continuo cambiamento e per consentire ad ogni dominio di utilizzare la tecnologia ritenuta più opportuna.
 - Scalabilità: la crescita potenziale del carico elaborativo a cui l'Infrastruttura di Cooperazione Applicativa deve far fronte non è interamente quantificabile a priori, per cui deve essere possibile estendere l'infrastruttura elaborativa gradualmente al crescere della mole di dati e di transazioni

- Modularità ed estendibilità: per permettere a ogni ente di decidere, in modo svincolato dalla tecnologia, quali componenti utilizzare dell'infrastruttura a seconda delle proprie necessità.
 - Semplicità di gestione: per permettere l'installazione ed il controllo delle componenti dell'infrastruttura in modo semplice.
- dagli standard de jure e de facto nell'ambito dell'ICT e trend del mercato delle tecnologie, in particolare:
- Web services (SOAP, WSDL, UDDI)
 - XML
 - J2EE
 - i nuovi protocolli: WS-I Basic Profile 1.0, SAML
 - PKI



La piattaforma evidenziata in figura fa un ampio utilizzo di tecnologie Open Source.

Ciò viene incontro alle indicazioni espresse nel rapporto “*Indagine conoscitiva sul software a codice sorgente aperto nella Pubblica Amministrazione*” emesso dal Ministero dell’Innovazione Tecnologica [OS] e, nello stesso tempo, favorisce la diffusione del sistema, in quanto tutti gli enti interessati possono usufruirne senza vincoli nell’ambito delle indicazioni sulla riusabilità illustrate nel rapporto suddetto.

Le componenti descritte di seguito costituiscono l’Architettura Esecutiva della **Porta di Dominio del livello nazionale di NSIS**.

Web server è responsabile di accettare le richieste http. Come web server si utilizza *Apache http Server*.

Application server si occupa di gestire la parte applicativa di un sistema e di integrarsi in modo diretto con le risorse esterne (database, broker, etc.). L'Application server è distinto in due componenti: Web Container e EJB Container

Web container si occupa di gestire le richieste che provengono dal web server. Si utilizza come web container *Red Hat Jboss*.

Application container si occupa di elaborare le richieste ed esegue la logica applicativa che implementa le funzioni dell'infrastruttura di cooperazione. Si utilizza *Red Hat Jboss* come application container.

Framework applicativo un insieme di templates, patterns e servizi preconfezionati per costruire rapidamente applicazioni J2EE.

Web Services Container si occupa di gestire le chiamate sotto forma di web services, si prevede di utilizzare a questo scopo *Apache CXF*. *Apache CXF* è una implementazione open-source delle specifiche SOAP version 1.1 e SOAP Messages with Attachments. CXF è il successore di Apache SOAP.

Database server si occupa di memorizzare i dati ed i messaggi. Si utilizza il DBMS Oracle RAC 12c.

5. Il modello infrastrutturale della sicurezza

5.1. Introduzione

L'Infrastruttura di Cooperazione Applicativa è finalizzata a supportare lo scambio di dati fra sistemi eterogenei attestati su una rete di comunicazione e a consentire l'accesso ai servizi messi a disposizione dai singoli sistemi a tutti gli utenti interessati a fruirne. Da questa descrizione deriva immediatamente che l'Infrastruttura di Cooperazione Applicativa deve garantire che l'accesso ai servizi sia protetto da utilizzi fraudolenti e che i dati scambiati non siano alterati durante la trasmissione.

A tale fine l'Infrastruttura di Cooperazione Applicativa deve essere corredata di un sistema che ne garantisca la *sicurezza logica*. In termini generali, la sicurezza logica è l'insieme di *policies* adottate all'interno di un sistema informatico per garantire l'utilizzo controllato delle risorse che il sistema manipola.

Identificazione e Autenticazione, è il servizio che determina in modo inequivocabile l'agente (processo o utente) che sta richiedendo l'accesso ad una risorsa e garantisce al sistema destinatario della richiesta l'identità del richiedente.

Autorizzazione, il servizio che stabilisce a quali risorse del sistema (dati o applicazioni) l'agente autenticato può accedere.

Confidenzialità, il servizio garantisce che una data informazione possa essere letta soltanto dal destinatario

Integrità il servizio che garantisce di poter verificare che il contenuto di una richiesta non è stato alterato dal momento in cui è stato emesso dal richiedente

Non Ripudio, il servizio che garantisce che l'esecutore di un'azione, sia esso mittente che destinatario, non possa negare di averla eseguita

L'elenco dei servizi appena dato costituisce l'architettura funzionale dell'infrastruttura di sicurezza implementata per garantire la sicurezza logica dell'Infrastruttura di Cooperazione Applicativa.

L'infrastruttura di sicurezza è una componente cardine dell'intera infrastruttura di cooperazione, per cui è necessario analizzare in dettaglio tutte le problematiche da affrontare per la sua corretta implementazione.

