Consiglio Nazionale delle Ricerche

Descrizione e gestione di workflow documentali con una applicazione basata su XML

M. Tesconi, A. Marchetti, O. Signore

IIT TR-06/2003

Technical Report

Luglio 2003

Istituto di Informatica e Telematica

Descrizione e gestione di workflow documentali con una appplicazione basata su XML

Maurizio Tesconi
Ufficio Italiano W3C
maurizio@w3c.it

Andrea Marchetti
IIT-CNR
Andrea.Marchetti@iit.cnr.it

Oreste Signore
ISTI-CNR e Ufficio Italiano W3C
oreste@w3.org

Sommario - I sistemi di workflow coordinano tutte le operazioni che riguardano l'elaborazione e la trasmissione dei documenti, specificando le attività ed i ruoli di tutti gli appartenenti al processo di lavoro. Un document workflow segue un documento durante tutto il suo ciclo di vita, fornendo un'azione di controllo costante per la sua compilazione.
Nello studio presentato si cerca di far luce sulle varie problematiche che sorgono quando si descrivono iter documentali.
A tale proposito viene definito un modello concettuale che permette di descrivere in maniera dettagliata un iter documentale e tutte le attività che si possono effettuare sul documento. Per sviluppare il modello si è scelto di adottare la tecnologia XML, sia per strutturare i documenti che tutte le informazioni relative al flusso. Come *agente* si è intesa una qualsiasi entità, sia umana che software, capace di interagire con il documento, mentre con flusso di documenti si è inteso tutti i possibili percorsi che il documento stesso segue nel suo ciclo di vita, passando da un agente all'altro. Il flusso documentale viene descritto tramite un linguaggio dichiarativo attraverso l'elencazione di tutti gli agenti che partecipano al flusso, specificando tutte le operazioni che ogni agente può svolgere sull'istanza del documento.
I documenti elaborati dai vari agenti hanno una struttura definita da uno schema XML e sono accompagnati per tutto il loro ciclo di vita da altri documenti, che contengono informazioni sul flusso, sui vincoli e sulla visualizzazione dei dati. Una particolare enfasi è data ai problemi relativi alla fusione di due o più documenti compilati da molteplici agenti in maniera concorrente.
Per quanto concerne la progettazione di un sistema di gestione di workflow documentali, sono due le soluzioni architetturali analizzate: quella centralizzata e quella distribuita.
Al fine di rappresentare graficamente i documenti da elaborare si utilizza il browser XSmiles, in grado di visualizzare documenti Xhtml con all'interno moduli XForms.
Adoperando tecniche innovative, come XML-Signature, sono stati presi in esame tutti gli aspetti legati alla firma dei documenti modificati dagli agenti.

Introduzione

Solitamente l'enorme quantità di documenti presenti all'interno delle strutture pubbliche e private segue un proprio iter che comprende varie tappe prima di arrivare a compimento.

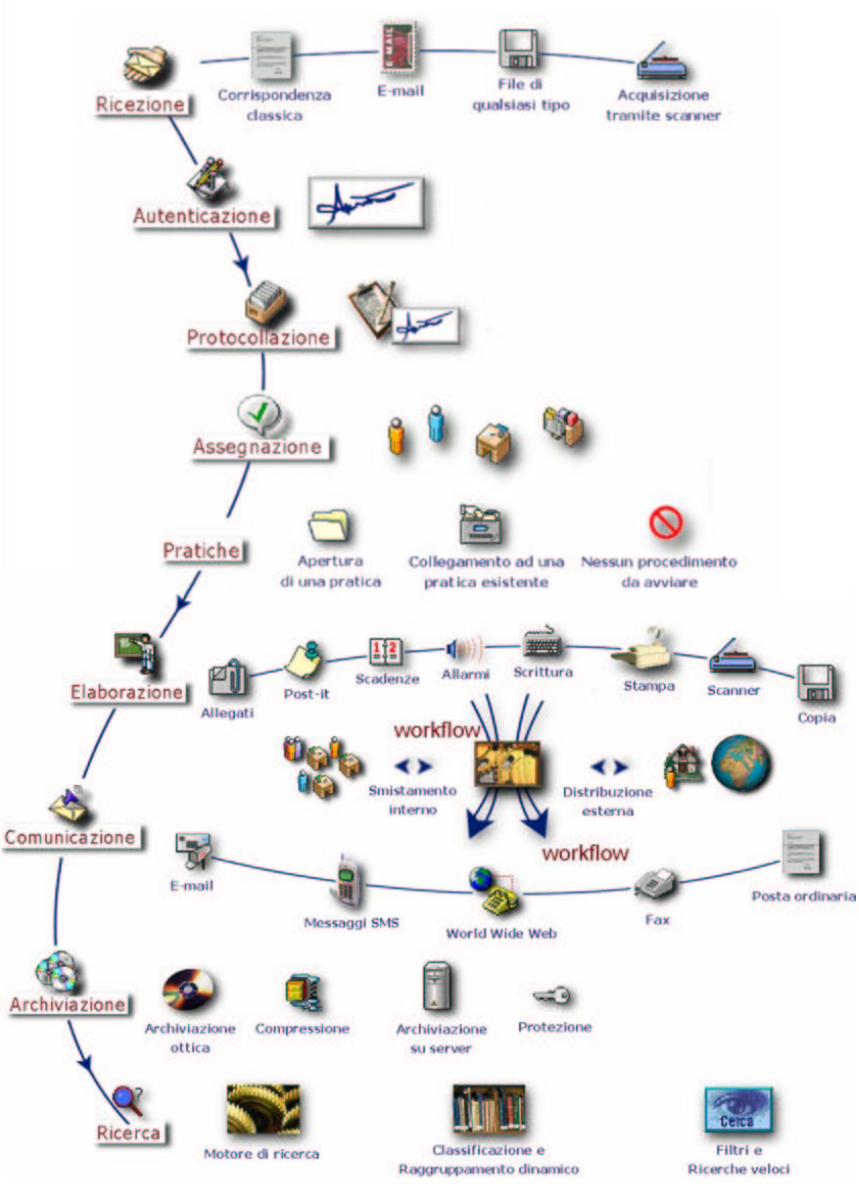

Figura 1 Esempio di ciclo di vita di un documento

Nella figura 1 si può vedere un esempio del ciclo di vita di un documento: un documento dopo che è stato creato, acquisito, convertito in formato elettronico e firmato deve essere protocollato ed assegnato ad una certa area di lavoro.
I sistemi di workflow coordinano tutte le operazioni che riguardano l'elaborazione e la comunicazione dei documenti prodotti.
Un workflow documentale (document workflow) può essere inteso come l'automazione e la gestione dell'iter di particolari documenti (pratiche) tramite degli specifici software di supporto.
Un processo di document workflow nella sostanza segue un particolare documento durante tutto il suo ciclo di vita, fornendo una azione di controllo costante per la sua compilazione.
L'idea che sta alla base di questo studio è quella di associare ad un documento delle informazioni riguardanti sia la propria vita sia il percorso che questo dovrà effettuare nel corso della propria esistenza.

