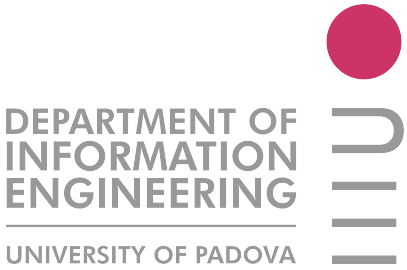


Tesi di laurea - Gestione documentale attraverso piattaforme di Enterprise Content Management e Business Process Management open-source

Laureando: Mauro Roiter, matricola 543338

Data: 22.07.2011

Relatore: Federico Filira

Indice

Elenco delle figure

Elenco delle tabelle

1 Introduzione

1.1 Realtà aziendale

Per rendere meno pesante la gestione documentale, attualmente realizzata con supporto cartaceo e solo in parte informatizzata, l'azienda vuole attuare un processo di dematerializzazione per adeguarsi alle nuove esigenze.

> *"La dematerializzazione dei documenti è il processo mediante il quale gli atti transazionali tra due o più soggetti e, in generale, quelli riguardanti la formazione di documenti rilevanti sotto il profilo giuridico, si realizzano senza altro supporto se non quello informatico e/o telematico per l'acquisizione degli elementi costitutivi l'elaborazione, l'archiviazione, il trasporto e la conservazione, con pieno valore tra le parti e verso terzi."*

L'obiettivo principale è riuscire a rappresentare i processi di business interni in maniera fedele, mantenendo tutte le informazioni ad essi associati, ossia: partecipanti e attività da essi svolte, i documenti creati e i metadati che si ricavano. Successivamente, ottenuta una sufficiente familiarità con il nuovo metodo di lavoro, si vuole estendere tale metodologia agli uffici del Comune di Venezia.
Una precisazione importante riguarda proprio la parola "dematerializzazione" che appare inappropriata, in quanto i documenti digitali sono considerati oggetti durevoli nel tempo e perciò affatto privi di materialità. Per tale motivo sarebbe forse più adeguato chiamare tale processo **digitalizzazione** oppure **informatizzazione**.
I benefici attesi dalla dematerializzazione sono:

- *miglioramento delle comunicazioni interne*: sostituendo il flusso di documenti cartacei con comunicazioni attraverso posta elettronica ordinaria o certificata;

- *miglioramento delle comunicazioni tra amministrazioni*: diminuendo notevolmente i tempi di trasporto dei documenti da un ufficio ad un altro, mantenendo traccia dei loro spostamenti grazie alla posta elettronica accoppiata alla firma digitale;

- *istituzione per la diffusione delle informazioni*: permettendo la comunicazione anche dai soggetti esterni verso l'Amministrazione, conferendo a cittadini ed imprese la possibilità di monitorare lo stato di avanzamento delle pratiche su tutti i procedimenti pubblici;
- *minore occupazione di spazio* rispetto ad archivi cartacei;
- *grandi potenzialità di riuso*.

I problemi derivanti dall'attuazione di un processo di dematerializzazione possono essere di varia natura:

- Uno tra i problemi centrali è quello della conservazione permanente delle informazioni digitali, che risulta complessa non tanto per la probabile necessità di spostamenti da un hardware ad un altro, ma per un eventuale cambio di formato dei dati (specialmente in caso si tratti di documenti firmati digitalmente).
- utilizzare archivi digitali al posto di quelli cartacei, comporta cambiamenti significativi a livello organizzativo per quanto riguarda il mantenimento e l'aggiornamento degli archivi stessi, ma anche a livello di semplificazione dei processi che producono i documenti;
- la semplificazione dei processi utilizzando intranet o portali per la comunicazione tra cittadini e amministrazioni non sono interventi semplici, non tanto per i costi ma per la complessità di progettazione e gestione dell'impatto organizzativo;
- le conoscenze e l'esperienza del personale potrebbero non essere adeguate, perciò si renderebbe utile se non necessario un periodo di formazione, introducendo nuovi costi.

1.2 Gestione dei processi

I processi aziendali sono definiti sulla base di una struttura standard costituita dalle fasi seguenti:

1. attivazione, conduzione e monitoraggio del progetto (per i punti da 2 a 7);
2. analisi;
3. valutazione di mercato;
4. risorse, tempi e costi;

5. offerta;
6. approvazione;
7. realizzazione del progetto e erogazione dei servizi;
8. chiusura del progetto.

Ciascuna di queste fasi coinvolge delle figure aziendali, le quali svolgeranno dei compiti funzionali al completamento della fase. Inoltre verranno prodotti dei documenti necessari allo svolgimento delle fasi successive.

1.2.1 Sottoprocesso di gestione dell'offerta

L'attività di studio condotta nei successivi capitoli, considera come riferimento il sottoprocesso di gestione dell'offerta. Tale processo è costituito dalle seguenti attività:

- redazione documenti protocollati;
- registrazione importo offerta (con le competenze per anno e per commessa);
- rendere esecutivi i costi preventivi;
- piano di fatturazione (fasi, importi e date di previsione);
- previsione ordini e esternalizzazioni.

La redazione dei documenti protocollati è un'attività a carico del responsabile tecnico, il quale ha il compito di redigere un fascicolo di documenti che costituiscono l'offerta. Una volta redatti, i documenti vengono passati al capo progetto che confeziona l'offerta fornendo una sintesi degli obiettivi, una valutazione economica e la definizione dei metadati. Il passo successivo è svolto dalla segreteria, che effettua una revisione formale dell'offerta, impostando il layout dei documenti e correggendo eventuali errori. Al termine della revisione il capo progetto dovrà validare il documento finale; nel caso in cui lo respinga, la segreteria dovrà correggere la revisione prima di inoltrarlo al capo progetto. A questo punto, gli organi direttivi dovranno firmare i documenti d'offerta per renderli ufficiali. In generale, se il progetto rimane sotto un certo valore economico per l'investimento (tipicamente intorno ai 20000 €), è sufficiente la firma del dirigente, altrimenti devono esserci sia la firma del direttore, sia quella del dirigente. Un'offerta ha un periodo di validità, oltre il quale si può prorogarne la validità oppure procedere alla sua cancellazione. Dopo essere stati firmati, i documenti vengono inviati al protocollo

tramite PEC (*Posta Elettronica Certificata*), perciò prima di passare alla fase successiva, bisogna attendere che il gestore del destinatario fornisca al mittente la ricevuta di avvenuta consegna con indicazione del momento esatto in cui è stata ricevuta. In questo modo il documento protocollato, ossia dotato di un numero, unico e obbligatorio, che lo identifica univocamente e ne riproduce l'ordine di arrivo nel registro di protocollo, assume valore legale.

1.3 Business Process Management

> *La Business Process Management è l'insieme di attività necessarie per definire, ottimizzare, monitorare e integrare i processi aziendali al fine di creare un processo orientato a rendere efficiente ed efficace il business dell'azienda.*

I software di BPM dovrebbero velocizzare e semplificare la gestione e il miglioramento dei processi aziendali. Per ottenere questi obiettivi, un software di BPM deve monitorare l'esecuzione dei processi, consentire ai manager di fare analisi, cambiare tecnologia o organizzazione, sulla base di dati concreti.
Una metodologia corretta per progettare ed implementare la gestione dei processi può essere così sintetizzata:

- identificazione del processo analizzato;
- definizione dei processi fornitori e dei processi client;
- definizione di input e output scambiati tra gli attori del processo;
- definizione delle attività che ne regolano lo svolgimento;
- analisi temporale;
- definizione delle prestazioni attese;
- definizione delle responsabilità.

I vantaggi introdotti dall'utilizzo di software open-source piuttosto che proprietario, riguardano il risparmio sui costi di licenza e una maggiore collaborazione da parte degli utenti. Naturalmente esistono anche degli svantaggi, in quanto, si devono valutare attentamente i rischi che comporta, poichè, solitamente, i software proprietari offrono una maggiore stabilità e perchè introducono una variazione sostanziale nel modo di

lavorare. Infatti, avere il codice sorgente liberamente modificabile, è un fattore positivo che potrebbe anche diventare negativo nel caso in cui la personalizzazione richieda tempi molto superiori a quelli stimati. L'open-source sta comunque prendendo sempre più piede, arrivando ad un livello in cui viene utilizzato non solo come strumento di lavoro, ma anche come metodo di sviluppo software e come modello di business, dal momento che il codice personalizzato o integrato con nuove funzionalità può essere ridistribuito.
In questo periodo di crisi economica, si è potuto notare che l'open-source non è solo una scelta degli utenti con un alto grado di esperienza, ma è anche una scelta dei dirigenti aziendali che hanno necessità di soddisfare esigenze di costo.

2 Alfresco

Alfresco è un software web-based e open-source che offre funzionalità di tipo ECM (Enterprise Content Management), quali:

- Document Management;
- Web Content Management;
- Record Management;
- Image Management;
- Content Management;
- Collaboration;
- Knowledge Management;

È disponibile in due versioni: Enterprise e Community. La prima, a pagamento, comprende tutte le funzionalità; la seconda, scaricabile gratuitamente dal sito `http://wiki.alfresco.com/wiki/Download_and_Install_Alfresco`, contiene solo le più importanti e la sua evoluzione dipende da una comunità di sviluppo.
Lo studio è stato effettuato sulla versione Community installata su un server di prova, supportata da un architettura composta:

- dal sistema operativo "Microsoft Windows 7";
- dal database server "MySQL";
- dal web application server "Apache Tomcat";
- dal web server "Apache HTTP Server".

Nel nucleo di Alfresco c'è un repository supportato da un server che memorizza contenuti, metadati, associazioni e indici full text. Grazie ad un insieme di interfacce che supportano diversi linguaggi e protocolli, è possibile creare applicazioni personalizzate.

Una visione globale del sistema di ECM Alfresco è data dalla figura 2.1:

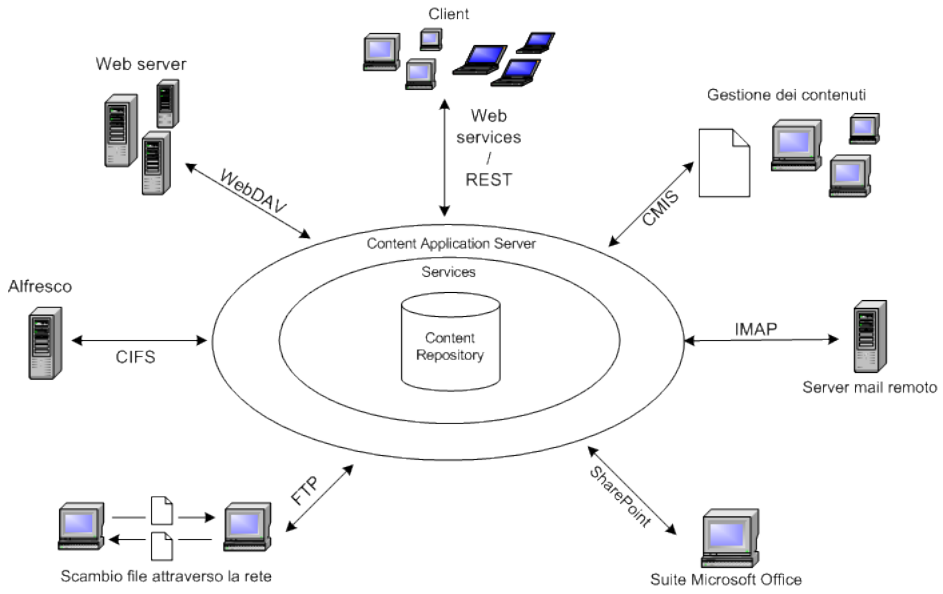


Figura 2.1: *Alfresco ECM overview*

Alfresco è sostanzialmente un'applicazione Java e perciò può essere caricato in ogni sistema che possa eseguire Java Enterprise Edition. Alla base c'è una piattaforma Spring che fornisce ad Alfresco la capacità di modularizzare alcune delle sue funzionalità, permettendo anche di aggiungerne di nuove e di creare interfacce personalizzate.
Una volta installato Alfresco, sarà possibile utilizzare due applicazioni per interagire con il repository dei contenuti, che può essere considerato come un database in grado di gestire contenuti di ogni tipo e non esclusivamente dati.
Alfresco Share e **Alfresco Explorer** sono le applicazioni in questione:

- *Alfresco Explorer*, costruito usando Java Server Faces (JSF), permette di sfogliare il repository, impostare regole e azioni, gestire contenuti e relativi metadati, associazioni e classificazioni; perciò può essere considerato come uno strumento di amminstrazione del sistema. Nonostante molte estensioni siano state costruite appositamente per Alfresco Explorer, Alfresco Share sta prendendo sempre più piede.