5.2. Sicurezza dei web services: le problematiche

Nell'ambito delle architetture basate su Web sono state già definiti standard ed implementate infrastrutture software che rispondono ai requisiti di sicurezza. Di seguito è presentato l'approfondimento delle problematiche legate alla sicurezza dei web services.

Per garantire confidenzialità ed integrità si è utilizzato il protocollo standard TLS1.2 che supera le vulnerabilità del suo predecessore SSL v3. Tale protocollo opera a livello di trasporto e protegge le sessioni di comunicazione stabilite fra due entità distribuite sulla rete.

Per supportare i processi di autenticazione e autorizzazione, si utilizzano sistemi che identificano gli utenti attraverso le credenziali presentate e che associano all'utente autenticato un insieme di abilitazioni che sono state precedentemente stabilite; le credenziali e le autorizzazioni associate ai singoli utenti vengono mantenute in repository che devono far parte del sistema informativo che governa l'accesso alle risorse.

L'Infrastruttura di Cooperazione Applicativa si basa sulla tecnologia dei Web Services. Le caratteristiche intrinseche di tale tecnologia rendono insufficienti gli standard e le infrastrutture software utilizzate per garantire la sicurezza logica delle applicazioni web.

In particolare:

- le applicazioni web sono *connection oriented*, cioè la comunicazione avviene attraverso una connessione diretta tra sistema richiedente e sistema servente; ciò permette di implementare le protezioni di sicurezza a livello di connessione;
- i web services supportano comunicazioni message oriented, in cui non vi è la garanzia dell'instaurazione di una connessione diretta tra sistema richiedente e sistema servente, così gli approcci applicabili ai sistemi connection-oriented per rispondere ai requisiti di sicurezza sono inapplicabili o insufficienti per rendere sicura un'architettura basata su Web Services. In particolare, il protocollo TLS 1.2:
 - non può essere utilizzato per proteggere trasmissioni al di fuori del contesto di una particolare sessione di trasporto;
 - non può essere utilizzato per determinare l'identità dei partecipanti al di fuori di una specifica sessione di trasporto
 - l'autenticazione via TLS perde di valore al termine di una sessione
 - non supporta il non ripudio.

Il processo di autenticazione basato sulle credenziali scambiate all'inizio della sessione non può essere applicato nelle architetture basate sui Web Services perché ogni scambio di dati avviene su una connessione diversa che non ha memoria dei precedenti stati della comunicazione.

5.3. Sicurezza dei Web Services: gli standard

Nell'ambito delle architetture basate su web services sono emersi diversi nuovi standard, in particolare per garantire la confidenzialità ed integrità dei messaggi scambiati e per gestire i processi di autenticazione e autorizzazione in architetture distribuite. Tali standard sono:

- **WS-Security** è una estensione SOAP per garantire l'integrità e confidenzialità dei messaggi

- **SAML** (Security Assertion Markup Language) [SAML] per lo scambio di credenziali fra sistemi distribuiti che devono interoperare

Tali standard fanno tesoro di altri standard già definiti nell'ambito della protezione dei dati strutturati in documenti XML. Infatti, nelle architetture distribuite basate sui Web Services vengono scambiati messaggi strutturati in XML, quindi, ne discende che le soluzioni approntate per proteggere i documenti XML possono essere facilmente riportate in tali architetture.

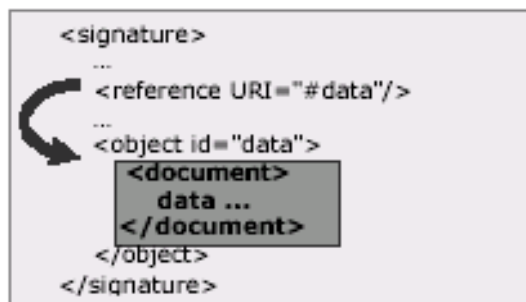
Per tale motivo illustriamo brevemente le caratteristiche degli standard nell'ambito della protezione dei documenti XML. Tali standard sono:

- **XML Signature** che definisce una sintassi XML per firmare digitalmente documenti XML
- **XML Encryption** che definisce una sintassi XML per crittografare i documenti XML
- **XKMS** che definisce un protocollo di dialogo basato su XML che implementa il dialogo con infrastrutture PKI

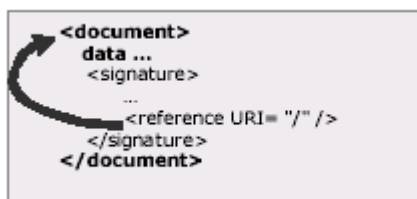
La specifica XML Signature definisce una sintassi XML per rappresentare firme digitali, basata su tecnologie esistenti (XML, Xpath, Xlink ecc.) e le procedure per creare e verificare firme. I dati firmati possono essere un documento XML o parti di un documento XML.