Lo scopo principale è riuscire a definire un modello astratto che descriva in maniera dettagliata un iter documentale e tutte le attività che si possono effettuare sul documento.
Per sviluppare il modello la prima scelta fatta è stata quella di adottare la tecnologia XML sia per strutturare i documenti che tutte le informazioni relative al flusso.
È necessario a questo punto definire alcuni termini che verranno correntemente usati per la descrizione del modello: si definisce *agente* una qualsiasi entità, sia umana che software, che interagisca con il documento; per flusso di documenti intendiamo tutti i possibili percorsi che il documento stesso segue nel suo ciclo di vita, passando da un agente all'altro.
Naturalmente il flusso non è riferito ad un solo documento, ma ad una classe di documenti definiti da uno schema XML.

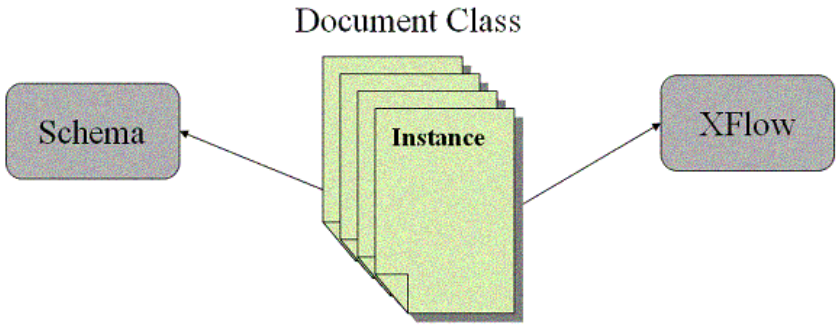

Figura 2 Associazione di uno schema e un flusso ad una classe di documenti

Un documento, una volta creato, viene elaborato dagli agenti che incontra durante il proprio ciclo di vita. Il percorso che deve effettuare può essere visto come un grafo orientato.
La descrizione di flusso unitamente allo schema devono accompagnare ogni istanza di documento lungo il percorso tra i vari agenti.
Ogni agente potrà modificare l'istanza rispettando delle regole scritte nella descrizione del flusso.

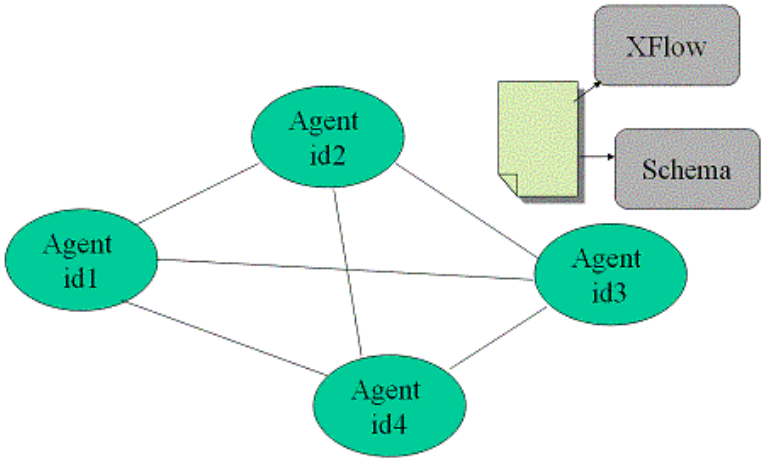

Figura 3 Scambio di documenti tra agenti

Durante la progettazione del modello sono state provate numerose sintassi per descrivere un flusso complesso. Una possibile soluzione è quella di usare una notazione simile ai linguaggi di programmazione concorrente. Istruzioni come fork e join possono essere usate per descrivere ramificazioni e ricongiungimenti di flusso ma questa soluzione implica l'uso di etichette per specificare i punti di sincronizzazione.
Data la natura dell'XML, ben rappresentabile con una struttura ad albero, la soluzione più appropriata è apparsa quella di descrivere il flusso dal punto di vista degli agenti.
Ogni agente può ricevere, elaborare ed inviare il documento.
Per descrivere un flusso documentale quindi basta elencare tutti gli agenti e scrivere in maniera dettagliata tutte le operazioni che ogni agente può svolgere sull'istanza del documento. In questo

caso per operazione si intende anche quella di ricevere un documento da un certo agente o di inviarlo ad un altro.
Nella progettazione del modello sono stati perseguiti i seguenti fini:

- **generalità**: si è cercato di descrivere il flusso nel modo più generale possibile ed indipendente dalle implementazioni. Per questo motivo all'interno della descrizione del flusso non si fa mai riferimento a degli URL specifici ma si usano dei nomi di carattere generale che devono essere risolti dall'ambiente che gestisce il flusso documentale;
- **semplicità**: per quanto possibile, si è cercato di descrivere tutte le informazioni riguardanti il flusso in maniera chiara e leggibile. La tecnologia XML si presta bene a descrivere le informazioni strutturate in modo comprensibile sia da un agente software che umano.

Rappresentazione grafica del flusso

Un workflow documentale, e più in generale qualunque workflow, può essere rappresentato in maniera molto semplice ed intuitiva tramite un diagramma di attività.