- *Alfresco Share*, è l'interfaccia utente costruita interamente secondo la teconologia *Web Script* di Alfresco. Può essere utilizzata sia come gestore dei contenuti (come Alfresco Explorer) ma anche per estendere l'applicazione stessa.
  Mette a disposizione una serie di strumenti di collaborazione, quali: Wiki,

Discussion, Blogs. Alfresco Share è organizzato in un insieme di siti utilizzati come "punti di incontro", per favorire la collaborazione tra gli utenti che possono accedervi, nella creazione e pubblicazione dei nuovi contenuti. È stato implementato utilizzando Spring Surf.

2.1 Architettura

Alfresco è stato creato in modo tale che la sua architettura non fosse troppo complessa, come per la maggior parte dei sistemi ECM proprietari, offrendo servizi scalabili, in grado di adattarsi all'incremento costante della mole di dati che dovrà trattare.
È stato progettato con un approccio modulare per consentire agli utilizzatori di ottenere un sistema completamente personalizzabile, a seconda delle esigenze di business, offrendo la possibilità di installare solo le funzionalità necessarie. Trattandosi di un software open-source, sarà possibile ancge modificare alcuni moduli o crearne di nuovi.
Alcune delle principali applicazioni incluse nel sistema ECM di Alfresco sono: Document Management (DM), Web Content Management (WCM), Records Management (RM), Digital Asset Management (DAM) e Search.

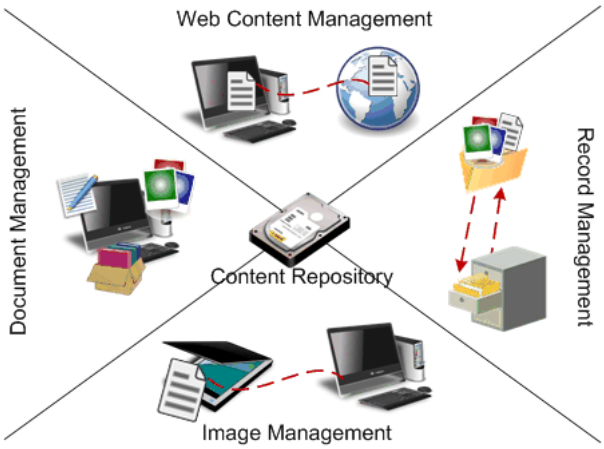


Figura 2.2: *Applicazioni incluse in Alfresco*

Gli utenti interagiscono con Alfresco come se fosse un disco di dati qualsiasi (figura 2.2), grazie alla teconologia JLAN (implementazione Java, lato server, del protocollo CIFS), con la differenza che i dati sono caricati e gestiti dal Content Application Server. Quest' ultimo comprende un Content Repository e una serie di servizi per la costruzione della soluzione ECM desiderata.
Il Content Repository, definito dagli standard CMIS (Content Management Interoperabil-

ity Services) e JCR (Java Content Repository), fornisce le specifiche per definire i contenuti e il loro immagazzinamento, il loro recupero, versionamento e gestione dei permessi di lettura/scrittura su di essi.
I servizi offerti vengono divisi in 3 categorie:

1. **Content services** (ad esempio tagging, estrazione metadati, trasformazioni, etc);
2. **Control services** (ad esempio workflow, records management, etc);
3. **Collaboration services** (ad esempio wiki, attività, etc).

L'utente comunica con il Content Application Server attraverso delle interfacce (che sono l'unica parte visibile del server) e questo comunica con i servizi attraverso vari protocolli, quali: HTTP, SOAP, CIFS, FTP, WebDAV, IMAP e Microsoft SharePoint, come si può vedere nella figura 2.3, a seconda del servizio richiesto.

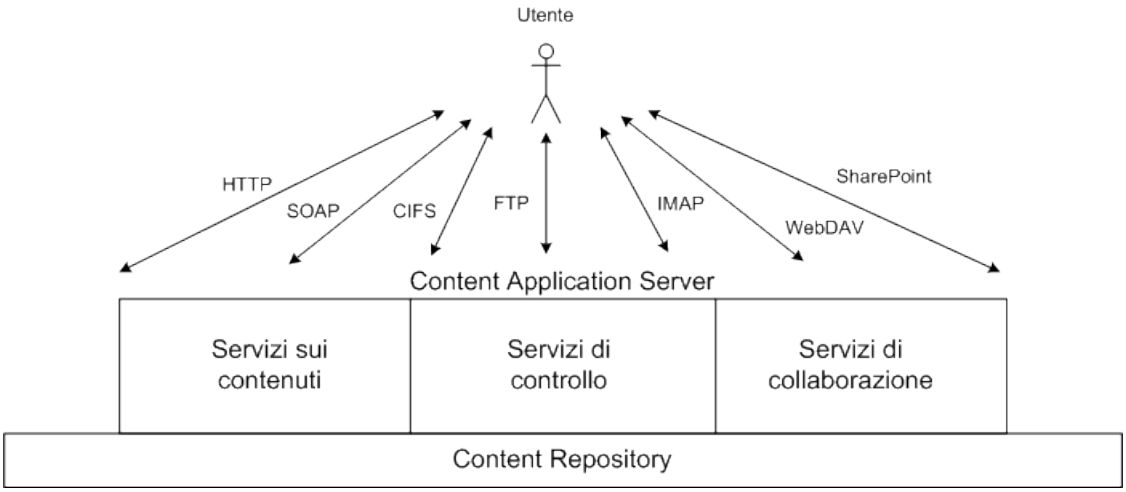


Figura 2.3: *Architettura del Content Application Server*

2.2 Document Management

Il repository di Alfresco offre trasparenza su tutti i servizi di ECM, quali:

- Virtual File System, rende l'ECM semplice come se si utilizzasse un'unità disco condivisa, con supporto del protocollo CIFS;
- navigazione, estrazione e classificazione automatica dei metadati;
- ricerca di contenuti in stile Google;

- check in/out, per il controllo delle versioni di un documento;
- auditing, per specificare chi e quando ha creato o modificato un documento;
- supporto jBPM, per la creazione di workflow associati ad un documento. Nel nostro caso si utilizzerà uno strumento esterno, Activiti Modeler, per questo scopo;
- gestione del ciclo di vita di un documento.

2.3 Web Content Management

Il WCM di Alfresco ha le seguenti caratteristiche:

- fornisce il completo controllo dei contenuti web, permettendo agli utenti di gestire contenuti di ogni tipo (documenti, immagini, video, audio, etc);
- è un ambiente di completa collaborazione per la creazione, approvazione e pubblicazione di contenuti sul web;

2.4 Record Management

Un record deve identificare un particolare stato di un dato, in un dato momento. Questa funzionalità permette di mantenere informazioni su un documento, anche nel medio-lungo periodo, registrando tutte le azioni svolte da un utente su di esso. Presenta caratteristiche di elevate disponibilità, scalabilità e fault tolerance. Dispone di:

- un sistema di acquisizione documenti di tipo drag-and-drop che supporta Microsoft Office, Outlook e Open Office, nel senso che consente di importare i documenti direttamente da questi;
- un sistema di conversione da formati Microsoft Office a ODF oppure PDF;
- estrazione e classificazione automatica dei metadati.

2.5 Content Repository

L'architettura del repository è completamente orientata al servizio. I servizi da esso forniti sono fondamentali per l'elaborazione dei contenuti:

- *gestione di file e cartelle.* Questo servizio mette a disposizione tutte i metodi per creare, leggere, aggiornare, eliminare contenuti e definire le associazioni tra essi;

- *gestione delle versioni e del check in/out*, di ogni singolo contenuto. Consente di ricavare lo storico delle versioni oppure eliminarlo, sapere qual è l'ultima versione, oppure ricaricare un contenuto ad una sua precedente versione (ottenuta dal suo storico delle versioni). Per mantenere uno storico delle versioni, bisogna assegnargli l'aspetto **versionable**.
  Il *check in/out* ha il compito di controllare gli aggiornamenti di un documento ed evitare sovrascritture dello stesso (gestisce la mutua esclusione). Quando si effettua il check out di un documento, questo viene bloccato e può essere modificato esclusivamente dall'utente o dall'applicazione che l'ha bloccato. Questa funzione può essere usata sia in contenuti "versionable" che non. Se il versionable non è attivo, quando si effettua il check in, il contenuto attuale viene sovrascritto, altrimenti viene creata una nuova versione;

- *auditing*, fornisce un record, di eventi o azioni;

- *autenticazione, autorizzazione e permessi*, fornendo metodi di creazione, eliminazione e richiesta di autorizzazioni e permessi per l'accesso o la modifica di un contenuto;

- *modeling*, servizio che registra nel repository i modelli di contenuti, le loro poprietà e associazioni, rendendoli riutilizzabili;

- *ricerca*, fornisce metodi di interrogazione del repository utilizzando diversi linguaggi: Lucene, XPath, Alfresco Full Text, CMIS QL.

2.5.1 Servizi sui contenuti

- gestione del ciclo di vita dei contenuti;

- conversione di un contenuto da un tipo ad un altro;

- estrazione dei metadati dai contenuti;

- aggiunta di tag generati dall'utente;

2.5.2 Servizi di controllo

Hanno il compito di collegare i processi ad un particolare insieme di contenuti. Sono:

- *workflow*, processi strutturati che includono una sequenza di passi concatenati, che spesso necessitano dell'interazione umana;

- *record*, descrizione dei tipi di record, politiche di archiviazione e conservazione, disposizione e reporting;
- *change set*, area di lavoro per mantenere sicure le modifiche ai contenuti;
- *preview*, anteprima dei contenuti;
- *deployment*, pubblicazione dei contenuti.

2.5.3 Servizi di collaborazione

Insieme di servizi per la produzione e pubblicazione dei contenuti nei social network:

- *social graph*, rappresentano le persone e le realzione tra esse, direttamente o indirettamente, attraverso gruppi di lavoro;
- *attività*, feed personalizzati delle attività svolte da tutti gli utenti;
- *wiki*, creazione e modifica di pagine web interconnesse;
- *blog*, traccia, regolarmente mantenuta, di commenti, eventi e altri materiali tipo documenti e video;
- *discussioni*, conversazioni in stile forum, su un particolare contenuto o sito.

2.5.4 Protocolli

I protocolli supportati consentono agli utenti di utilizzare strumenti a loro più familiari , pur maneggiando i contenuti all'interno del repository. Questi protocolli sono:

- *CIFS*, permette di vedere Alfresco come se fosse un'unità disco condivisa (JLAN);
- *WebDAV*, insieme di metodi basati sul protocollo HTTP per migliorare la gestione di documenti o file situati su web server. Supporta la mutua esclusione nell'accesso ad un dato, la creazione, rimozione e ricerca di metadati o risorse associate ad un documento.
- *FTP*, protocollo di rete standard per lo scambio e la manipolazione di file attraverso una rete;
- *IMAP*, consente l'accesso ad un server mail remoto, dal quale si possono importare le e-mail semplicemente spostandole o copiandole visualizzando anche i metadati;

- *Microsoft SharePoint*, consente ad Alfresco di agire come un server SharePoint, creando una forte interazione con la suite di Microsoft Office. Le sue funzioni di collaborazione sono tutte mappate ad Alfresco.