Vengono definiti tre tipi di firme:

Enveloping: la firma incorpora il documento XML da firmare (attraverso un URI interna); la figura sotto esemplifica la definizione



Enveloped: la firma è incorporata nel documento XML che si sta firmando attraverso una URI interna; la figura sotto esemplifica la definizione



Detached: la firma referencia un documento che è separato dal documento XML (external URI)

```
<signature>  
...  
<reference URI="http://x.com/x.txt"/>  
...  
</signature>
```

XML Signature permette di identificare il mittente del messaggio e di verificare che il messaggio non è stato modificato, per cui l'applicazione di tale standard garantisce l'integrità di documenti XML scambiati e il non ripudio.

XML Encryption definisce una sintassi XML per rappresentare documenti XML cifrati. L'applicazione delle specifiche XML Encryption garantisce la confidenzialità nello scambio di documenti XML.

Questi standard applicano gli algoritmi di firma e cifratura già esistenti a documenti XML e strutturano il risultato di tali algoritmi in documento XML. Quest'approccio fa sì che i risultati delle operazioni effettuate, in accordo agli standards XML Signature e XML Encryption, sono a loro volta documenti XML manipolabili attraverso componenti software che riconoscono la sintassi XML. In tal modo, si può aggiungere il supporto alla firma e alla cifratura a qualsiasi architettura si basi sullo scambio di documenti XML senza dover apportare significative alterazioni alle componenti software già realizzate.

XKMS è una specifica W3C che ha l'obiettivo di semplificare l'integrazione di PKI e la gestione dei certificati digitali nelle applicazioni.

Tale specifica standardizza le modalità di gestione, registrazione e revoca dei certificati. I servizi XKMS risiedono su un server e sono accessibili attraverso un semplice protocollo basato su strutture XML.

La specifica XKMS comprende due sotto-servizi:

- *XML Key Information Service Specification (X-KISS)* che definisce la specifica utilizzata per rintracciare le informazioni corrispondenti ad una chiave pubblica; in particolare, la specifica X-KISS stabilisce che il firmatario fornisca le informazioni necessarie ad autenticare la sua firma digitale. Ciò può essere fatto incorporando la chiave pubblica del firmatario nel documento XML oppure invocando un servizio XKMS e richiedere informazioni sul certificato del firmatario ad una certification authority, in modo da determinare l'identità del firmatario e verificare che il certificato non sia stato revocato
- *XML Key Registration Service Specification (X-KRSS)* definisce tutte le procedure che gestiscono il ciclo di vita delle chiavi sono eseguite da servizi X-KRSS: registrazione, revoca, recupero.

L'accesso ai servizi XKMS è codificato attraverso messaggi XML e può essere utilizzato SOAP come protocollo di comunicazione.

Il processo di autenticazione basato sullo scambio di credenziali all'inizio di una sessione di comunicazione tra due entità comunicanti non può essere applicato nelle architetture distribuite basate sui Web Services; in tali architettura infatti ogni scambio di dati avviene su una sessione di comunicazione diversa che non ha memoria di precedenti stati. Per questo sono implementati approcci di autenticazione completamente nuovi per garantire che ogni messaggio scambiato su tali architetture contenga informazioni di autenticazione e che questa informazione possa essere confermata ad ogni scambio di messaggi.

Lo standard *Security Assertion Markup Language* (SAML) è una specifica che definisce un protocollo per lo scambio di *asserzioni di sicurezza tra entità diverse*, definito da un ente internazionale no-profit OASIS - Organization for the Advancement of Structure Information Standards, è stato progettato per permettere l'instaurazione di relazioni garantite tra sistemi che comunicano via web services, in modo tale che gli utenti di un sistema possono essere automaticamente autenticati su un altro sistema.

SAML si basa sulla propagazione e validazione di asserzioni di sicurezza (messaggi strutturati in XML) tra i diversi sistemi cooperanti in un'architettura basata sui Web Services.

SAML si basa sul seguente assunto:

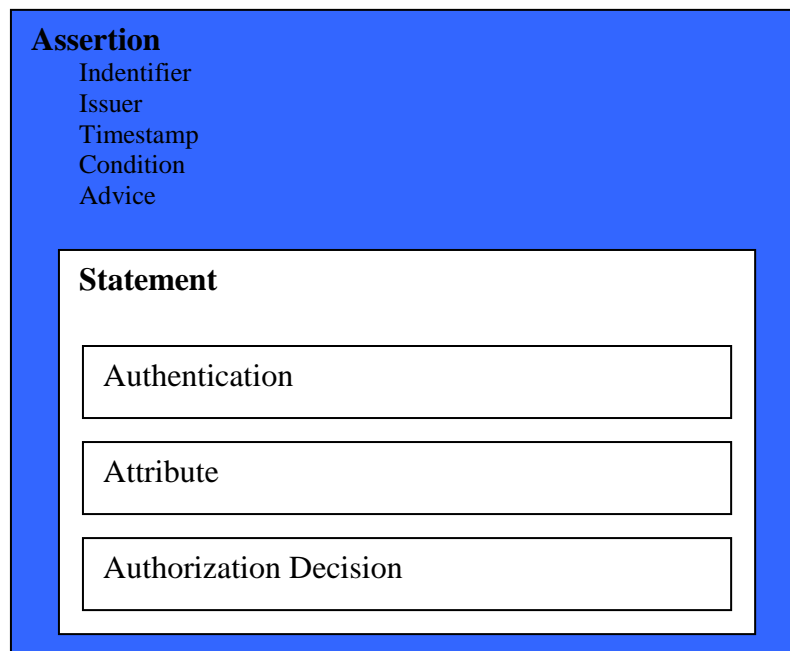
quando esiste una relazione garantita tra due organizzazioni è possibile gestire le autorizzazioni nel seguente modo:

un utente dell'organizzazione A viene autenticato dal sistema di identificazione di A e può accedere anche ai servizi di B, perché l'organizzazione B riconosce valido il token di autenticazione rilasciato da A e non richiede un ulteriore processo di autenticazione, poiché viene ritenuto equivalente ad un token di autenticazione rilasciato dal sistema di identificazione del ricevente.

Sulla scorta di ciò, la specifica SAML prevede diverse tipologie di oggetti: asserzioni, protocolli, bindings e profili.

Le asserzioni sono di 3 tipi:

- asserzioni di autenticazione; del tipo *l'utente ha provato la sua identità*;
- asserzioni di attributo; del tipo *l'utente ha un limite di spesa prefissato*;
- asserzioni di decisione; del tipo *l'utente è autorizzato ad acquistare il bene*.



Ogni asserzione SAML contiene i seguenti dati:

- assertion identifier - un identificatore unico per l'asserzione;
- assertion issuer - un nome unico globale associato all'autentication authority che ha emesso l'asserzione;
- assertion timestamp - i dati temporali sulla emissione della asserzione;
- assertion conditions - i vincoli che determinano quando e come una asserzione è valida;
- assertion advice - informazioni aggiuntive che possono essere ignorate senza inficiare la validità dell'asserzione.

SAML Bindings specificano come i messaggi SAML sono mappati sul protocollo di trasporto, in particolare, SOAP-over-http binding specifica come i messaggi SAML sono trasmessi attraverso il protocollo SOAP.

SAML Profiles specificano come SAML assertions sono incapsulate in un messaggio ed estratte da un messaggio strutturato in accordo al protocollo di trasporto utilizzato. Sono definiti due profili: Web browser profile e WS-Security profile (parte della specifica WS-Security)

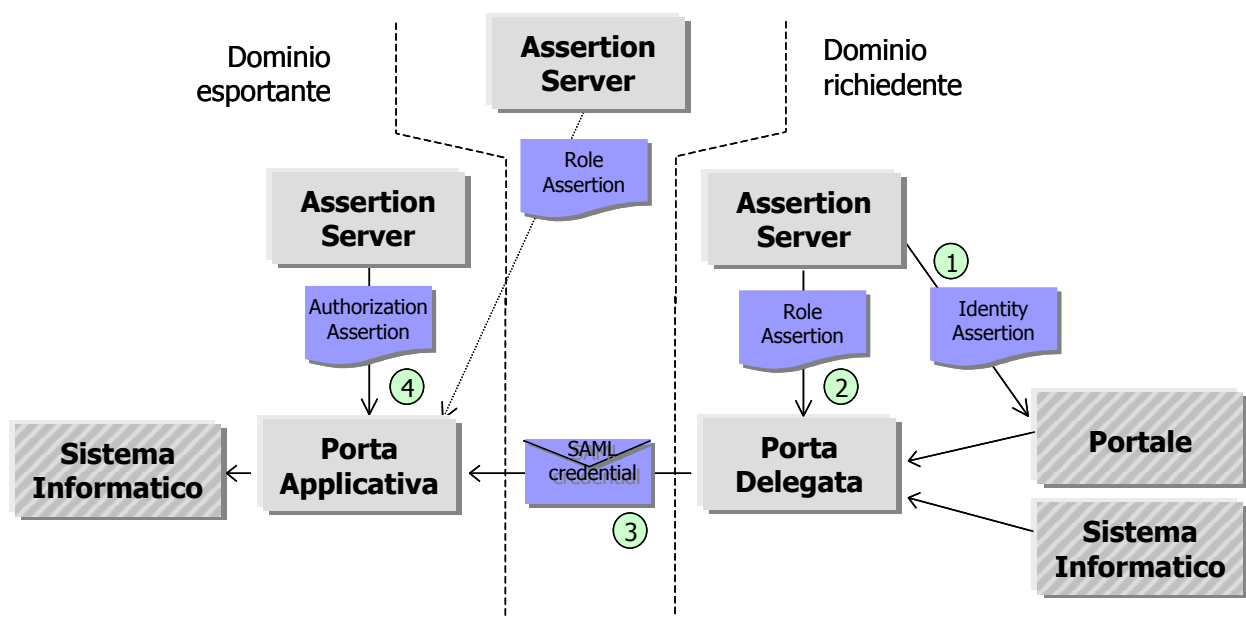
Nell'ambito dell'Infrastruttura di Cooperazione Applicativa, l'utilizzo dello standard SAML permette di risolvere il problema riguardante l'accesso ai servizi di un dato Dominio da parte di utenti registrati in altri Domini.