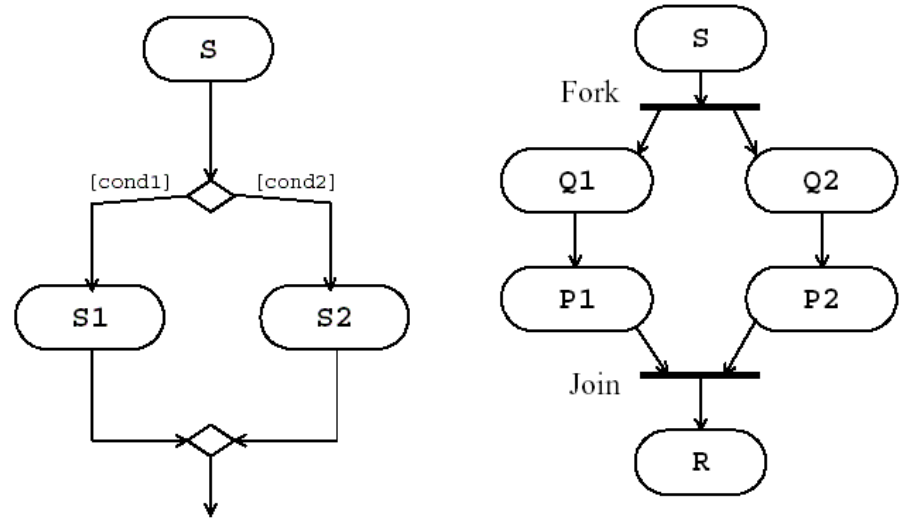

Figura 4 Diagramma di attività con Fork e Join

Nello studio che segue quello che vogliamo descrivere non è un workflow ma l'iter che compie un documento durante il proprio ciclo di vita.
Dal punto di vista del workflow documentale le attività vanno intese come tutte le operazioni che un agente può svolgere sul documento, mentre i flussi paralleli possono essere interpretati come una duplicazione del documento ed un successivo invio in più copie del documento stesso.
Sotto questo punto di vista l'operazione rappresentata dalla Join deve essere intesa come una fusione di un insieme di documenti elaborati in parallelo.

Rappresentazione con XML

Per rappresentare in XML il flusso di una classe di documenti si è creato un modello chiamato XFlow, che permette di descrivere in maniera dettagliata il percorso e le varie attività che vengono svolte sul documento.
Il documento una volta istanziato contiene al suo interno tutti i dati necessari per risalire alle informazioni sul flusso, sulla rappresentazione e sulle attività.
Per fare una descrizione di flusso che fosse il più generale possibile si è scelto di utilizzare nomi del tutto generici sia per gli agenti che per le attività svolte sul documento.

Il compito di risolvere i nomi si demanda al motore software, che deve gestire il flusso di documenti oppure a degli agenti locali a seconda dell'architettura utilizzata.

Il documento
La classe di documenti viene definita attraverso un XML Schema, in cui si forniscono informazioni precise riguardo la struttura del documento e soprattutto sul tipo dei campi.
L'istanza oltre ai valori dei vari campi del documento ha al suo interno dei riferimenti al flusso, allo schema, e alla sua visualizzazione grafica: deve avere come radice un elemento `<Instance>` con un insieme di attributi, quali:

- **Name**: identifica in modo univoco il documento;
- **Schema**: indica l'URI dello schema xml che definisce la classse di documenti xml a cui il documento appartiene;
- **Header**: indica l'URI del documento XML che contiene tutte le informazioni sullo stato attuale del documento;
- **Log**: indica l'URI del documento XML dove sono salvate tutte le operazioni effettuate sul documento e le firme dei vari agenti;
- **XFlow**: indica l'URI del documento xml che descrive il flusso;
- **Form**: è un attributo opzionale che indica l'URI del documento xml contenente alcune informazioni relative alla visualizzazione tramite XForms.

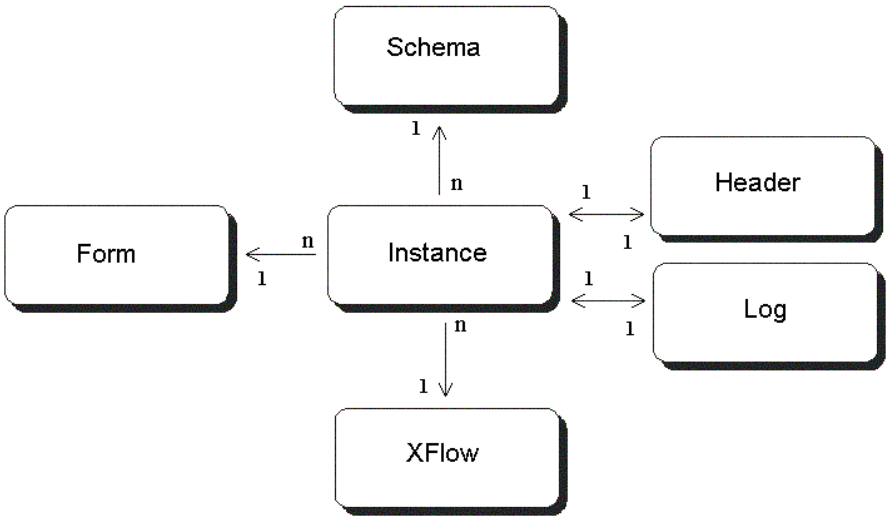

Figura 5 Diagramma E-R di un'istanza di documento

Uno schema XML definisce una classe di documenti, inoltre tramite gli XML Schema è possibile definire dei tipi per i campi e la struttura del documento.
Alla classe di documenti viene aggiunta la descrizione del suo ciclo di vita tramite il documento XFlow ed una descrizione dei controlli per visualizzare il documento tramite dei moduli XForms.
Una possibile istanza di documento potrebbe essere la seguente:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<Instance
        Name = "Document.xml"
        Schema = "Document.xsd"
        Header = "Header.xml"
        Log = "Log.xml"
        XFlow = "XFlow.xml"
```

```
        Form = "Form.xml">
        <Esempio>
                <Campo1>Prova</Campo1>
                <Campo2>777</Campo2>
                <Campo3>true</Campo3>
        </Esempio>
</Instance>
```

Il documento Header

Il documento Header contiene informazioni sullo stato di un'istanza di documento durante il proprio percorso tra gli agenti. Esiste una corrispondenza biunivoca tra un documento ed il suo header in quanto quest'ultimo è una sorta di intestazione che accompagna il documento durante il suo iter tra gli agenti.
Si può pensare all'header di un documento come a quello che viene aggiunto ai messaggi dai protocolli di trasporto per aggiungere delle informazioni di controllo.
Ogni volta che un agente invia un documento deve aggiornare i valori contenuti nell'header come ad esempio il nome del mittente e del destinatario del documento oppure le informazioni sulla data e sull'ora di invio.
La radice del documento deve essere un elemento `<Header>` ed al suo interno sono previsti vari elementi:

- `<ID>`: contiene un identificatore univoco
- `<Sender>`: contiene il nome dell'agente mittente del documento
- `<Receiver>`: contiene il nome dell'agente destinatario del documento
- `<Time>` : contiene l'ora in cui è stato inviato il documento.
- `<Date>` : contiene la data in cui è stato inviato il documento.
- `<DeadLine>` : contiene la data e l'ora entro la quale il documento deve completare il proprio percorso. Può essere utile per gestire delle politiche di priorità.
- `<Duplicate>`: (yes/no) indica se il documento è stato duplicato.

L'elemento `<Duplicate>` viene impostato al valore `yes` dal sistema quando deve inviare più copie del documento in modo tale che, come vedremo in seguito, un agente di tipo Merge sia informato che l'istanza ricevuta è una copia del documento.
Un esempio di documento Header potrebbe essere:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<Header>
        <ID>1234567890</ID>
        <Sender>Agent1</Sender>
        <Receiver>Agent2</Receiver>
        <Time>10:35:12</Time>
        <Date>2003-3-5</Date>
        <DeadLine>2003-6-7 12:00:00</DeadLine>
        <Duplicate>no</Duplicate>
</Header>
```