2.5.5 API

2.5.5.1 API Remote

Utilizzate dai client per comunicare in maniera remota con il Content Application Server di Alfresco. Si dividono in 2 categorie: **Web services API** e **RESTful API**. Le prime, basate sul protocollo SOAP, sono utilizzate per dialogare con interfacce scritte ad esempio in Java; le seconde, basate sul protocollo HTTP, sono utilizzate sono utilizzate per interagire con tutti i servizi offerti da Alfresco. Con le API RESTful si potrà implementare l'interazione tra i business process ed eventuali web service.

2.5.5.2 API Embedded

Utilizzate per sviluppare estensioni dei servizi offerti dal Content Application Server, spesso dipendenti da servizi preesistenti. Ce ne sono di diversi tipi a seconda delle esigenze, per esempio:

- **JCR - Java Content Repository**, insieme di interfacce Java per l'interazione con il Content Repository;
- **JavaScript API**, versione object-oriented delle Java Foundation API create appositamente per l'utilizzo in JavaScript;
- **Definizione workflow**, API per la definizione dei business process.

3 Alfresco Share

Alfresco Share consente l'organizzazione di documenti, record, contenuti web ed attività inerenti vari progetti. Si concentra soprattutto sugli aspetti di collaborazione nella gestione dei contenuti, introducendo caratteristiche tipo anteprime web, tag e social networks. Il concetto principale in Alfresco Share è la nozione di sito, ovvero il luogo dove gli utenti collaborano alla produzione e pubblicazione di contenuti. È implementato in Spring Surf ed è personalizzabile anche senza avere conoscenze di JSF. Inoltre, può essere distribuito separatamente dal Content Applciation Server, e gestito dal modulo di WCM di Alfresco.

3.1 Siti

Dopo che un utente ha effettuato l'accesso, inserendo i propri nome utente e password, accederà ad una pagina detta pannello di controllo. Questa pagina è completamente personalizzabile sia dal punto di vista del layout, sia dal punto di vista delle componenti, che ogni utente può scegliere, a seconda di quali siano quelle di maggior interesse, tramite un apposito strumento. A partire dal pannello di controllo l'utente può creare dei siti, ovvero pagine all'interno delle quali possono essere raggruppati altri utenti (membri) legati, ad esempio, da un progetto di lavoro. I siti hanno appunto lo scopo di riunire quegli utenti che sono coinvolti dalle informazioni contenute al loro interno. Ciascun utente può essere membro di più siti. Ogni sito può essere pubblico o privato, e conterrà un insieme di documenti, a loro volta mantenuti in un repository, nel quale risiedono tutte le informazioni riguardo i contenuti di tutti i siti e di Alfresco Share in generale.

3.2 Wiki

Strumento che consente di creare pagine web per un sito. Ogni utente che può accedere al sito, può contribuire allo sviluppo oppure modificare il contenuto delle pagine, utilizzando un linguaggio di markup semplificato.

3.3 Blog

Sezione dove si possono aggiungere commenti, descrizioni di eventi ed altro materiale, tipo documenti e video, relativo ad un sito. Tutti i membri del sito possono postare contenuti nel blog, con la possibilità di registrare inizialmente il post solo come bozza e successivamente, quando il post è pronto, pubblicarlo nel blog. Un post può essere pubblicato anche in un blog esterno al sito di appartenenza del membro che l'ha creato. A ciascun post gli utenti possono aggiungere commenti per rendere il blog più interattivo, ma solo se si tratta di utenti membri del sito. Il resto degli utenti, che hanno accesso al sito ma non ne sono membri, possono solo vedere la conversazione.

3.4 Link

Strumento utilizzabile dai membri di un sito per creare una lista di link ad altri siti interni ad Alfresco Share oppure a qualunque altro indirizzo web, che sono correlati con il sito in questione. Inoltre i membri di un sito possono commentare la pubblicazione di un link.

3.5 Discussioni

Sezione in cui un utente del team di progetto (gruppo di membri di un sito) può avviare una discussione, che sarà gestita come se avvenisse all'interno di un forum, o per postare contenuti relativi al sito, generati dagli utenti.

3.6 Calendario

Strumento utile per mantenere un'agenda degli eventi assegnando a ciascuno di questi una descrizione, data e ora di inizio e fine, tag relativi a qualsiasi tipo di contenuto del sito e eventuali documenti utili.

3.7 Elenchi dati

Strumento che permette ai membri di un sito di creare e gestire elenchi di dati rilevanti per il sito. Gli elenchi possibili sono:

- agenda eventi;
- agenda riunioni;

- elenco di cose da fare;
- elenco di compiti (semplice);
- elenco di compiti (avanzato);
- elenco di contatti;
- elenco di eventi;
- elenco di località;
- elenco di problemi.

3.8 Gestione documento

La funzionalità principale, è sicuramente la capacità di gestione di un documento unita a tutte le informazioni ad esso associate, quali:

- utente che lo crea;
- utenti che lo leggono e/o lo modificano;
- gestione dei metadati ad esso associati;
- permessi di accesso;
- gestione del suo ciclo di vita;
- gestione degli aspetti.

Una aspetto importante è che mentre un utente modifica un documento, nessun altro utente può farlo, ossia il documento viene bloccato per quell'intervallo di tempo (mutua esclusione).

3.8.1 Permessi

L'utente che inserisce un documento nel repository, può definire i permessi di accesso a seconda che l'utente sia un collaboratore, consumatore, contribuente o un semplice utilizzatore.

3.8.2 Versioni

Uno degli aspetti fondamentali caratterizzanti un documento, è la gestione del suo ciclo di vita. Dopo che un utente ha caricato un documento, questo potrebbe subire delle modifiche di vario tipo. Per non perdere le informazioni modificate, Alfresco tiene traccia delle versioni del documento precedenti la modifica con la possibilità anche di ripristinarle, se necessario, mantendo anche le informazioni riguardanti l'utente che esegue le modifiche e quando le effettua.

3.8.3 Workflow

Ad ogni documento è possibile assegnare un workflow, che ne descrive il flusso, dichiarando quale utente dovrà esaminarlo, se e quale compito dovrà svolgere, entro quando e con quale priorità. Un workflow può interessare anche più di un utente; in questo caso si deve specificare anche quale percentuale di utenti deve approvare il documento per portare a termine il workflow. Si può anche impostare che il successivo utente che dovrà esaminare un dato documento, riceva un notifica quando il suo predecessore ha eseguito i suoi compiti. Questa funzionalità è senza dubbio quella fondamentale per la rappresentazione e gestione dei business process aziendali.

3.8.4 Metadati

I metadati sono per definizione i dati che descrivono i dati. Ad ogni documento è possibile associare i metadati, utilizzabili poi come chiave di ricerca, e modificarli anche nel corso del tempo se ad esempio il documento dovesse subire delle modifiche sostanziali. La modifica di un metadato provoca la creazione di una nuova versione del relativo documento.

3.8.5 Aspetti

Ad un documento possono essere assegnati vari aspetti, che sono delle proprietà che si vuole il documento abbia. Ad esempio, un documento non manterrà traccia delle varie versioni, se non gli viene assegnato l'aspetto versionable, oppure non potranno essere definiti dei tag se non gli viene assegnato l'aspetto taggable.

3.8.6 Tag

I tag sono parole chiave associate ad un documento, utili per effettuare ricerche, soprattutto se non si conosce il nome con cui il documento è stato salvato nel repository.

4 Activiti BPMN 2.0

Activiti BPMN 2.0 è il motore di gestione dei processi di business su cui si basa Alfresco. È una piattaforma di Business Process Management open-source, su licenza Apache, progettata per implementare il nuovo standard BPMN 2.0 dell'OMG (Object Management Group). L'ultima versione può essere scaricata liberamente dal sito: `www.activiti.org/download`.
La versione utilizzata per i test è la 5.4, anche se l'evoluzione del software è rapida e continua, infatti in pochi mesi si è già arrivati alla release 5.5.
L'installazione del software è un processo molto articolato poichè di default, viene fornito un file di configurazione che installa tutto quello di cui si ha bisogno per lavorare, ovvero un database server (H2), dove verrà caricato il database di riferimento e tutte le tabelle necessarie, ed un web application server (Apache Tomcat 6.0.32), dove verranno caricate tutte le web application (Activiti Cycle, Activiti Probe, Activiti Explorer, Activiti Modeler), presupponendo che l'utente non possieda già tali strumenti. Nel nostro caso si vogliono utilizzare il database server MySQL e l'ultima versione di Tomcat, la 7.0.12, già installati correttamente sulla macchina. Ciò comporta una serie di modifiche ai file di configurazione:

- nel file di configurazione principale, *build.properties*, si dovranno modificare:
  - il valore della variabile "*db*", impostandolo a "*mysql*";
  - il valore della variabile "*tomcat.version*", impostandolo al valore corrispondente alla versione di Tomcat che si vuole utilizzare (nel nostro caso 7.0.12);
  - il valore della variabile "*downloads.dir*" impostandola alla directory dove è stata scaricata la versione di Tomcat di riferimento;
  - aggiungere due variabili per specificare quali saranno la directory in cui verranno posizionate le web applications e la directory del repository (nel nostro caso: *dir-tomcat-webapps=C:/Tomcat7/webapps/* e *fileSystemRootDirectory=C:/repository*).

- nel file *build.xml* (il file di riferimento per l'installazione), si dovranno cancellare le parti relative a download e installazione del DB server H2, e di Tomcat facendo attenzione a non eliminare parti necessarie al funzionamento di Activiti 5.5. In particolare si dovranno modificare:

  - la variabile "*activiti.home*" impostando come valore la directory di installazione di Activiti (nel nostro caso "*C:/activiti-5.5*");

  - eliminare la variabile "*tomcat.url*";

  - nella sezione <!– DEMO –> si dovranno eliminare tutti i riferimenti alle installazioni di H2 e Tomcat. Devono rimanere però i riferimenti alla creazione e popolazione del DB di supporto ad Activiti, ovvero "*db.create*" e "*db.demo.data*". Da notare quindi che la successiva sezione <!– DB –>, che si occupa appunto di questo non deve essere toccata;

  - la sezione <!– DB –> relativa all'installazione di H2 può essere intermante eliminata;

  - nella sezione <!– TOMCAT –> di deve modificare la variabile "*tomcat.home*" impostando come valore la directory di installazione di Tomcat (nel nostro caso "*C:/Tomcat7*");

  - opzionalmente si può eliminare la sezione <!– OPEN BROWSER –> in quanto superflua;

Dopo aver effettuato tali modifiche è possibile installare Activiti, aprendo un prompt dei comandi, andando alla directory setup di Activiti e lanciando il comando "*ant demo.start*". Ovviamente per eseguire il suddetto comando, si deve installare e configurare propriamente Ant. Al termine, se l'installazione è avvenuta con successo si vedrà un messaggio "BUILD SUCCESSFUL".
A questo punto si dovrebbero trovare all'interno del Tomcat Manager tutte le webapps di Activiti installate e avviate.

Cliccando ad esempio su Activiti Probe si dovrebbe presentare la pagina di login:

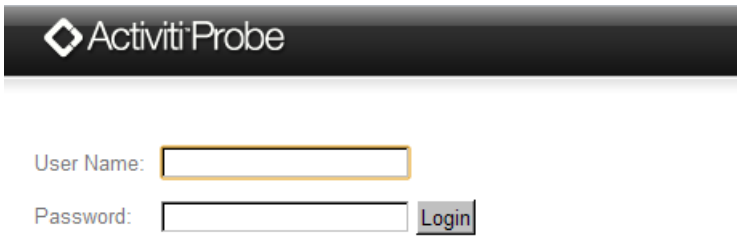


Figura 4.1: *Activiti Probe Login*

4.1 Componenti

Activiti è stato progettato in modo che risultasse semplice da usare per gli sviluppatori Java e che non fosse troppo pesante da eseguire.

4.1.1 Activiti Engine

È il nucleo di Activiti. Si tratta di un motore che esegue processi, definiti in BPMN 2.0, localmente. Ha le seguenti caratteristiche:

- funziona su qualsiasi ambiente Java (Spring, JTA) indipendente;
- semplice da installare ed eseguire;
- costruito per supportare la scalabilità del sistema;
- possibilità di aggiungere tipi di attività personalizzati, così da rappresentare la realtà aziendale il più fedelmente possibile, e di usare un linguaggio di definizione dei processi dedicato;
- velocità d'esecuzione;
- utilizza listener di eventi, in modo da nascondere i dettagli impementativi dai diagrammi di livello business;
- consente di testare l'esecuzione dei processi attravareso un Java Unit Test.