In particolare, in tal caso la dinamica è la seguente:

- il Dominio da cui parte la richiesta produce una credenziale con l'identificazione (asserzione di identità) ed il ruolo (asserzione di ruolo) del richiedente; per ruolo s'intende la posizione o la carica assunta nel Dominio richiedente; tale Dominio si fa quindi garante di quanto asserito firmando digitalmente la credenziale;
- la credenziale viene presentata dalla Porta Delegata del Dominio richiedente alle Porte Applicative a cui si richiede il servizio;
- verificata l'autenticità della credenziale, il Dominio che espone il servizio richiesto utilizzerà la stessa per ottenere l'asserzione d'autorizzazione dal sistema che gestisce le politiche d'accesso allo specifico servizio;
- dopo aver ottenuto l'autorizzazione, la Porta Applicativa sottoporà la richiesta al Sistema Informatico.

Il Dominio che espone il servizio può richiedere, utilizzando la stessa credenziale, ad altri Domini, asserzioni di ruolo per il richiedente.

Lo schema dato sotto illustra quanto appena descritto.



5.4. Architettura Applicativa dell'Infrastruttura di Sicurezza

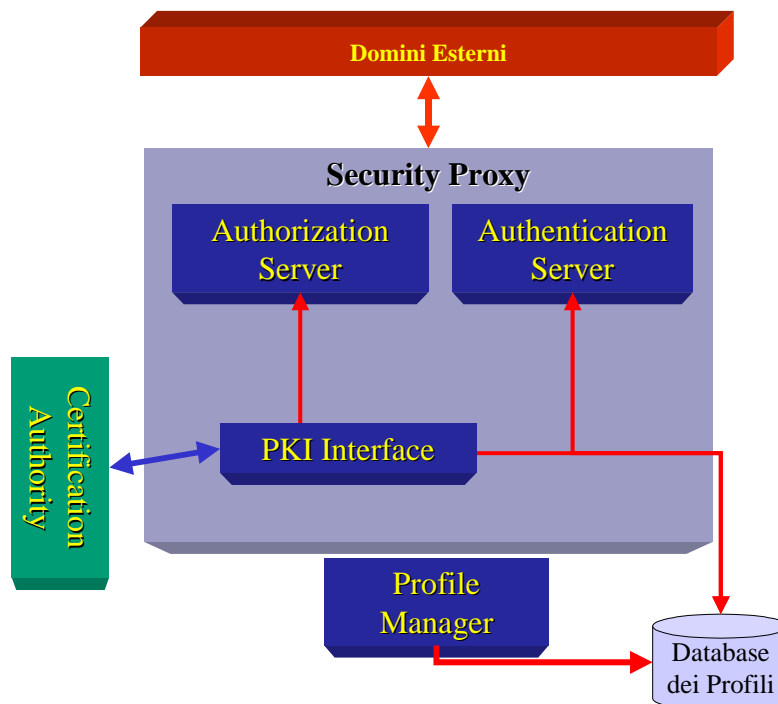
I principali requisiti che la soluzione soddisfa sono:

- *garantire l'autonomia operativa ed organizzativa dei singoli Domini* per ciò che riguarda i sistemi e i modelli di sicurezza utilizzati; l'Infrastruttura di Cooperazione Applicativa necessita dell'implementazione di uno strato di sicurezza interorganizzativo, tale strato non deve intaccare l'autonomia organizzativa e gestionale delle organizzazioni che espongono i servizi sulla rete;
- *essere trasversale alle applicazioni*; i servizi di sicurezza devono essere implementati esternamente a tutti i servizi applicativi e di comunicazione previsti dall'Infrastruttura di Cooperazione Applicativa e devono essere richiamabili da tutte le applicazioni.
- *essere indipendente dalle tecnologie e dagli standard*; data la mutabilità del panorama sia tecnologico che di standard riguardo la sicurezza per architetture *web services based*, la soluzione deve essere implementata in modo tale da:
 - disaccoppiare la semantica dei servizi dalla loro implementazione, non vincolandosi a specifiche tecnologie di sicurezza;
 - rendere le applicazioni *future-proof*, poiché gli standards riguardanti la sicurezza dei web services sono in evoluzione costante, la soluzione deve poter facilmente recepire l'avvento di nuovi standard;
- garantire tutte le funzionalità di sicurezza logica: autenticazione, autorizzazione, non ripudio, confidenzialità e integrità

L'architettura applicativa dell'Infrastruttura di Sicurezza per rispondere a questi requisiti utilizza le seguenti componenti:

- **Security Manager**: una componente collocata sulle Porte di Dominio che contempla un insieme di moduli software che implementano i servizi di sicurezza logica;
- **Profile Manager**: un sistema per la gestione del ciclo di vita del profilo di sicurezza di ciascuna entità: creazione, memorizzazione, dispatching, aggiornamento, cancellazione.

La figura sotto illustra lo schema a blocchi del Gestore della Sicurezza.