Il documento Log

Il documento Log è un documento XML che contiene la lista di tutte le operazioni che vengono effettuate sul documento e le firme degli agenti.
Grazie a questo documento è possibile:

- monitorare semplicemente il ciclo di vita di un documento;
- verificare l'integrità e l'autenticità dei dati inseriti;
- recuperare il flusso in caso di crash;
- effettuare statistiche sui documenti.

La struttura del documento di Log è molto semplice, infatti questo viene costruito in maniera sequenziale dai vari agenti ogni volta che questi effettuano l'operazione di submit di un documento.

Per ogni documento esiste un unico Log che cresce mano a mano che il documento viene compilato lungo il suo iter.
Quando un agente effettua l'operazione di submit il sistema esegue le seguenti operazioni: aggiunge al documento Log un elemento `<Log>` specificando nell'attributo `agentname` il nome dell'agente; in seguito inserisce all'interno dell'elemento le seguenti informazioni:

- l'Header ricevuto insieme al documento;
- il frammento di documento XML che è accessibile all'Agente;
- la digital signature relativa solo al frammento;
- le informazioni relative al tempo di invio e ai destinatari del documento.

Un agente firma solo un frammento del documento (la parte accessibile) perché l'intero documento potrebbe contenere molte altre informazioni compilate da altri agenti.
Quando un documento viene duplicato vengono creati tanti Header, uno per ogni istanza di documento. Il documento Log invece resta unico.
Le informazioni temporali presenti all'interno del Log permettono di monitorare il flusso del documento ed individuare eventuali problemi come ad esempio dei colli di bottiglia.

Il documento Form

Il documento Form viene utilizzato dal sistema per fornire agli agenti umani una rappresentazione grafica del documento e permette di descrivere gli elementi grafici tramite i quali si possono modificare i campi del documento.
Il documento XML viene in questo modo visualizzato come un modulo con dei campi da compilare, dei bottoni ed altri elementi grafici.
I tag usati per descrivere i vari controlli grafici sono quelli usati nella tecnologia XForms.
Il documento Form non viene preso in considerazione dagli agenti software.
Un documento Form deve avere come radice un elemento `<Form>`, all'interno del quale si trova una descrizione di tutti i controlli associati ai vari campi dell'istanza del documento.
In questo modo si riescono a definire i vari tipi di controlli che dovranno essere visualizzati per interagire con i campi dell'istanza.
Nel documento Form sono descritti tutti gli elementi grafici del modulo; gli agenti però devono visualizzare solo alcuni di questi controlli perché nella descrizione del flusso sono espressi per ciascun agente dei vincoli sulla visibilità dei campi.
Per rendere la rappresentazione grafica più chiara e piacevole è possibile assegnare dei fogli di stile al documento specificandone l'URI con un attributo stylesheet dell'elemento `<Form>`.
Il documento Form potrebbe essere generato in maniera automatica dallo schema del documento e poi modificato da chi progetta il flusso per migliorare la visualizzazione dei documenti.

```
<Form stylesheet="FoglioDiStile.css"
              xmlns:xfm="http://www.w3.org/2002/xforms/cr"
              xmlns:ev="http://www.w3.org/2001/xml-events">

              <xfm:input ref="//Esempio/Campo1">
                      <xfm:label>Campo1</xfm:label>
                </xfm:input>
              <xfm:trigger>
                     <xfm:label>Button</xfm:label>
                     <xfm:action ev:event="click">
                        <xfm:load ev:event="click" src="File.xml"
                                                          show="new"/>
                      </xfm:action>
              </xfm:trigger>

              <xfm:submit>
                     <xfm:label>SUBMIT</xfm:label>
              </xfm:submit>
       </body>
</Form>
```

In questo esempio viene inserito un controllo di input per il Campo1, un bottone che permette di caricare e mostrare un file XML ed un bottone per fare il submit del form.

Il documento XFlow

Il documento XFlow contiene la descrizione del flusso ed ha come radice l'elemento `<XFlow>`; al suo interno ci possono essere uno o più elementi `<Agent>` che a loro volta conterranno tre elementi (`<Receive>, <Action>, <Send>`).
Vediamo uno schema per questo documento:

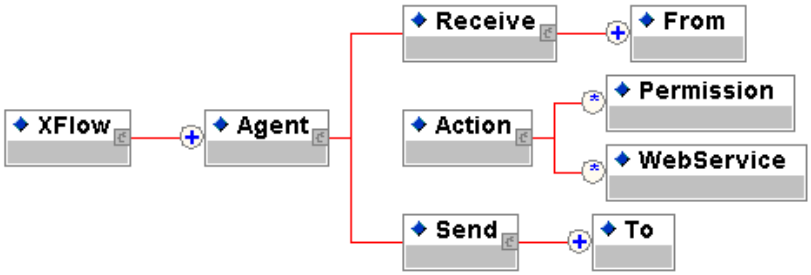

Figura 6 Schema del documento XFlow

L'elemento Agent.

```
<Agent      name="NomeAgente"
            merge="yes|no"
            sign=" yes|no"
            timeout="maxdelay">
```

Ogni elemento `<Agent>` ha un attributo obbligatorio `name` in cui si deve specificare il nome dell'agente e un attributo opzionale `merge` che può valere yes o no a seconda che l'agente sia di tipo Merge o meno.
Gli agenti di tipo Merge sono quelli che si occupano della fusione delle istanze di documento quando questo è stato duplicato in seguito ad una Fork.
Per default un agente non è di tipo Merge.
Con l'attributo `sign` si può specificare se un agente deve firmare i dati immessi.
L'attributo opzionale `timeout` permette di specificare il tempo massimo di permanenza del documento su quell'Agente.