Alla base dell'architettura dell'Activiti Engine c'è la Process Virtual Machine che permette l'introduzione di nuovi tipi di attività, caratteristiche e linguaggi di definizione di processo. Il linguaggio BPMN 2.0 è ideato per ridurrre il gap tra progettazione e gestione dell'esecuzione dei processi di business.
Nella figura 4.2 viene rappresentato come Activiti è collegato ad Alfresco:

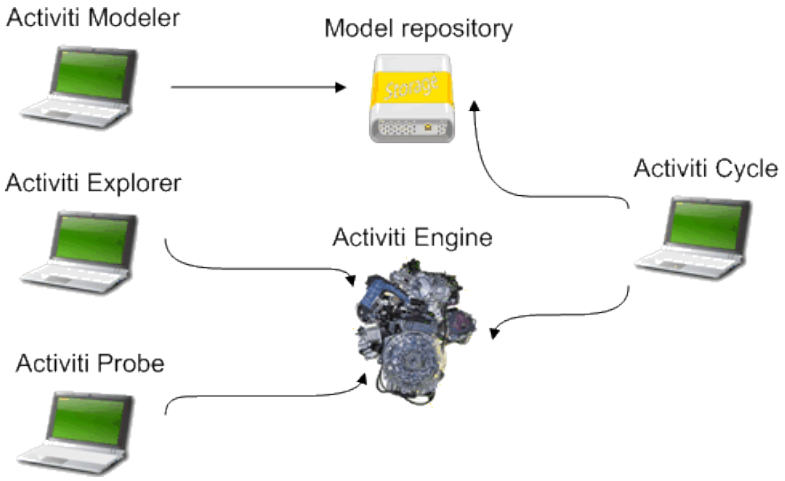


Figura 4.2: *Panoramica dell'architettura di Activiti*

4.1.2 Activiti Explorer

Activiti Explorer è un'applicazione web che fornisce la possibilità a tutti gli utenti di sfruttare l'Activiti Engine. Ha funzionalità di gestione dei task, che può essere vista secondo diversi punti di vista:

- *utente*, che può visualizzare la lista dei task che gli sono stati assegnati, oppure quelli per cui è candidato. Potrà inoltre completare un task, compilando anche dei form (se richiesto);
- *manager*, che può visualizzare le liste di task dei dipendenti;
- *amministratore*, che potrà verificare i dettagli dell'istanza di processo collegata ad un task.

4.1.3 Activiti Probe

Activiti Prove è un'applicazione web che fornisce

- capacità di amministrazione e monitoraggio di un processo in esecuzione,

- strumenti per caricare un'istanza di processo e renderla eseguibile.

Questa applicazione è rivolta agli amministatori e agli operatori di sistema, i quali hanno il compito di mantenere il sistema funzionante, offrendo anche la capacità di trarre informazioni importanti, come il tempo di esecuzione di un task o di un processo, a livello di business.
I casi d'uso sono:

- controllare se l'Activiti Engine è in esecuzione o se c'è un problema da qualche parte;
- gestione dei deployment;
- visualizzare le risorse in esecuzione;
- gestione delle definizioni di processo;
- controllare le tabelle del database;
- visualizzare la durata di una transazione;

4.1.4 Activiti Modeler

Activiti Modeler è un editor di processi web based, utilizzato per modellare processi in modo grafico e per definire proprietà o attributi per ogni elemento del modello, che permette anche l'esportazione dei modelli in altri editor.
I file di processo vengono memorizzati dal server in una directory, specificata dall'utente in fase di installazione, detta *model repository*.

4.1.5 Activiti Cycle

Activiti Cycle è un'applicazione web based che migliora la collaborazione tra tutto il personale aziendale, dal dirigente allo sviluppatore. Permette di ottenere il codice BPMN 2.0, relativo ai processi creati con Activiti Modeler.
Lavorare sul codice invece che in forma grafica è senz'altro più conveniente, in quanto si acquisisce familiarità con il linguaggio e, in caso di errori, è più semplice individuarli e correggerli, ma soprattutto si ottiene un codice più efficiente.
Activiti Cycle è basato sul concetto di Process Cycle Layer.

Process Cycle Layer Il Process Cycle Layer è un concetto di BPM che identifica il livello di collaborazione tra gli attori in un processo di business, tenendo presente il fatto che l'automatizzazione integrale di un processo di business, che porta alla creazione di un insieme di documenti, non è realizzabile. Si deve perciò mantenere una distinzione tra i vari tipi di documento, associandoli a chi li crea. L'obiettivo è dunque migliorare la collaborazione tra i vari gruppi di lavoro, fornendo loro la possibilità di avviare discussioni o pubblicare link per aggiungere informazione sull'argomento trattato.

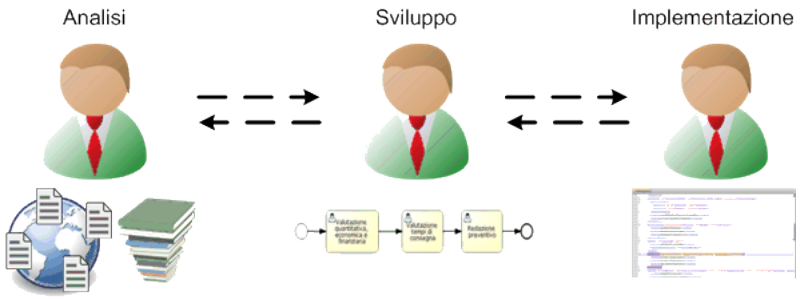


Figura 4.3: *Process Cycle layer*

Activiti Cycle è stato creato proprio per raggiungere questo obiettivo. Il collegamento tra il modello di processo, creato con il Modeler, nel model repository e il processo eseguibile sull'Activiti Engine, monitorabile con Probe ed Explorer, è il punto chiave.

4.2 BPMN 2.0

Il ***Business Process Modeling and Notation*** è uno standard di modellazione dei business process che ha l'obiettivo di fornire una notazione comprensibile a tutti gli utenti, dall'analista che progetta il processo, agli sviluppatori che implementano la tecnologia in grado di eseguire il processo, alle persone che gestiranno il processo. Inoltre assicura che i linguaggi XML, progettati per l'esecuzione dei processi di business, possano essere visualizzati secondo una notazione business-oriented, ovvero definisce un Business Process Diagram (BPD), che rappresenta un adattamento della tecnica di stesura di diagrammi di flusso alla descrizione dei processi di business.

Lo standard BPMN fornisce alle aziende la capacità di rappresentare le loro procedure di business attraverso una notazione grafica e conferendo loro la possibilità di comunicare tali procedure in maniera standardizzata, facilitando l'interoperabilità con l'utente. È anche una notazione che permette di creare definizioni di processo portabili, cosicchè l'utente possa usarle anche in ambienti diversi da quello in cui le aveva create.

4.2.1 Elementi definiti nella BPMN

Gli elementi che si utilizzano per modellare i business process vengono classificati in in 5 categorie di base:

1. *Flow objects*;
2. *Data*;
3. *Connecting objects*;
4. *Swimlanes*;
5. *Artifacts*.

I *Flow objects* sono gli elementi principali per definire il comportamento di un processo di business. Ce ne sono di 3 tipi:

- **Events**, indicano qualcosa che accade durante il corso del processo, influenzando il flusso del processo stesso, e solitamente hanno una causa (trigger) e un impatto (result). Gli eventi sono di 3 tipi a seconda del punto in cui influenzano il flusso: start, intermediate, end.

Evento	Descrizione	Simbolo
Start event	indica il punto in cui il processo ha inizio	
Start message event	il processo verrà avviato nel momento in cui si riceve un messaggio	
Start timer event	il processo partirà alla scadenza di un timer, ad un dato istante oppure dopo un certo periodo di tempo	
Start signal event	il processo verrà eseguito in base alla ricezione di un segnale, lanciato da un processo esterno, che può essere rilevato anche più di una volta	
Start multiple event	indica che il processo ha inizio al verificarsi di uno tra un insieme di possibili eventi	
Start parallel multiple event	indica che il processo inizia al verificarsi di tutti gli eventi attesi	
Start escalation event	utilizzato solo all'interno di sottoprocessi, reagisce all'escalation a un altro ruolo dell'organizzazione	
Start error event	utilizzato solo all'interno di sottoprocessi fà partire il processo al verificarsi di un predefinito errore	
Start conditional event	il processo inizia al verificarsi di una data condizione	
Start compensation event	utilizzato solo all'interno di sottoprocessi gestisce l'arrivo di una "compensation"	

Tabella 4.1: *Rappresentazione grafica di flow object di tipo* ***Start Event***

Evento	**Descrizione**	**Simbolo**	
		Catching	**Throwing**
Intermediate event	indica l'occorrenza di un particolare evento che non ritarda l'esecuzione del processo 		
Intermediate message event	Reagisce all'arrivo di un messaggio		
Intermediate timer event	Rimanda l'esecuzione del processo ad un determinato momento oppure allo scadere di un timeout		
Intermediate escalation event	Deve essere attaccato al bordo di un'attività e reagisce all'escalation di un particolare caso		
Intermediate conditional event	L'esecuzione del processo è ritardata fino al verificarsi di una data condizione		
Intermediate link event	Porta direttamente ad un certo punto del processo. Due link corrispondenti equivalgono ad un sequence flow		
Intermediate error event	Dev'essere attaccato al bordo di un'attività. Reagisce al verificarsi di un errore provocato da un sottoprocesso		
Intermediate cancel event	Dev'essere attaccato al bordo di un'attività. Reagisce al verificarsi di una transazione interna ad un sottoprocesso		
Intermediate compensation event	Dev'essere attaccato al bordo di un'attività. Funge da compensazione in caso di parziale fallimento di un'operazione		
Intermediate signal event	Ritarda l'esecuzione del processo fino all'arrivo di un segnale		
Intermediate multiple event	Ritarda l'esecuzione del processo fino a che uno dei possibili eventi che possono essere causati non si verifica		
Intermediate parallel multiple event	Ritarda l'esecuzione del processo finchè tutti i possibili eventi non si verificano		

Tabella 4.2: *Rappresentazione grafica di flow object di tipo* ***Intermediate Event***

Come si può vedere nella tabella 4.2, ci sono due tipi di *intermediate event*, quelli con un trigger di tipo **catch** e quelli con un trigger di tipo **throw**. Gli intermediate events di tipo throw lanciano un segnale corrispondente all'evento che si è verificato, mentre quelli di tipo catch catturano questo segnale e lo gestiscono in modo appropriato.

Evento	**Descrizione**	**Simbolo**
End event	indica la fine del processo	
End message event	alla fine del processo viene inviato un messaggio di notifica	
End error event	il processo termina in maniera errata, inviando un particolare errore che dev'essere trattato	
End cancel event	il processo termina cancellando una particolare transazione attiva	
End compensation event	alla fine del processo viene eseguita una particolare sequenza di task come compensazione	
End escalation event	alla fine del processo viene eseguita una particolare sezione del processo	
End multiple event	Alla fine del processo viene eseguito uno tra un insieme di possibili eventi	
End signal event	Alla fine del processo viene lanciato un segnale che dev'essere catturato	
End terminate event	indica la terminazione immediata del processo, interrompendo tutti i task ancora in esecuzione	

Tabella 4.3: *Rappresentazione grafica di flow object di tipo* ***End Event***

- **Activities**, indicano un compito che l'azienda deve svolgere all'interno del processo. Può essere atomica o complessa, ossia può essere un task oppure un sottoprocesso.