Dalla figura si rileva che Security Proxy è costituito dai moduli: *Authentication Server*, *Authorization Server*, *PKI Interface*, *Profile Manager*

Authentication Server implementa il meccanismo di autenticazione che consiste nell'associare in modo certo un richiedente di un servizio a una **identità digitale certificata**, validando le credenziali presentate da un Dominio Esterno che richiede un servizio.

Con **identità digitale** si indica la rappresentazione univoca che un sistema (nel caso in oggetto il sistema informatico di una pubblica amministrazione) ha dell'identità reale dell'utente richiedente.

L'identità del soggetto può essere dedotta:

- dalle credenziali emesse dal Dominio che riceve la richiesta;
- dalle credenziali emesse da una autorità esterna accreditata.

Nel primo caso il processo di autenticazione è integralmente basato su informazioni possedute dal Dominio che riceve la richiesta di servizio. Per fare questo il Dominio deve farsi carico della gestione delle credenziali di ciascun potenziale richiedente di servizio e ciò può essere notevolmente complesso nel caso in cui la platea di utenti potenziali è molto vasta e eterogenea.

Inoltre, uno dei principi su cui si basa l'infrastruttura di cooperazione applicativa è la piena autonomia dei Domini partecipanti e questo principio deve essere ottemperato anche nell'ambito delle *policies* di sicurezza logica, in particolare per ciò che riguarda la validazione delle credenziali.

Per ovviare alle problematiche di complessità sopra dette e per rispettare il principio di autonomia appena enunciato, è possibile implementare un *modello federato di autenticazione* nel quale sono presenti più entità che possono validare credenziali e determinare l'identità digitale.

I Domini facenti parte dell'infrastruttura di cooperazione stabiliscono un rapporto *fiduciario* con tali entità e da ciò consegue che se l'identità digitale di un richiedente è stata determinata da una tale entità, il Dominio a cui viene inoltrata una richiesta di servizio riconosce valida tale identità e viene concesso l'accesso. In definitiva, nel modello federato il modulo Authentication Server di un dato Dominio:

- può eseguire direttamente la validazione delle credenziali se possiede le informazioni necessarie a certificare l'identità digitale del richiedente.
- può delegare ad una entità esterna fidata (che può essere un altro Dominio) la validazione delle credenziali del richiedente e riceverne in risposta l'identità digitale certificata.

L'implementazione del modello federato richiede la sincronizzazione delle informazioni tra le diverse entità deputate ad effettuare l'autenticazione. La sincronizzazione può sfruttare gli stessi meccanismi di interoperabilità messi a disposizione dall'Infrastruttura di Cooperazione Applicativa (Porta di Dominio).

Le credenziali presentate per farsi autenticare possono essere diverse a seconda della tipologia di utenza e dell'operazione richiesta. La differenziazione delle credenziali permette di stabilire diverse policies di autenticazione in funzione dell'utente che effettua la richiesta e del servizio richiesto.

Una policy di autenticazione mira a verificare che il richiedente dell'accesso al sistema possieda una determinata quantità di sicurezza che può essere di diversa forma: memorizzata in un dispositivo fisico, eventualmente attivo (smartcard) oppure essere rappresentata unicamente da informazioni che l'utente stesso conosce.

Pertanto, una policy di autenticazione si caratterizza in termini di:

- *tipo di autenticazione* che può essere login/password, challenge/response, pin ecc.
- *livello di autenticazione* che ne definisce l'affidabilità. I livelli di autenticazione che la soluzione proposta è in grado di gestire si possono scegliere in funzione del grado di sicurezza che si intende raggiungere in un determinato scambio di dati e sono:
 - Livello 0: servizi invocabili senza identificazione del chiamante e senza cifratura dei dati scambiati;
 - Livello 1: servizi invocabili da client individuati attraverso un controllo di user e password, senza cifratura dei dati scambiati;
 - Livello 2: servizi invocabili da Domini individuati attraverso un controllo di user e password, con cifratura dei dati scambiati;
 - Livello 3: i Domini coinvolti nella richiesta e nell'erogazione del servizio vengono individuati attraverso certificati digitali, senza cifratura dei dati scambiati;
 - Livello 4: i Domini coinvolti nella richiesta e nell'erogazione del servizio vengono individuati attraverso certificati digitali, con cifratura dei dati scambiati.

- *token di sicurezza* che indica il dispositivo fisico o il tipo di informazioni che l'utente deve fornire. Il token di sicurezza può essere una password, un certificato digitale, una smartcard, un'impronta digitale ecc. I diversi token possono essere combinati fra loro ed il loro utilizzo è, ovviamente, condizionato dalle caratteristiche del canale tramite il quale l'utente accede al sistema.

Le tre caratteristiche appena menzionate (tipo di autenticazione, livello e token di sicurezza) possono essere combinate in tutte le varianti valide possibili per implementare la specifica policy di autenticazione di un dato utente.