L'elemento Receive.

```
<Receive>
      <From>NomeAgente</From>
      …
</Receive>
```

Nella sezione Receive vengono specificati gli agenti da cui è possibile ricevere il documento.
All' interno di un elemento `<Receive>` possono essere presenti uno o più elementi `<From>`, il cui valore indica il nome degli agenti da cui si può ricevere il documento.

L'elemento Action.

Nella sezione Action si possono invocare dei web-services tramite il seguente costrutto:

```
<WebService action="ServiceName" />
```

In queso modo l'istanza del documento viene passata al web-service associato al ServiceName, che potrà così effettuare tutte le operazioni richieste dal servizio e ritornare l'istanza modificata.

All'interno dell'elemento `<Action>` possono esistere inoltre uno o più elementi `<Permission>` dove sono specificati i permessi dell'Agent.

```
<Permission nodeset=”XpathExpr.” NomeAttributo=”valore” />
```

L'elemento `<Permission>` deve avere un attributo `nodeset` in cui con una espressione XPath si selezionano un insieme di nodi; inoltre questo elemento può avere uno o più attributi che specificano il permesso.
La sintassi adottata è la stessa usata nella tecnologia XForms.
Gli attributi consentiti sono:

- **readonly** : indica che l'insieme di nodi è a sola lettura.
- **required**: indica che l'insieme di nodi è richiesto per poter proseguire
- **relevant**: fornisce delle indicazioni sulla visibilità del documento. Per default tutti i campi hanno l’attributo relevant impostato al valore true, quindi occorrerà impostarlo al valore false per tutti i campi che non si vogliono rendere visibili ad un certo agente.

Il valore di questi attributi può essere una qualunque espressione XPath che restituisca un booleano.

L’elemento Send.

```
<Send>
        <To>NomeAgente</To>
        …
</Send>
```

La sezione Send indica i successivi destinatari del documento.
Il documento può essere inviato contemporaneamente anche a più agenti, ognuno dei quali compila vari campi, unificati in seguito da un agente di merging.
All’interno dell'elemento Send possono essere presenti uno o più elementi `<To>`, il cui valore indica il nome degli agenti a cui spedire l'istanza.
Se ci sono più elementi `<To>` il documento deve essere replicato e inviato a tutti i destinatari.
Vediamo un esempio di flusso XFlow:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<XFlow>
…
<Agent Name="Agent2">
        <Receive>
                <From>Agent1</From>
        </Receive>
        <Action>
                <Permission nodeset="//Campo1" required="true()"/>
                <Permission nodeset="//Campo2" readonly="true()"/>
                <Permission nodeset="//Campo3" relevant="false()"/>
                <WebServices action="Azione" />
        </Action>
        <Send>
                <To>Agent3</To>
        </Send>
</Agent>
<Agent Name="Agent3">
…
</Agent>
…
</XFlow>
```

Per poter descrivere dei flussi complessi sono state introdotte delle istruzioni condizionali usando la stessa sintassi dell'XSLT.
All'interno di una qualunque delle tre sezioni dell'agente si possono utilizzare i seguenti costrutti:

- <**if** test="XPathExpression">: gli elementi interni all' if vengono considerati solo se l'espressione XPath vale true.
- <**choose**>: indica una scelta fra più opzioni.
  All'interno di un elemento choose si possono scegliere le opzioni con i seguenti costrutti:
  - <**when** test="XPathExpression"> questa opzione è verificata se l'espressione XPath vale true.
  - <**otherwise**> viene verificata se tutte le <when> precedenti sono risultate false

Come valore per gli attributi test ci può essere una qualunque espressione XPath che restituica un booleano. Può risultare molto utile fare riferimento alle informazioni contenute dentro l'Header, in questo modo infatti si può risalire all'agente che ha inviato il documento.
Le espressioni XPath quindi possono fare riferimento sia al valore dei nodi dell'istanza del documento sia al valore dei nodi del Header, in questo caso nel percorso si premette //Header/

Vediamo un esempio:

```
<Agent Name="Agent7">
	<Receive>
		<From>Creator</From>
		<From>Merging</From>
	</Receive>
	<Action>
		<choose>
			<when test="//Header/Sender[.='Agent1']">
			<Permission nodeset="//Campo1" relevant="false()"/>
			</when>
			<otherwise>
				<Permission nodeset="//Campo2"
                                        readonly="true()"/>
			</otherwise>
		</choose>
		</Action>
		<Send>
			<choose>
				<when test="//*[Campo3[.='true']">
                                        <To>Agent3</To>
                                        <To>Agent5</To>
				</when>
				<otherwise>
                                        <To>Agent8</To>
				</otherwise>
			</choose>
		</Send>
</Agent>
```

In questo esempio si vede come applicare delle politiche di accesso ai dati diverse in base all'agente che ha inviato il documento.
Inoltre si vede come si può inviare il documento ad agenti diversi controllando il valore di un campo interno all'istanza.
Naturalmente nella parte Send viene controllato il valore assunto dal campo dopo la pressione del pulsante Submit.

Agenti di Merging

Gli agenti di tipo Merge sono agenti software che si occupano della fusione dei documenti.
Questi agenti sono anche dei punti di sincronizzazione, infatti attendono l'invio di documenti da parte di tutti gli agenti compresi nella sezione Receive, per poi procedere ad una fusione controllata dei vari campi del documento.
La fusione del documento viene effettuata attraverso passi successivi:

- i valori di ogni campo di un documento proveniente da un agente vengono racchiusi da un elemento `<value>` con un attributo `author` che contiene il nome dell'agente che ha inviato il documento;
- un volta arrivati tutti i documenti, ne viene creato uno con la stessa struttura ma contenente per ogni campo la lista dei valori assegnati dai vari agenti;
- in seguito vengono applicate le politiche di merging scritte nella sezione `<Action>` del relativo agente.

Dopo queste operazioni gli elementi `<value>` vengono eliminati dal documento fuso.
Vediamo in dettaglio quali possono essere le varie politiche di merging.
Le politiche di merging vengono descritte tramite l'elemento `<Select>` contenuto nella sezione `<Action>` degli agenti di tipo Merge.
Il Select permette di selezionare i valori dei campi dei vari documenti da fondere secondo quanto specificato nei suoi attributi.
La sintassi di questo elemento è la seguente:

```
<Select
      Nodeset= ”XPathExpression”
      From=”AgentName”
      Action=”ActionType” />
```

Gli attributi hanno il seguente significato:

- **Nodeset:** identifica un insieme di campi su cui si vuole effettuare una certa azione
- **From:** indica che i valori devono essere presi dall'Agente specificato. In questo caso non è necessario usare l'attributo action.
- **Action:** indica il tipo di azione da effettuare sui valori selezionati che provengono da tutti gli agenti specificati nella Receive.

Le azioni permesse sono:

- **Min:** seleziona il minimo tra tutti i valori;
- **Max:** seleziona il max tra tutti i valori;
- **Sum:** esegue la somma dei valori;
- **Append:** i valori vengono aggiunti ad una lista racchiusi nell'elemento value;
- **First:** prende il valore dal primo agente che invia il documento
- **Last:** prende il valore dall'ultimo agente che invia il documento

Per i nodi del documento per cui non è specificata una `<Select>` il sistema deve semplicemente effettuare un controllo sui valori provenienti dai vari agenti, e segnalare un errore nel caso in cui siano presenti dei contenuti diversi.
Vediamo un esempio di documento XML che il sistema deve generare prima di effettuare l'operazione di merging:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<Instance form="Form.xml" header="Header.xml"
        name="Document.xml" schema="Document.xsd" xflow="XFlow.xml">
       <Esempio>
              <Campo1>
                            <Value author="Agent1">Prova1</Value>
                            <Value author="Agent2">Prova2</Value>
                            <Value author="Agent3">Prova3</Value>
              </Campo1>
              <Campo2>
                            <Value author="Agent1">5</Value>
                            <Value author="Agent2">10</Value>
                     <Value author="Agent3">7</Value>
              </Campo2>
              <Campo3>
                            <Value author="Agent1">true</Value>
```

```
                                <Value author="Agent2">false</Value>
                                <Value author="Agent3">true</Value>
                </Campo3>
        </Esempio>
</Instance>
```

Vediamo un esempio di agente di merging che ricompone il documento seguendo determinate politiche:

```
<Agent Name="Agent4" Merge="yes">
        <Receive>
                <From>Agent1</From>
                <From>Agent2</From>
                <From>Agent3</From>
        </Receive>
        <Action>
                <Select nodeset="//Campo1" from="Agent3" />
                <Select nodeset="//Campo2" action="Max" />
                <Select nodeset="//Campo3" action="First" />
        </Action>
        <Send>
                <To>Agent5</To>
        </Send>
</Agent>
```

Questo agente prende il valore del campo1 dall'agente numero tre, seleziona il max tra tutti i valori presenti nel campo2 e sceglie il valore del campo tre prendendolo dal primo agente che invia il documento.
L'istanza risultato del merge che verrà poi inviata all'Agent5 sarà fatta nel seguente modo:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<Instance form="Form.xml" header="Header.xml"
         name="Document.xml" schema="Document.xsd" xflow="XFlow.xml">
        <Esempio>
                <Campo1>Prova3</Campo1>
                <Campo2>10</Campo2>
                <Campo3>true</Campo3>
        </Esempio>
</Instance>
```

Un Esempio di workflow documentale: trattamento ordini di missione

Durante lo svolgimento di questo lavoro sono stati presi in considerazione diversi flussi di documenti. Uno degli esempi che forse mette meglio in evidenza alcuni aspetti e potenzialità di XFlow è la procedura che segue un ordine di missione.
Una possibile rappresentazione grafica di questo flusso è la seguente:

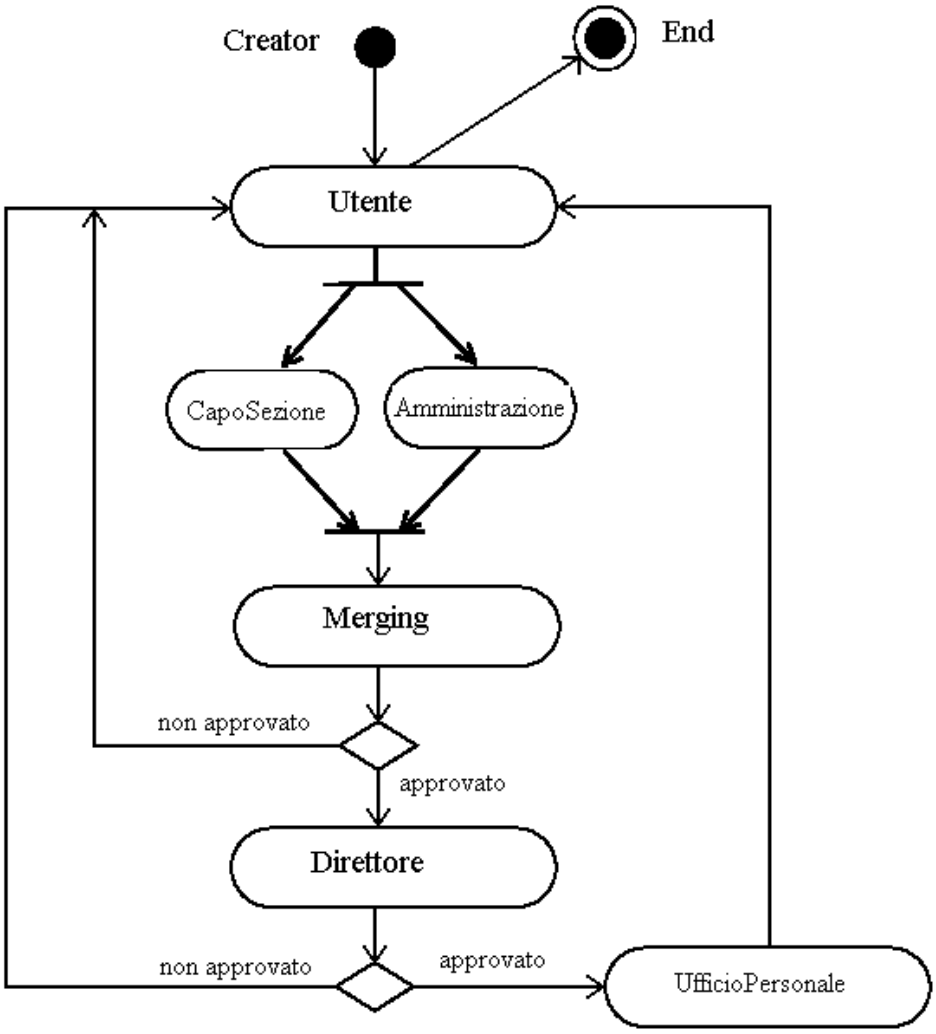

Figura 7 Diagramma di attività di un ordine di missione

Proviamo a descrivere il ciclo di vita di questo documento: un utente che riceve un ordine di missione deve compilare alcuni campi come ad esempio il luogo, la durata e il periodo nel quale dovrà andare in missione. Il documento poi deve essere approvato dal proprio capo ufficio e dall'amministrazione che verifica anche le disponibilità economiche richieste dalla missione. Queste due attività possono essere svolte in parallelo in quanto non dipendono l'una dall'altra. A questo punto il documento che è stato duplicato ed elaborato separatamente nei due uffici deve essere opportunamente riunito e passato al Direttore in quanto è lui ad avere l'ultima parola sull'approvazione della richiesta di missione. Se il documento è stato approvato da tutti passa all'ufficio personale dove semplicemente vengono registrati i dati sulla missione e poi torna all'utente che può verificare l'esito della richiesta. Nel caso in cui il documento non sia stato approvato torna subito verso l'utente, che può leggere le eventuali motivazioni che hanno portato a respingere la domanda.
Prima di descrivere in modo dettagliato il flusso analizziamo un documento XML che potrebbe rappresentare l'istanza di documento di un modulo di ordine di missione.

Mission.xml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<Instance form="MissionForm.xml" header="MissionHeader.xml"
         name="Mission.xml" schema="Mission.xsd" xflow="MissionXFlow.xml">
        <Missione>
                <Richiedente>
```

```
                <Nome>Maurizio</Nome>
                <Cognome>Tesconi</Cognome>
                <Matricola>170353</Matricola>
                <Organizzazione>ISTI</Organizzazione>
            </Richiedente>
            <DescrizioneMissione>
                <Oggetto>Conferenza</Oggetto>
                <Località>Milano</Località>
                <DataInizio>2002-12-10</DataInizio>
                <DurataPrevista>7</DurataPrevista>
            </DescrizioneMissione>
            <CapoSezione>
                <Approvato>nil</Approvato>
                <Motivazione></Motivazione>
            </CapoSezione>
            <Amministrazione>
                <Approvato>nil</Approvato>
                <Capitolo></Capitolo>
                <Motivazione></Motivazione>
            </Amministrazione>
            <Direttore>
                <Approvato>nil</Approvato>
                <Motivazione></Motivazione>
            </Direttore>
        </Missione>
</Instance>
```