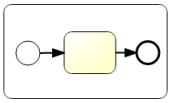
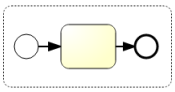
Attività	Descrizione	Simbolo
Task	Rappresenta l'attività da svolgere	
Collapsed subprocess	Rappresenta un'attività che può essere scomposta in più attività. Può essere collegata ad un altro diagramma di processo	
Expanded subprocess	Rappresenta un'attività decomponibile, e deve contenere un diagramma BPMN valido	
Collapsed event-subprocess	Attività decomponibile che deve essere inserita al'interno di un sottoprocesso. Si attiva quando il suo start event verifica la condizione. Può essere eseguito in parallelo al sottoprocesso in cui è contenuto oppure può interromperlo fino al termine della sua esecuzione. Può riferirsi ad un altro diagramma di processo	
Event-subprocess	Attività decomponibile che deve essere inserita al'interno di un sottoprocesso. Si attiva quando il suo start event verifica la condizione. Può essere eseguito in parallelo al sottoprocesso in cui è contenuto oppure può interromperlo fino al termine della sua esecuzione.	

Tabella 4.4: *Rappresentazione grafica di flow object di tipo* ***Activity***

In particolare l'attività di tipo Task, può essere di diversi tipi:

- **Send Task**, è un task che invia un messaggio ad un *Participant* esterno. Un Participant è ciò che viene rappresentato mediante l'oggetto Pool della categoria Swimlanes;
- **Receive Task**, è un task che attende la ricezione di un messaggio in arrivo da un Participant esterno;
- **User Task**, è un task che prevede l'interazione umana attraverso una qualche applicazione software;

  - **Manual Task**, è un task che viene eseguito senza l'aiuto di alcun motore d'esecuzione di business process o applicazione software;
  - **Service Task**, è un task che utilizza un qualche servizio (ad esempio un web service o una qualche applicazione automatizzata);
  - **Business Rule Task**, è un task che fornisce al processo un meccanismo per passare dei dati in input al motore d'esecuzione dei business process, ed ottenere dei dati in output;
  - **Script Task**, è un task che viene eseguito dal motore d'esecuzione dei business process, il quale dev'essere in grado di interpretare il linguaggio con il quale lo script è stato definito.

- **Gateways**, sono usati per controllare le divergenze e le convergenze del flusso del processo. Eventuali marker interni al simbolo indicano particolari comportamenti di controllo.

Gateway	Descrizione	Simbolo
Exclusive gateway	Instrada il flusso sul sequence flow che verifica la condizione d'uscita, quando divide ha funzione di splitting del flusso. Quando ha funzione di merging, attende un sequence flow in ingresso prima di far proseguire il flusso	
Event-based gateway	È sempre seguito da un catch event o da un receive task. Il flusso viene instradato verso il primo evento o task che si verifica.	
Parallel gateway	Quando viene usato per dividere il flusso, tutti i sequence flow d'uscita vengono attivati simultaneamente. Quando è usato per riunire più flussi, attende che tutti questi siano arrivati.	
Inclusive gateway	Quando viene usato per dividere il flusso, attiva tutti i sequence flow che soddisfano la loro condizione. Quando ha funzione di merging attende l'arrivo di tutti i sequence flow attivi.	
Complex gateway	Attiva uno o più flussi sulla base di condizioni complesse o descrizioni verbali.	

Tabella 4.5: *Rappresentazione grafica di flow object di tipo* ***Gateway***

La categoria *Data* è composta dagli elementi:

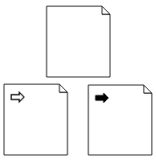
Data	Descrizione	Simbolo
Data object	può essere di due tipi: *Data Input* o *Data Output*. Rappresenta l'informazione che circola durante il processo, come ad esempio documenti o e-mail	
Data store	rappresenta il luogo dove il processo può leggere o scrivere dati (ad esempio un database). La sua esistenza non è limitata al periodo di esecuzione del processo	
Message	viene utilizzato per rappresentare il contenuto di una comunicazione tra due *Participant*	

Tabella 4.6: *Tabella degli elementi di tipo Data*

I *Connecting objects* rappresentano tutti i differenti modi in cui i *Flow objects* sono collegabili tra loro:

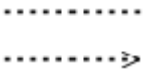
Object	Descrizione	Simbolo
Sequence flow	definisce l'ordine di esecuzione delle attività	
Message flow	utilizzato per mostrare il flusso di messaggi tra due partecipanti abilitati per l'invio e la ricezione di messaggi. In BPMN, due pool separati in un diagramma rappresentano due partecipanti	
Association	utilizzata per collegare un'informazione ad un Artifact. Le Text Annotation e altri Artifact possono essere associati con elementi grafici (ad esempio Task). L'Association, quando è appropriato, può essere indicata con una freccia per esprimere un flusso direzionale	

Tabella 4.7: *Tabella dei Connecting Object*

Per raggruppare gli elementi precedenti si utilizzano le *Swimlanes*, che possono essere:

Swimlane	Descrizione	Simbolo
Pool	un pool è la rappresentazione grafica di un partecipante ad una collaborazione. Agisce anche da "swimlane" e da contenitore grafico per partizionamenti di insiemi di attività provenienti da altre pool (in contesto di situazioni B2B)	
Lane	è una sottopartizione interna a un processo, qualche volta interna ad un pool, utilizzata per organizzare e categorizzare le attività	

Tabella 4.8: *Tabella degli Swimlane*

Infine gli *Artifacts* vengono utilizzati per rappresentare informazione aggiuntive:

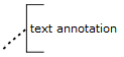
Artifact	Descrizione	Simbolo
Group	è un raggruppamento di attività appartenenti alla stessa categoria.Un Group non influisce sul flusso del processo. Vengono utilizzati soprattutto per visualizzare graficamente le categorizzazioni di attività in un diagramma	
Text Annotation	utilizzata per fornire, a chi legge il diagramma, informazioni aggiuntive. Possono essere associate ad un Association	

Tabella 4.9: *Tabella degli Artifact*

4.2.2 Confronto con altri standard

Business Process Modeling and Notation è senz'altro lo standard più efficiente. Quello che più gli si avvicina è il diagramma delle attività UML, che utilizza dei concetti molto simili (in alcuni casi differenti solo nel nome). In generale però la simbologia BPMN è più leggibile e facilmente comprensibile anche da utenti non esperti proprio perchè pensata appositamente per rappresentare processi aziendali.

5 Implementazione

La parte pratica di questo progetto riguarda l'implementazione in BPMN 2.0 del processo di gestione dell'offerta, descritto nel paragrafo 1.2.1, e la verifica del suo corretto funzionamento. Innanzitutto è stata affrontata una fase di raccolta dei requisiti, per effettuare una modellazione fedele del processo, mediante una serie di interviste con il Tutor; successivamente è stata creata una rappresentazione grafica del processo utilizzando Activiti Modeler; dopodichè, attraverso Activiti Cycle, si è ricavato il codice XML (BPMN 2.0) relativo al sottoprocesso stesso; in conclusione, sono state effettuate tutte le personalizzazioni necessarie per implementare alcuni dettagli che per via grafica non si riuscirebbero a rappresentare. Inoltre lavorando sul codice si comprende meglio il linguaggio e il funzionamento di certe proprietà da includere in alcuni oggetti. Il sottoprocesso coinvolge:

- diversi attori, che in BPMN 2.0 vengono modellati con l'oggetto *lane* e raggruppati nell'oggetto *pool* che rappresenterà l'organizzazione che li comprende;
- attività semplici, da assegnare all'attore che le dovrà eseguire, modellate con oggetti di tipo *task*;
- attività complesse, costituite da più attività semplici, delle quali si vogliono nascondere i dettagli implementativi, modellate con l'oggetto *call activity*;
- eventi, che stabiliscono il flusso del processo, modellati con l'oggetto *exclusive gateway*;
- stati del processo, in cui il normale flusso viene suddiviso in più flussi, paralleli o meno, e successivamente si riuniscono a formare nuovamente il flusso prinicipale, modellati tramite l'oggetto *parallel gateway*.
- eventi, che segnano l'inzio e la fine del processo, modellati dagli oggetti *start event*, *end event*, e più un particolare *end event* all'interno dei quali si dichiara la proprietà *terminateEventDefinition,* la quale indica al BPMN engine che nel caso di raggiungimento di questo particoalre evento, qualsiasi altra attività ancora in esecuzione viene terminata insieme al processo stesso;

- l'ordine delle attività, rappresentato mediante l'oggetto *sequence flow*.

È possibile dettagliare ulteriormente il processo includendo nella modellazione i documenti che vengono prodotti dopo ogni attività, così da rappresentare anche il flusso documentale oltre a quello dei compiti da svolgere per portare a termine il processo, ed eventuali messaggi che vengono scambiati tra due o più partecipanti (pool) al processo. In questo caso si dovranno utilizzare l'oggetto *association* per collegare un documento, identificato dall'oggeto *data object*, al relativo task, gli oggetti *message* e *message flow*, rispettivamente per collegare i due partecipanti che dialogano tra loro e per rappresentare un scambio di messaggi, ed infine l'oggetto *text annotation* per inserire informazioni aggiuntive sull'oggetto a cui viene associato.
I seguenti codici sono stati scritti basandosi sulla configurazione demo di Activiti. Ciò significa che l'installazione crea un database contenente:

- 3 utenti di prova, Kermit, Gonzo e Fozzie:

UserId	**Password**	**Security roles**
kermit	kermit	admin
gonzo	gonzo	manager
fozzie	fozzie	user

- 7 gruppi di prova i cui id sono: accountancy, admin, engineering, management, manager, sales e user.

- il popolamento dei gruppi definito nel seguente modo:

UserId	**GroupId**
fozzie	accountancy
fozzie	user
gonzo	accountancy
gonzo	management
gonzo	manager
gonzo	sales
kermit	accountancy
kermit	admin
kermit	engineering
kermit	management
kermit	manager
kermit	sales