Nel contesto operativo specifico dell'Infrastruttura di Cooperazione Applicativa si possono definire policies di autenticazione distinte per gruppi di utenza, per domini, per famiglie applicative ecc. In altri termini si può creare una matrice gruppi di utenza, famiglie applicative che individua le diverse policies di autenticazione attuata a seconda del gruppo di utenza che accede alla data famiglia applicativa. In questo modo si possono tarare il grado di rigidità dei controlli a cui deve sottostare un utente che intende accedere ad una specifica applicazione.

Il modulo *Authorization Server* ha il compito di verificare le abilitazioni all'accesso possedute dal richiedente autenticato. *Authorization Server* riceve asserzioni SAML e può verificarne la validità nei seguenti modi:

- accede al Database dei Profili del proprio Dominio e verifica se le abilitazioni contenute nelle asserzioni soddisfano le regole stabilite per l'accesso alla specifica risorsa richiesta;
- il Dominio Ricevente ritiene valida l'asserzione poiché proviene da un Dominio fidato e consente l'accesso alla risorsa richiesta.

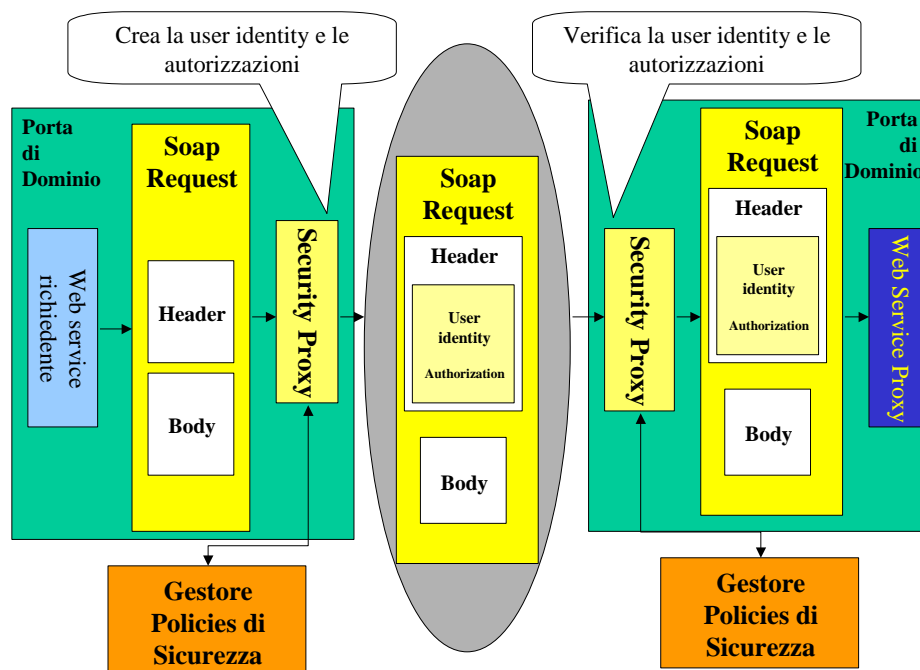
Le due soluzioni prima esposte non sono mutuamente esclusive, ma possono coesistere ed essere utilizzate a seconda dei casi.

Il processo di definizione delle regole che costituiscono le policies di sicurezza deve essere supportato da uno strumento adeguato che permetta di definire le regole di sicurezza e di propagare tali regole ai Domini interessati. Il modulo *Profile Manager* (del sottosistema di Sicurezza) ha il compito di supportare la definizione di tali regole ed alimentare con tali definizioni il Database dei Profili.

5.5. Architettura Esecutiva dell'Infrastruttura di Sicurezza

La soluzione implementata per realizzare le funzioni dell'Infrastruttura di Sicurezza è basata sugli standard XML Encryption, XML-Signature e SAML.

La figura sotto schematizza la soluzione implementativa:



Descriviamo di seguito come i moduli evidenziati in figura operano.

Il Richiedente viene autenticato in accordo allo schema di autenticazione previsto dal Dominio a cui appartiene (HTTP Basic, SSL ecc.).

1. Il messaggio SOAP è catturato da **Security Proxy** che, dialogando col Gestore delle Policies di sicurezza locale, crea le credenziali sotto forma di assertion SAML (user identity), eventualmente firma e cifra il messaggio in accordo allo standard WS Security, utilizzando le funzioni messe a disposizione dal modulo PKI Interface.
2. La request SOAP così modificata, contenente la user identity e altri attributi informativi, viene spedita al destinatario.
3. La request SOAP viene ricevuta dal Security Proxy del Dominio Ricevente che, utilizzando la user identity contenuta nel messaggio, verifica le credenziali e autorizza all'accesso e, eventualmente, verifica la firma digitale apposta e decifra il messaggio, utilizzando le funzioni messe a disposizione dal modulo PKI Interface. La verifica delle credenziali può essere risolta localmente o demandata a una terza parte fidata

Security Proxy è implementato sotto forma di *SOAP Message Handler*, un meccanismo che permette di intercettare messaggi SOAP sia in uscita (messaggi in partenza da un client) sia in arrivo (messaggi ricevuti da un server) prima che essi siano ricevuti o trasmessi.