Come si può notare questa istanza è già stata compilata con valori relativi all'utente e con dati sul periodo e il luogo della missione.
Questo documento ha una struttura ben definita che separa i vari campi a seconda dell'agente che dovrà interagire direttamente con gli stessi: questo semplifica la descrizione dei permessi per i vari agenti.
I campi `Approvato` sono istanziati per default con il valore 'nil': questo valore indica che non è ancora stata effettuata una scelta sull'approvazione o meno del documento.
Nell'esempio si può notare l'elemento radice di questo documento ed i suoi attributi che legano l'istanza a tutte le altre informazioni necessarie alla vita del documento.
I valori relativi al Richiedente potrebbero anche essere inseriti in maniera automatica durante la creazione dell'istanza del documento da parte del sistema che, una volta autenticato l'agente, può accedere ad una base di dati dove reperire le informazioni, compilare il documento ed inviarlo all'utente che in questo modo dovrà inserire solo i dati relativi alla missione.
Vediamo di seguito un possibile XML schema per documenti di questo tipo.

Mission.xsd

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
  <xsd:element name="Missione">
    <xsd:complexType mixed="false">
      <xsd:sequence minOccurs="1" maxOccurs="1">
        <xsd:element name="Richiedente">
            <xsd:complexType mixed="false">
                <xsd:sequence minOccurs="1" maxOccurs="1">
                        <xsd:element name="Nome"
                                                type="xsd:string"/>
                        <xsd:element name="Cognome"
                                                type="xsd:string"/>
                        <xsd:element name="Matricola"
                                                type="xsd:string"/>
                        <xsd:element name="Organizzazione"
                                                type="xsd:string"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="DescrizioneMissione">
            <xsd:complexType mixed="false">
                <xsd:sequence minOccurs="1" maxOccurs="1">
                        <xsd:element name="Oggetto"
                                                type="xsd:string"/>
                        <xsd:element name="Località"
```

```
                                                type="xsd:string"/>
                        <xsd:element name="DataInizio"
                                                type="xsd:date"/>
                        <xsd:element name="DurataPrevista"
                                                type="xsd:integer"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="CapoSezione">
            <xsd:complexType mixed="false">
                <xsd:sequence minOccurs="1" maxOccurs="1">
                        <xsd:element name="Approvato"
                                type="xsd:boolean" default="nil"/>
                        <xsd:element name="Motivazione"
                                            type="xsd:string"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="Amministrazione">
            <xsd:complexType mixed="false">
                <xsd:sequence minOccurs="1" maxOccurs="1">
                        <xsd:element name="Approvato"
                                type="xsd:boolean" default="nil"/>
                        <xsd:element name="Capitolo"
                                                type="xsd:string"/>
                        <xsd:element name="Motivazione"
                                                type="xsd:string"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="Direttore">
            <xsd:complexType mixed="false">
                <xsd:sequence minOccurs="1" maxOccurs="1">
                        <xsd:element name="Approvato"
                                type="xsd:boolean" default="nil"/>
                        <xsd:element name="Motivazione"
                                                type="xsd:string"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

A questo punto possiamo analizzare agente per agente il documento XML che descrive il flusso e quindi definisce il ciclo di vita dell'ordine di missione.

MissionXFlow.xml

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<XFlow>
<Agent Name="Utente">
        <Receive>
                <From>Creator</From>
                <From>Merging</From>
                <From>Direttore</From>
                <From>UfficioPersonale</From>
        </Receive>
        <Action>
                <choose>
                        <when test="//Header/Sender[.='Creator']">
                                <Permission
                nodeset="//CapoSezione/*|//Amministrazione/*|
                           //Direttore/*" relevant="false()"/>
                        </when>
                        <otherwise>
        <Permission nodeset="//*"
                                                readonly="true()"/>
                        </otherwise>
                </choose>
        </Action>
        <Send>
                <choose>
                        <when test="//Header/Sender[.='Creator']">
                                <To>CapoSezione</To>
                                <To>Amministrazione</To>
```