5.1 Processo di gestione dell'offerta

```
<?xml version="1.0" encoding="UTF-8"?>
  <definitions xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
      xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
      xmlns:omgdc="http://www.omg.org/spec/DD/20100524/DC"
      xmlns:omgdi="http://www.omg.org/spec/DD/20100524/DI"
      xmlns:signavio="http://www.signavio.com"
      xmlns:activiti="http://activiti.org/bpmn"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      id="sid-a83411fe-e5e1-49be-bfc1-bb5b9802132e"
      targetNamespace="http://activiti.org/bpmn20"
      typeLanguage="http://www.w3.org/2001/XMLSchema"
      xsi:schemaLocation="http://www.omg.org/spec/BPMN/20100524/MODEL
          http://www.omg.org/spec/BPMN/2.0/20100501/BPMN20.xsd">
      <process id="offerManagement" name="Processo di gestione dell'offerta">
        <laneSet id="offerManagementProcess">
          <lane id="managerLane" name="Direzione">
            <flowNodeRef>assResp</flowNodeRef>
            <flowNodeRef>signOfferSubProcess</flowNodeRef>
            <flowNodeRef>protocolSignedOffer</flowNodeRef>
            <flowNodeRef>exGtwAcpt</flowNodeRef>
            <flowNodeRef>terminateEnd</flowNodeRef>
          </lane>
          <lane id="adminLane" name="Amministrazione">
            <flowNodeRef>adminSubProcess</flowNodeRef>
          </lane>
          <lane id="areaLane" name="Settore">
            <flowNodeRef>claimProject</flowNodeRef>
            <flowNodeRef>analisysSubProcess</flowNodeRef>
            <flowNodeRef>verifyAnalisys</flowNodeRef>
            <flowNodeRef>exGtwForniture</flowNodeRef>
            <flowNodeRef>offerReview</flowNodeRef>
            <flowNodeRef>developmentSubProcess</flowNodeRef>
          </lane>
          <lane id="secretaryLane" name="Segreteria">
            <flowNodeRef>startOfferProcess</flowNodeRef>
            <flowNodeRef>acquireDocument</flowNodeRef>
            <flowNodeRef>exGtwIncomingReq</flowNodeRef>
            <flowNodeRef>protocolReqSubProcess</flowNodeRef>
            <flowNodeRef>setLayoutSubProcess</flowNodeRef>
            <flowNodeRef>sendOffer</flowNodeRef>
            <flowNodeRef>internalDistribution</flowNodeRef>
            <flowNodeRef>exGtwSupply</flowNodeRef>
            <flowNodeRef>parGtwDiv1</flowNodeRef>
```

```
        <flowNodeRef>parGtwConv</flowNodeRef>
        <flowNodeRef>endOfferProcess</flowNodeRef>
        <flowNodeRef>parGtwDiv2</flowNodeRef>
      </lane>
      <lane id="officeLane" name="Ufficio acquisti">
        <flowNodeRef>supplySubProcess</flowNodeRef>
        <flowNodeRef>fornitureSubProcess</flowNodeRef>
      </lane>
    </laneSet>
    <userTask id="assResp" name="Individuazione competenze"
        activiti:assignee="kermit">
      <incoming>exGtwIncomingReqTOassResp</incoming>
      <incoming>protocolReqSubPrcTOassResp</incoming>
      <outgoing>assRespTOclaimProject</outgoing>
    </userTask>
    <callActivity id="signOfferSubProcess" name="Firmare offerta"
        calledElement="signProcess">
      <incoming>setLayoutSubProcTOsignOfferSubProc</incoming>
      <outgoing>signOfferSubProcTOprotocolSignedOffer</outgoing>
    </callActivity>
    <callActivity id="protocolSignedOffer" name="Protocollare offerta firmata"
        calledElement="protocolProcess">
      <extensionElements>
        <activiti:in source="provenienza" target="documento"/>
        <activiti:out source="documento" target="provenienza"/>
      </extensionElements>
      <incoming>signOfferSubProcTOprotocolSignedOffer</incoming>
      <outgoing>protocolSignedOfferTOsendOffer</outgoing>
    </callActivity>
    <exclusiveGateway gatewayDirection="Diverging" id="exGtwAcpt"
        name="Accettazione?">
      <incoming>sendOfferTOexGtwAcpt</incoming>
      <outgoing>exGtwAcptTOterminateEnd</outgoing>
      <outgoing>exGtwAcptTOinternalDistrib</outgoing>
    </exclusiveGateway>
    <endEvent id="terminateEnd" name="">
      <incoming>exGtwAcptTOterminateEnd</incoming>
      <terminateEventDefinition id="terminateEventDef"/>
    </endEvent>
    <callActivity id="adminSubProcess" name="Processo amministrativo"
        calledElement="adminProcess">
      <incoming>parGtwDiv1TOadminSubProc</incoming>
      <incoming>parGtwDiv2TOadminSubProc</incoming>
      <outgoing>adminSubProcTOparGtwConv</outgoing>
```

```
</callActivity>
<userTask id="claimProject" name="Presa in carico progetto"
        activiti:formKey="assegnazione.html"
        activiti:candidateGroups="accountancy, management, engineering">
    <incoming>assRespTOclaimProject</incoming>
    <outgoing>claimProjectTOanalisysSubProc</outgoing>
</userTask>
<callActivity id="analisysSubProcess" name="Analisi"
        calledElement="analisysProcess">
    <extensionElements>
        <activiti:in source="group" target="group"/>
        <activiti:out source="group" target="group"/>
    </extensionElements>
    <incoming>claimProjectTOanalisysSubProc</incoming>
    <outgoing>analisysSubProcTOverifyAnalisys</outgoing>
</callActivity>
<userTask id="verifyAnalisys" name="Verifica esito analisi"
        activiti:formKey="forniture.html"
        activiti:candidateGroups="${group}">
    <incoming>analisysSubProcTOverifyAnalisys</incoming>
    <outgoing>verifyAnalisysTOexGtwForniture</outgoing>
</userTask>
<exclusiveGateway gatewayDirection="Diverging"
        id="exGtwForniture" name="Necessaria fornitura esterna?">
    <incoming>verifyAnalisysTOexGtwForniture</incoming>
    <outgoing>exGtwFornitureTOofferReview</outgoing>
    <outgoing>exGtwFornitureTOsupplySubProc</outgoing>
</exclusiveGateway>
<userTask id="offerReview" name="Bozza d'offerta / Riesame"
        activiti:candidateGroups="${group}">
    <incoming>exGtwFornitureTOofferReview</incoming>
    <incoming>supplySubProcTOofferReview</incoming>
    <outgoing>offerReviewTOsetLayoutSubProc</outgoing>
</userTask>
<callActivity id="developmentSubProcess" name="Sviluppo Progetto"
        calledElement="projectDevelopmentProcess">
    <extensionElements>
        <activiti:in source="group" target="group"/>
        <activiti:out source="group" target="group"/>
    </extensionElements>
    <incoming>parGtwDiv2TOdevelopSubProc</incoming>
    <incoming>parGtwDiv1TOdevelopSubProc</incoming>
    <outgoing>developSubProcTOparGtwConv</outgoing>
</callActivity>
```

```
<startEvent id="startOfferProcess" name="">
  <outgoing>startProcessTOacquireDoc</outgoing>
</startEvent>
<userTask id="acquireDocument" name="Acquisizione documento richiesta"
      activiti:formKey="provenienza.html"
      activiti:assignee="kermit">
  <incoming>startProcessTOacquireDoc</incoming>
  <outgoing>acquireDocTOexGtwIncomingReq</outgoing>
</userTask>
<exclusiveGateway gatewayDirection="Diverging" id="exGtwIncomingReq"
      name="Provenienza?">
  <incoming>acquireDocTOexGtwIncomingReq</incoming>
  <outgoing>exGtwIncomingReqTOprotocolReqSubProc</outgoing>
  <outgoing>exGtwIncomingReqTOassResp</outgoing>
</exclusiveGateway>
<callActivity id="protocolReqSubProcess" name="Protocollare richiesta cliente"
      calledElement="protocolProcess">
  <extensionElements>
    <activiti:in source="provenienza" target="documento"/>
    <activiti:out source="documento" target="provenienza"/>
  </extensionElements>
  <incoming>exGtwIncomingReqTOprotocolReqSubProc</incoming>
  <outgoing>protocolReqSubPrcTOassResp</outgoing>
</callActivity>
<callActivity id="setLayoutSubProcess" name="Impostazione layout offerta"
      calledElement="layoutProcess">
  <incoming>offerReviewTOsetLayoutSubProc</incoming>
  <outgoing>setLayoutSubProcTOsignOfferSubProc</outgoing>
</callActivity>
<userTask id="sendOffer" name="Invio offerta / distribuzione interna"
      activiti:formKey="accettazione.html"
      activiti:assignee="kermit">
  <incoming>protocolSignedOfferTOsendOffer</incoming>
  <outgoing>sendOfferTOexGtwAcpt</outgoing>
</userTask>
<userTask id="internalDistribution" name="Distribuzione interna"
      activiti:assignee="kermit">
  <incoming>exGtwAcptTOinternalDistrib</incoming>
  <outgoing>internalDistribTOexGtwSupply</outgoing>
</userTask>
<exclusiveGateway gatewayDirection="Diverging" id="exGtwSupply" name="">
  <incoming>internalDistribTOexGtwSupply</incoming>
  <outgoing>exGtwSupplyTOparGtwDiv1</outgoing>
  <outgoing>exGtwSupplyTOparGtwDiv2</outgoing>
```

```
</exclusiveGateway>
<parallelGateway gatewayDirection="Diverging" id="parGtwDiv1" name="">
  <incoming>exGtwSupplyTOparGtwDiv1</incoming>
  <outgoing>parGtwDiv1TOdevelopSubProc</outgoing>
  <outgoing>parGtwDiv1TOfornitureSubProc</outgoing>
  <outgoing>parGtwDiv1TOadminSubProc</outgoing>
</parallelGateway>
<parallelGateway gatewayDirection="Converging" id="parGtwConv" name="">
  <incoming>developSubProcTOparGtwConv</incoming>
  <incoming>fornitureSubProcTOparGtwConv</incoming>
  <incoming>adminSubProcTOparGtwConv</incoming>
  <outgoing>parGtwConvTOendProcess</outgoing>
</parallelGateway>
<endEvent id="endOfferProcess" name="">
  <incoming>parGtwConvTOendProcess</incoming>
</endEvent>
<parallelGateway gatewayDirection="Diverging" id="parGtwDiv2" name="">
  <incoming>exGtwSupplyTOparGtwDiv2</incoming>
  <outgoing>parGtwDiv2TOdevelopSubProc</outgoing>
  <outgoing>parGtwDiv2TOadminSubProc</outgoing>
</parallelGateway>
<callActivity id="supplySubProcess" name="Approvvigionamenti"
      calledElement="supplyProcess">
  <incoming>exGtwFornitureTOsupplySubProc</incoming>
  <outgoing>supplySubProcTOofferReview</outgoing>
</callActivity>
<callActivity id="fornitureSubProcess" name="Processo approvvigionamenti"
      calledElement="fornitureProcess">
  <incoming>parGtwDiv1TOfornitureSubProc</incoming>
  <outgoing>fornitureSubProcTOparGtwConv</outgoing>
</callActivity>
<sequenceFlow id="startProcessTOacquireDoc" name=""
      sourceRef="startOfferProcess" targetRef="acquireDocument"/>
<sequenceFlow id="acquireDocTOexGtwIncomingReq" name=""
      sourceRef="acquireDocument" targetRef="exGtwIncomingReq"/>
<sequenceFlow id="protocolReqSubPrcTOassResp" name=""
      sourceRef="protocolReqSubProcess" targetRef="assResp"/>
<sequenceFlow id="assRespTOclaimProject" name=""
      sourceRef="assResp" targetRef="claimProject"/>
<sequenceFlow id="claimProjectTOanalisysSubProc" name=""
      sourceRef="claimProject" targetRef="analisysSubProcess"/>
<sequenceFlow id="analisysSubProcTOverifyAnalisys" name=""
      sourceRef="analisysSubProcess" targetRef="verifyAnalisys"/>
<sequenceFlow id="verifyAnalisysTOexGtwForniture" name=""
```

```
        sourceRef="verifyAnalisys" targetRef="exGtwForniture"/>
    <sequenceFlow id="supplySubProcTOofferReview" name=""
        sourceRef="supplySubProcess" targetRef="offerReview"/>
    <sequenceFlow id="offerReviewTOsetLayoutSubProc" name=""
        sourceRef="offerReview" targetRef="setLayoutSubProcess"/>
    <sequenceFlow id="setLayoutSubProcTOsignOfferSubProc" name=""
        sourceRef="setLayoutSubProcess" targetRef="signOfferSubProcess"/>
    <sequenceFlow id="signOfferSubProcTOprotocolSignedOffer" name=""
        sourceRef="signOfferSubProcess" targetRef="protocolSignedOffer"/>
    <sequenceFlow id="protocolSignedOfferTOsendOffer" name=""
        sourceRef="protocolSignedOffer" targetRef="sendOffer"/>
    <sequenceFlow id="sendOfferTOexGtwAcpt" name=""
        sourceRef="sendOffer" targetRef="exGtwAcpt"/>
    <sequenceFlow id="internalDistribTOexGtwSupply" name=""
        sourceRef="internalDistribution" targetRef="exGtwSupply"/>
    <sequenceFlow id="parGtwDiv2TOdevelopSubProc" name=""
        sourceRef="parGtwDiv2" targetRef="developmentSubProcess"/>
    <sequenceFlow id="parGtwDiv1TOdevelopSubProc" name=""
        sourceRef="parGtwDiv1" targetRef="developmentSubProcess"/>
    <sequenceFlow id="parGtwDiv1TOfornitureSubProc" name=""
        sourceRef="parGtwDiv1" targetRef="fornitureSubProcess"/>
    <sequenceFlow id="parGtwDiv1TOadminSubProc" name=""
        sourceRef="parGtwDiv1" targetRef="adminSubProcess"/>
    <sequenceFlow id="parGtwDiv2TOadminSubProc" name=""
        sourceRef="parGtwDiv2" targetRef="adminSubProcess"/>
    <sequenceFlow id="adminSubProcTOparGtwConv" name=""
        sourceRef="adminSubProcess" targetRef="parGtwConv"/>
    <sequenceFlow id="parGtwConvTOendProcess" name=""
        sourceRef="parGtwConv" targetRef="endOfferProcess"/>
    <sequenceFlow id="developSubProcTOparGtwConv" name=""
        sourceRef="developmentSubProcess" targetRef="parGtwConv"/>
    <sequenceFlow id="fornitureSubProcTOparGtwConv" name=""
        sourceRef="fornitureSubProcess" targetRef="parGtwConv"/>
    <sequenceFlow id="exGtwIncomingReqTOprotocolReqSubProc" name="Richiesta cliente"
        sourceRef="exGtwIncomingReq" targetRef="protocolReqSubProcess">
      <conditionExpression xsi:type="tFormalExpression">
        ${provenienza == "cliente"}
      </conditionExpression>
    </sequenceFlow>
    <sequenceFlow id="exGtwIncomingReqTOassResp" name="Ordine interno"
        sourceRef="exGtwIncomingReq" targetRef="assResp">
      <conditionExpression xsi:type="tFormalExpression">
        ${provenienza == "interno"}
      </conditionExpression>
```

```
    </sequenceFlow>
    <sequenceFlow id="exGtwFornitureTOofferReview" name="No"
        sourceRef="exGtwForniture" targetRef="offerReview">
      <conditionExpression xsi:type="tFormalExpression">
        ${fornitura == "no"}
      </conditionExpression>
    </sequenceFlow>
    <sequenceFlow id="exGtwFornitureTOsupplySubProc" name="Si"
        sourceRef="exGtwForniture" targetRef="supplySubProcess">
      <conditionExpression xsi:type="tFormalExpression">
        ${fornitura == "si"}
      </conditionExpression>
    </sequenceFlow>
    <sequenceFlow id="exGtwAcptTOterminateEnd" name="No"
        sourceRef="exGtwAcpt" targetRef="terminateEnd">
      <conditionExpression xsi:type="tFormalExpression">
        ${accettazione == "no"}
      </conditionExpression>
    </sequenceFlow>
    <sequenceFlow id="exGtwAcptTOinternalDistrib" name="Si"
        sourceRef="exGtwAcpt" targetRef="internalDistribution">
      <conditionExpression xsi:type="tFormalExpression">
        ${accettazione == "si"}
      </conditionExpression>
    </sequenceFlow>
    <sequenceFlow id="exGtwSupplyTOparGtwDiv2" name="Senza forniture"
        sourceRef="exGtwSupply" targetRef="parGtwDiv2">
      <conditionExpression xsi:type="tFormalExpression">
        ${fornitura == "no"}
      </conditionExpression>
    </sequenceFlow>
    <sequenceFlow id="exGtwSupplyTOparGtwDiv1" name="Con forniture"
        sourceRef="exGtwSupply" targetRef="parGtwDiv1">
      <conditionExpression xsi:type="tFormalExpression">
        ${fornitura == "si"}
      </conditionExpression>
    </sequenceFlow>
  </process>
  <collaboration id="collaboration">
    <participant id="pool" name="Processo di gestione dell'offerta"
        processRef="offerManagement">
    </participant>
  </collaboration>
</definitions>
```