Come si può facilmente dedurre da quanto descritto sopra, **Security Proxy** agisce come un *application level firewall* che opera sul contenuto della comunicazione. Tale componente esamina i dati XML, che possono essere informazioni di sicurezza o informazioni di autenticazione come una

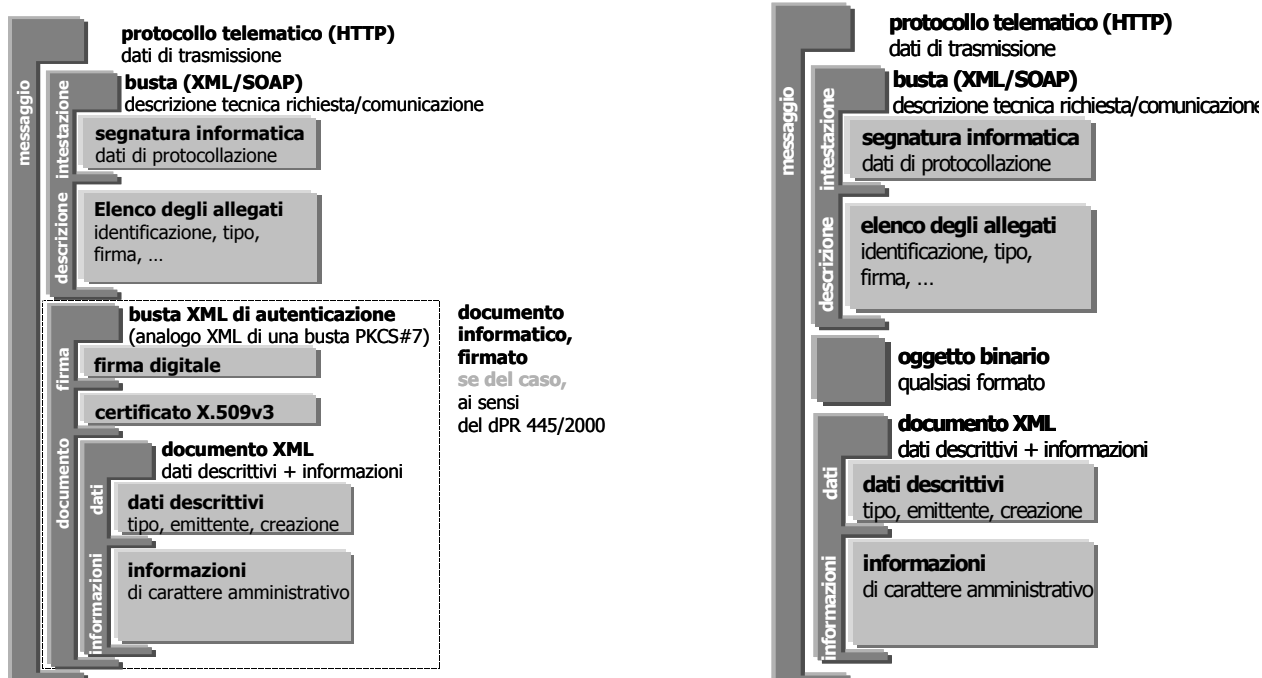
XML Signature o una SAML Assertion che viene valutata dal Gestore delle Policies di Sicurezza Locale.

In definitiva **Security Proxy** implementa tutti i servizi di sicurezza logica:

- autentica ed autorizza i mittenti dei messaggi SOAP, utilizzando identity e attributi per determinare se un utente è autorizzato ad accedere al Web Service (Autenticazione e Autorizzazione)
- protegge i messaggi utilizzando XML-based encryption e firme digitali (Integrità, Confidenzialità e Non Ripudio)

Inoltre, utilizza le policies di sicurezza di ciascuna organizzazione durante i processi di autenticazione e autorizzazione; in questo modo viene garantita l'autonomia delle singole organizzazioni nel definire e governare propri modelli di sicurezza e permette di definire i livelli di protezione indipendentemente dalle singole applicazioni.

L'inserimento della user identity e degli attributi di autorizzazione è una funzione incapsulata nel Security Proxy, per cui gli standard utilizzati (SAML o altro) possono essere aggiornati nel tempo senza che ciò impatti sulle applicazioni.



A puro scopo esemplificativo nella figura data sopra si mostrano due buste di e-government una che contiene una sezione Firma in cui viene mantenuta la firma digitale e il certificato e un'altra in cui

vi è una sezione Oggetto Binario che può contenere qualsiasi tipologia di informazione. Le sezioni appena indicate possono essere entrambe costruite dal Security Proxy in funzione delle regole definite nel Gestore delle Policies di Sicurezza Locale.

Security Proxy implementato come message handler (interceptor) rende trasparente l'infrastruttura di sicurezza alle applicazioni. Gli interceptors intercettano messaggi SOAP e li sottopongono ad un trattamento prima di inviarli al web service destinatario. Apache AXIS, previsto nella piattaforma di esecuzione proposta, permette di definire handlers per trattare SOAP request prima di trasmetterle.