```
                        </when>
                        <otherwise>
                        <To>End</To>
                </otherwise>
                </choose>
        </Send>
</Agent>
```

Questa parte relativa all'agente di nome Utente mette già in evidenza molti aspetti sulla semplicità e sulla chiarezza del modello per descrivere il flusso.
Si possono notare la sezione Receive con al suo interno i nomi degli agenti dai quali il documento può essere ricevuto.
Per Creator intendiamo l'agente che crea ed istanzia per primo il documento mentre con End intendiamo un agente che si occupa della terminazione del documento e di una sua eventuale archiviazione.
Nella parte Action non sono invocati dei web services ma si può notare che l'assegnazione dei permessi è diversa a seconda del momento in cui viene elaborato il documento.
Infatti se il documento è stato inviato all'utente da Creator significa che l'ordine di missione deve ancora essere compilato, quindi i campi relativi alla descrizione della missione devono essere accessibili e il documento deve poi essere inviato in parallelo al CapoSezione e all'Amministrazione.
In caso contrario se il documento è stato inviato da qualche altro agente significa che è già stato elaborato, quindi il documento può solo essere visionato dall'utente dopo di che viene archiviato dall'agente End.
L'elemento `<Sender>`, all'interno del Header, dove il sistema scrive chi ha inviato il documento, permette di eseguire il controllo sull'Agente mittente.
Vediamo ora la descrizione degli agenti CapoSezione e Amministrazione

```
<Agent Name="CapoSezione">
        <Receive>
                <From>Utente</From>
        </Receive>
        <Action>
                <Permission nodeset="//Richiedente/*|//DescrizioneMissione/*"
                                                        readonly="true()"/>
                <Permission nodeset="//Amministrazione/*"
                                                relevant="false()"/>
                <Permission nodeset="//Direttore/*"
                                                relevant="false()"/>
                <Permission nodeset="//CapoSezione/Approvato"
                                                required="true()"/>
        </Action>
        <Send>
                <To>Merging</To>
        </Send>
</Agent>


<Agent Name="Amministrazione">
        <Receive>
                <From>Utente</From>
        </Receive>
        <Action>
                <Permission nodeset="//Richiedente/*|//DescrizioneMissione/*"
                                                readonly="true()"/>
                <Permission nodeset="//CapoSezione/*"
                                                relevant="false()"/>
                <Permission nodeset="//Direttore/*"
                                                relevant="false()"/>
                <Permission nodeset="//Amministrazione/Approvato"
                                                required="true()"/>
        </Action>
        <Send>
                <To>Merging</To>
        </Send>
</Agent>
```

La descrizione di questi due agenti è simile: entrambi ricevono il documento da Utente e lo inviano all'agente che effettua la fusione.
Non ci sono istruzioni condizionali e l'unica differenza tra le due descrizioni sta nei permessi che gli agenti possiedono sui vari campi del documento.
La cosa più importante da mettere in evidenza sono le informazioni sulla visibilità: impostando relevant al valore false si possono rendere invisibili i campi che non devono essere visualizzati dagli agenti.
Il sistema una volta che l'Utente ha inviato il documento, lo duplica impostando al valore 'yes' il campo duplicate dentro l'Header dei documenti prodotti.
In questo modo gli agenti di destinazione sono informati che stanno lavorando su una copia duplicata.
L'agente di tipo Merge in seguito deve sincronizzare i due documenti.
Dopo che gli agenti Amministrazione e CapoSezione hanno inviato il proprio documento il sistema aggiunge l'elemento `<value>` ad ogni campo, specificando tramite il campo `author` l'agente che ha modificato il documento.
In questo modo l'agente Merging ha tutte le informazioni necessarie per poter fondere insieme i due documenti.
Vediamo in dettaglio come è strutturato l'agente che ha un attributo Merge con valore 'yes'.

```
<Agent Name="Merging" Merge="yes">
	<Receive>
		<From>CapoSezione</From>
		<From>Amministrazione</From>
	</Receive>
	<Action>
		<Permission nodeset="//*" readonly="true()"/>
		<Permission nodeset="//*" relevant="false()"/>
		<!-- Politiche di Merge-->
		<Select nodeset="//Amministrazione/*"
					from="Amministrazione" />
		<Select nodeset="//Caposezione/*"
					from="Caposezione" />
	</Action>
	<Send>
	<!-- Controlla se il documento e' stato approvato-->
	<choose>
		<when test="//Amministrazione[Approvato='true'] and
				//CapoSezione[Approvato='true']">
			<To>Direttore</To>
		</when>
		<otherwise>
			<To>Utente</To>
		</otherwise>
	</choose>
	</Send>
</Agent>
```

Nell'esempio vengono selezionati i campi relativi all'amministrazione dal documento inviato dall'agente Amministrazione e i campi relativi al capo sezione dal documento inviato dall'agente Caposezione.
Dopo la fusione gli elementi <value> sono eliminati e restano solo i valori.
A questo punto si possono controllare i campi Approvato per verificare se il documento è stato approvato da entrambi gli uffici e quindi recapitarlo all'agente Direttore.

```
<Agent Name="Direttore">
	<Receive>
		<From>Merging</From>
	</Receive>
	<Action>
		<Permission nodeset="//Richiedente/*|//DescrizioneMissione/*
```

```
                |//CapoSezione/*|//Amministrazione/*" Readonly="true()"/>
                <Permission nodeset="//Direttore/Approvato"
                                                required="true()"/>
        </Action>
        <Send>
                <choose>
                        <when test="//Direttore[Approvato='true']">
                                <To>UfficioPersonale</To>
                        </when>
                        <otherwise>
                                <To>Utente</To>
                        </otherwise>
                </choose>
        </Send>
</Agent>
```

Si può notare la potenza e la grande efficacia delle espressioni Xpath per assegnare il permesso di sola lettura ad un insieme di nodi.
L'ultima parte da analizzare è quello relativa all'agente UfficioPersonale dove semplicemente vengono registrati gli estremi della missione approvata.

```
<Agent Name="UfficioPersonale">
        <Receive>
                <From>Direttore</From>
        </Receive>
        <Action>
                <Permission nodeset="//*" readonly="true()"/>
                <WebServices action="Archivia" />
        </Action>
        <Send>
                <To>Utente</To>
        </Send>
</Agent>
```

L'operazione di archiviazione dei dati può essere automatizzata tramite un servizio web al quale viene passata l'istanza del documento tramite un messaggio SOAP.
Questo esempio mette in evidenza la semplicità del modello XFlow per descrivere dei flussi anche complessi.
Nella descrizione del flusso sono stati usati dei nomi generici per identificare gli agenti e i servizi web: è compito dell'architettura risolvere i nomi sostituendoli con gli indirizzi dei nodi dove risiedono fisicamente gli agenti.