Modellando i processi con Activiti Modeler, il codice creato a partire dal modello grafico del processo (figura 5.2), è integrato con una parte inserita in automatico (non funzionale al suo corretto funzionamento) che rappresenta la parte grafica vera e propria, cioè memorizza le coordinate dei vari elementi inseriti nel diagramma di processo, in modo che quest' ultimo venga correttamente rappresentato in qualsiasi altro software di rappresentazione di business process. Inserirò di seguito una parte di questo codice a titolo d'esempio:

```
<bpmndi:BPMNDiagram id="testOfferProcess">
  <bpmndi:BPMNPlane bpmnElement="collaboration" id="testOfferProcessPlane">
    <bpmndi:BPMNShape bpmnElement="adminLane" id="adminLane_gui" isHorizontal="true">

      <omgdc:Bounds height="250.66666666666674" width="1733.0"
                    x="220.16236852264115" y="1801.347182182625"/>
    </bpmndi:BPMNShape>
    ....
  </bpmndi:BPMNPlane>
</bpmndi:BPMNDiagram>
```

che in forma grafica corrisponde alla figura 6.1.
Quando si dichiara uno user task è necessario anche definire a chi verrà assegnato. Ci sono 3 attributi differenti per dichiarare l'utente incaricato:

1. *activiti:assignee="user_id"*, assegna il task ad un singolo utente specificato dal suo user_id. Nel momento in cui l'utente effettuerà il login in Activiti Explorer, troverà il task tra i suoi compiti ancora da eseguire (nella sezione MyTasks);

2. *activiti:candidateUsers="user_id1, user_id2, ..."*, fa in modo che più utenti siano candidati al completamento del task. Il primo di questi utenti che entrerà in Explorer, troverà il task tra i compiti non assegnati (sezione Unassigned tasks) e potrà prenderlo in carico;

3. *activiti:candidateGroups="group_id1, group_id2, ... "*, funziona come il candidateUsers, con la differenza che il task può essere svolto da un qualsiasi utente appartenente ad uno dei gruppi dichiarati. Il task verrà visualizzato nelle sezioni corrispondenti ai gruppi dichiarati e nella sezione Unassigned tasks.

Quando il processo giunge in corrispondenza di un gateway esclusivo, si devono impostare delle condizioni sui sequence flow d'uscita dichiarando delle variabili. Anche qui ci sono diversi modi, ma quello più semplice e diretto è dato dall'attributo *activiti:formKey="form_name.html"*. Ciò significa che quando si esegue il task, viene

visualizzato un form html, che l'utente deve compliare, dal quale si ottengono i valori delle variabili di interesse, per stabilire su quale sequence flow il processo viene instradato dall'Activiti Engine. Ad esempio, nel task denominato "Acquisizione documento richiesta", al momento dell'esecuzione visualizzerà il form definito nel file "provenienza.html", che dev'essere contenuto nella stessa cartella in cui sono memorizzati anche i file bpmn20.xml, come rappresentato in figura 5.1.
Per il corretto funzionamento del processo di gestione dell'offerta è stato necessario definire i seguenti form:

1. **provenienza.html**, come detto, associato al task "Acquisizione documento richiesta":

```
<html>
  <body>
    <h1>Acquisizione documento richiesta</h1>
    <select name="provenienza">
      <option value="cliente">Richiesta da cliente</option>
      <option value="interno">Ordine interno</option>
    </select>
  </body>
</html>
```

2. **forniture.html**, associato al task "Analisi":

```
<html>
  <body>
    <h1>Analisi</h1>
    Necessarie forniture esterne?
    <select name="fornitura">
      <option value="si">Si</option>
      <option value="no">No</option>
    </select>
  </body>
</html>
```

3. **accettazione.html**, associato al task "Invio offerta / Distribuzione interna":

```
<html>
  <body>
    <h1>Acettazione offerta?</h1>
    <select name="accettazione">
      <option value="si">Si</option>
      <option value="no">No</option>
    </select>
  </body>
</html>
```

L'attributo *name* del tag *select*, identifica il nome della variabile che, nel processo in cui il form viene utilizzato, vale come variabile di processo, ossia è visibile per tutta la durata della sua esecuzione. Queste vengono utilizzate all'interno dei tag *conditionExpression*, figli dei tag *sequenceFlow*, usando la sintassi $\$\{condizione\_sulla\_variabile\}$.
Le variabili sono rese visibili anche ai sottoprocessi chimati dal processo principale, utilizzando all'interno delle *callActivity* le istruzioni seguenti:

```
<extensionElements>
  <activiti:in source="nome variabile nel processo principale"
      target="nome variabile nel sottoprocesso"/>
  <activiti:out source="nome variabile nel sottoprocesso"
      target="nome variabile nel processo principale"/>
</extensionElements>
```

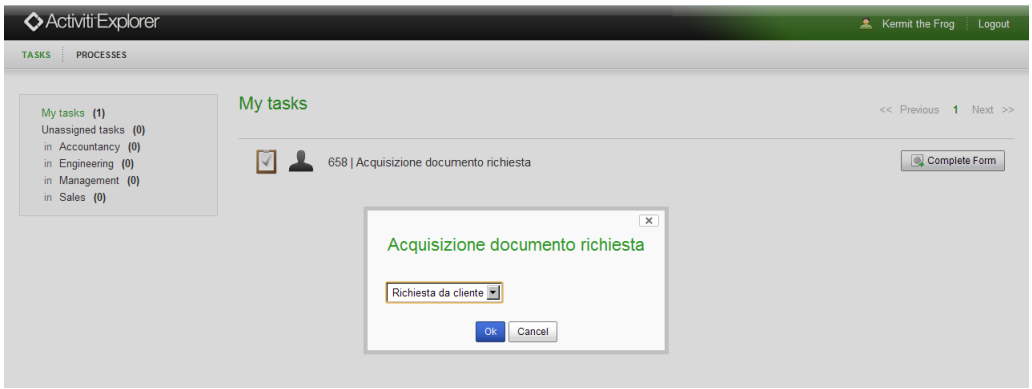


Figura 5.1: *Esempio di visualizzazione di un form associato ad un task*

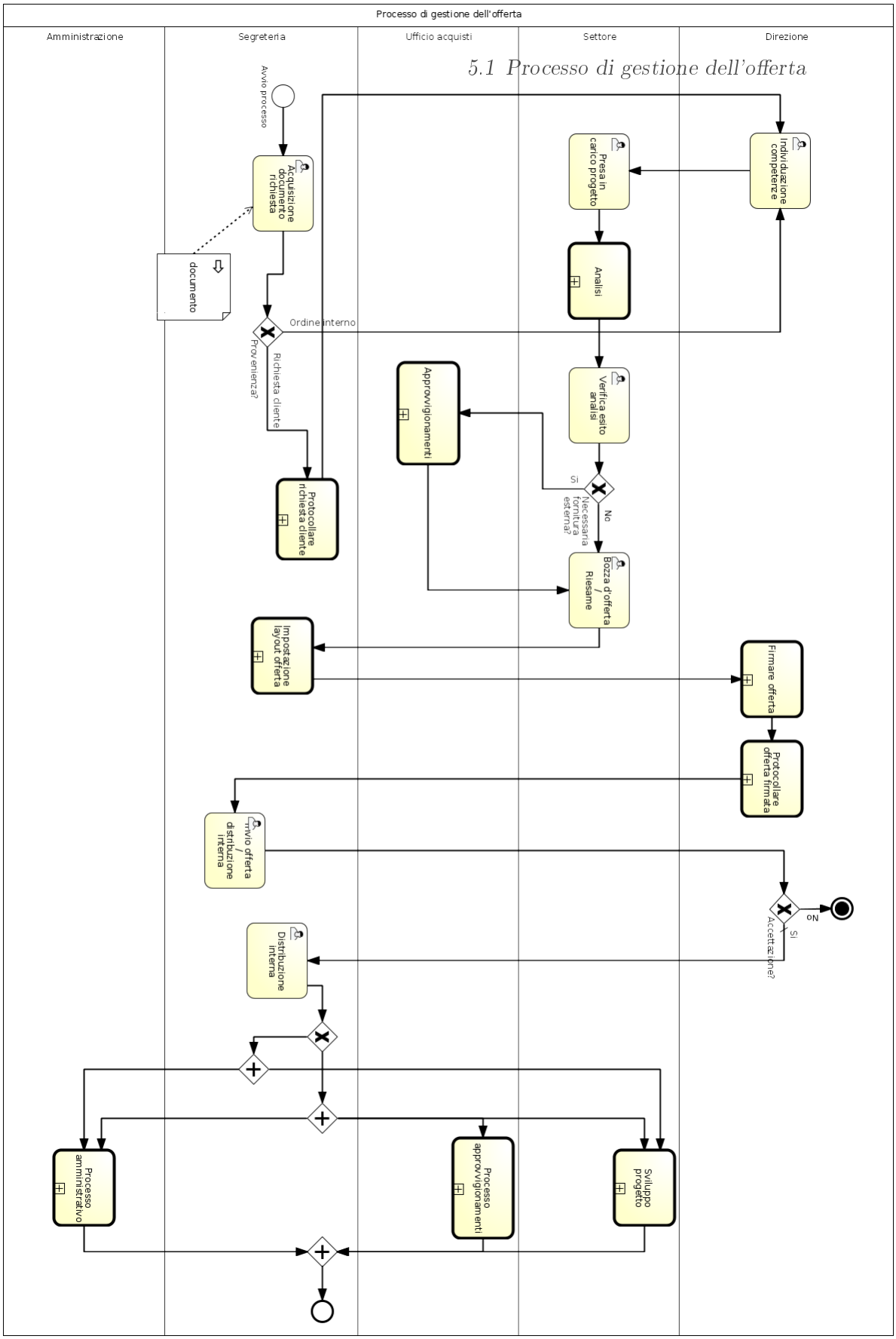


Figura 5.2: *BPD TestOfferta*

5.2 Sottoprocesso di protocollazione

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
    xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
    xmlns:omgdc="http://www.omg.org/spec/DD/20100524/DC"
    xmlns:omgdi="http://www.omg.org/spec/DD/20100524/DI"
    xmlns:signavio="http://www.signavio.com"
    xmlns:activiti="http://activiti.org/bpmn"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    id="sid-1245a4dd-1020-48db-b7da-8c4c88d92c10"
    targetNamespace="http://activiti.org/bpmn20"
    typeLanguage="http://www.w3.org/2001/XMLSchema"
    xsi:schemaLocation="http://www.omg.org/spec/BPMN/20100524/MODEL
        http://www.omg.org/spec/BPMN/2.0/20100501/BPMN20.xsd">
    <process id="protocolProcess" name="Processo di protocollazione documenti">
      <startEvent id="startProtocol" name="">
        <outgoing>startTOread</outgoing>
      </startEvent>
      <userTask id="readMessage" name="Leggi messaggio"
          activiti:candidateGroups="accountancy">
        <incoming>startTOread</incoming>
        <outgoing>readTOexGtw</outgoing>
      </userTask>
      <exclusiveGateway default="exGtwProtTOassProtOff"
          gatewayDirection="Diverging" id="exGtwProtocol"
          name="Documento da protocollare?">
        <incoming>readTOexGtw</incoming>
        <outgoing>exGtwProtTOassProtReq</outgoing>
        <outgoing>exGtwProtTOassProtOff</outgoing>
      </exclusiveGateway>
      <userTask id="assignProtocolReq" name="Assegnazione protocollo richiesta"
          activiti:candidateGroups="accountancy">
        <incoming>exGtwProtTOassProtReq</incoming>
        <outgoing>assProtReqTOscanDoc</outgoing>
      </userTask>
      <userTask id="scanDoc" name="Scannerizzazione documento"
          activiti:candidateGroups="accountancy">
        <incoming>assProtOffTOscanDoc</incoming>
        <incoming>assProtReqTOscanDoc</incoming>
        <outgoing>scanDocTOdigitalSign</outgoing>
      </userTask>
      <userTask id="assignProtocolOffer" name="Assegnazione protocollo offerta"
          activiti:candidateGroups="accountancy">
        <incoming>exGtwProtTOassProtOff</incoming>
```

```
        <outgoing>assProtOffTOscanDoc</outgoing>
      </userTask>
      <userTask id="digitalSign" name="Firma digitale documento"
          activiti:candidateGroups="accountancy">
        <incoming>scanDocTOdigitalSign</incoming>
        <outgoing>digitalSignTOsendPEC</outgoing>
      </userTask>
      <userTask id="sendPEC" name="Invia documento tramite PEC"
          activiti:candidateGroups="accountancy">
        <incoming>digitalSignTOsendPEC</incoming>
        <outgoing>sendTOend</outgoing>
      </userTask>
      <endEvent id="endProtocol" name="">
        <incoming>sendTOend</incoming>
      </endEvent>
      <sequenceFlow id="startTOread" name=""
        sourceRef="startProtocol" targetRef="readMessage"/>
      <sequenceFlow id="readTOexGtw" name=""
        sourceRef="readMessage" targetRef="exGtwProtocol"/>
      <sequenceFlow id="assProtReqTOscanDoc" name=""
        sourceRef="assignProtocolReq" targetRef="scanDoc"/>
      <sequenceFlow id="assProtOffTOscanDoc" name=""
        sourceRef="assignProtocolOffer" targetRef="scanDoc"/>
      <sequenceFlow id="scanDocTOdigitalSign" name=""
        sourceRef="scanDoc" targetRef="digitalSign"/>
      <sequenceFlow id="digitalSignTOsendPEC" name=""
        sourceRef="digitalSign" targetRef="sendPEC"/>
      <sequenceFlow id="sendTOend" name=""
        sourceRef="sendPEC" targetRef="endProtocol"/>
      <sequenceFlow id="exGtwProtTOassProtReq" name="Richiesta cliente"
        sourceRef="exGtwProtocol" targetRef="assignProtocolReq">
        <conditionExpression xsi:type="tFormalExpression">
          ${documento == "cliente"}
        </conditionExpression>
      </sequenceFlow>
      <sequenceFlow id="exGtwProtTOassProtOff" name="Offerta"
        sourceRef="exGtwProtocol" targetRef="assignProtocolOffer"/>
    </process>
</definitions>
```

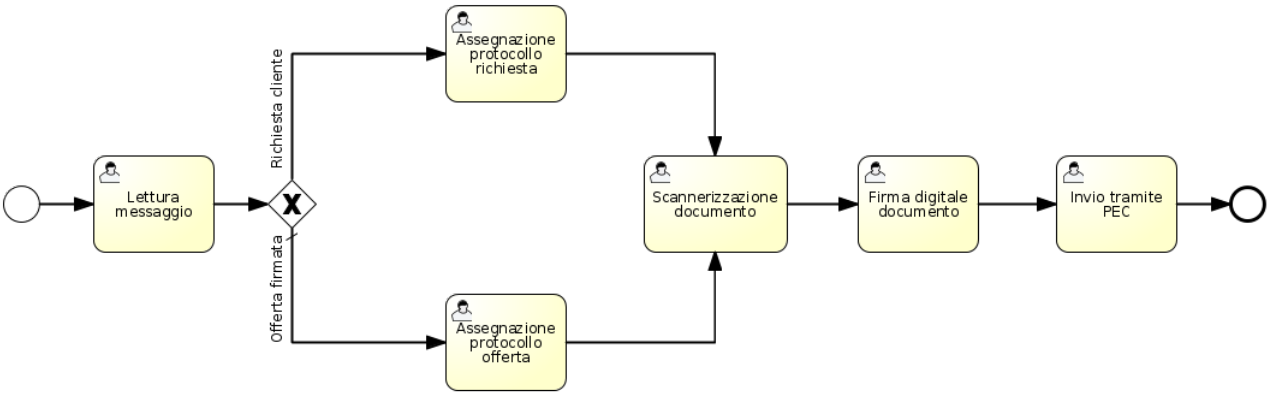


Figura 5.3: *BPD Protocollo*

5.3 Sottoprocesso di impostazione del layout dell'offerta

```
<?xml version="1.0" encoding="UTF-8"?>
  <definitions xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
      xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
      xmlns:omgdc="http://www.omg.org/spec/DD/20100524/DC"
      xmlns:omgdi="http://www.omg.org/spec/DD/20100524/DI"
      xmlns:signavio="http://www.signavio.com"
      xmlns:activiti="http://activiti.org/bpmn"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      targetNamespace="http://activiti.org/bpmn20"
      typeLanguage="http://www.w3.org/2001/XMLSchema"
      xsi:schemaLocation="http://www.omg.org/spec/BPMN/20100524/MODEL
          http://www.omg.org/spec/BPMN/2.0/20100501/BPMN20.xsd">
      <process id="layoutProcess" name="Impostazione layout offerta">
        <startEvent id="startLayout" name="">
          <documentation> Impostazione layout del documento
                          d'offerta a seconda del tipo di
                          documento
          </documentation>
          <outgoing>startTOget_type</outgoing>
        </startEvent>
        <userTask id="getDocType" name="Estrazione tipologia documento"
          activiti:formKey="layout.html" activiti:candidateGroups="accountancy">
          <incoming>startTOget_type</incoming>
          <outgoing>get_typeTOex_gtw</outgoing>
        </userTask>
        <exclusiveGateway gatewayDirection="Diverging" id="exGtwDiv" name="">
```

```
    <incoming>get_typeTOex_gtw</incoming>
    <outgoing>ex_gtwTOset_L3</outgoing>
    <outgoing>ex_gtwTOset_L2</outgoing>
    <outgoing>ex_gtwTOset_L1</outgoing>
    <outgoing>ex_gtwTOset_L4</outgoing>
  </exclusiveGateway>
  <userTask id="setLayout3" name="Impostazione Layout_3"
    activiti:candidateGroups="accountancy">
    <incoming>ex_gtwTOset_L3</incoming>
    <outgoing>set_L3TOerr_rev</outgoing>
  </userTask>
  <userTask id="errReview" name="Revisionare errori"
    activiti:candidateGroups="accountancy">
    <incoming>set_L1TOerr_rev</incoming>
    <incoming>set_L2TOerr_rev</incoming>
    <incoming>set_L3TOerr_rev</incoming>
    <incoming>set_L4TOerr_rev</incoming>
    <outgoing>err_revTOend</outgoing>
  </userTask>
  <userTask id="setLayout1" name="Impostazione Layout_1"
    activiti:candidateGroups="accountancy">
    <incoming>ex_gtwTOset_L1</incoming>
    <outgoing>set_L1TOerr_rev</outgoing>
  </userTask>
  <userTask id="setLayout2" name="Impostazione Layout_2"
    activiti:candidateGroups="accountancy">
    <incoming>ex_gtwTOset_L2</incoming>
    <outgoing>set_L2TOerr_rev</outgoing>
  </userTask>
  <userTask id="setLayout4" name="Impostazione Layout_4"
    activiti:candidateGroups="accountancy">
    <incoming>ex_gtwTOset_L4</incoming>
    <outgoing>set_L4TOerr_rev</outgoing>
  </userTask>
  <endEvent id="endLayout" name="">
    <incoming>err_revTOend</incoming>
  </endEvent>
  <sequenceFlow id="startTOget_type" name=""
    sourceRef="startLayout" targetRef="getDocType"/>
  <sequenceFlow id="set_L1TOerr_rev" name=""
    sourceRef="setLayout1" targetRef="errReview"/>
  <sequenceFlow id="set_L2TOerr_rev" name=""
    sourceRef="setLayout2" targetRef="errReview"/>
  <sequenceFlow id="set_L3TOerr_rev" name=""
```

```
        sourceRef="setLayout3" targetRef="errReview"/>
      <sequenceFlow id="set_L4TOerr_rev" name=""
        sourceRef="setLayout4" targetRef="errReview"/>
      <sequenceFlow id="err_revTOend" name=""
        sourceRef="errReview" targetRef="endLayout"/>
      <sequenceFlow id="get_typeTOex_gtw" name=""
        sourceRef="getDocType" targetRef="exGtwDiv"/>
      <sequenceFlow id="ex_gtwTOset_L2" name="Tipo 2"
        sourceRef="exGtwDiv" targetRef="setLayout2">
        <conditionExpression xsi:type="tFormalExpression">
          ${layout == "tipo2"}
        </conditionExpression>
      </sequenceFlow>
      <sequenceFlow id="ex_gtwTOset_L1" name="Tipo 1"
        sourceRef="exGtwDiv" targetRef="setLayout1">
        <conditionExpression xsi:type="tFormalExpression">
          ${layout == "tipo1"}
        </conditionExpression>
      </sequenceFlow>
      <sequenceFlow id="ex_gtwTOset_L3" name="Tipo 3"
        sourceRef="exGtwDiv" targetRef="setLayout3">
        <conditionExpression xsi:type="tFormalExpression">
          ${layout == "tipo3"}
        </conditionExpression>
      </sequenceFlow>
      <sequenceFlow id="ex_gtwTOset_L4" name="Tipo 4"
        sourceRef="exGtwDiv" targetRef="setLayout4">
        <conditionExpression xsi:type="tFormalExpression">
          ${layout == "tipo4"}
        </conditionExpression>
      </sequenceFlow>
    </process>
  </defintions>
```

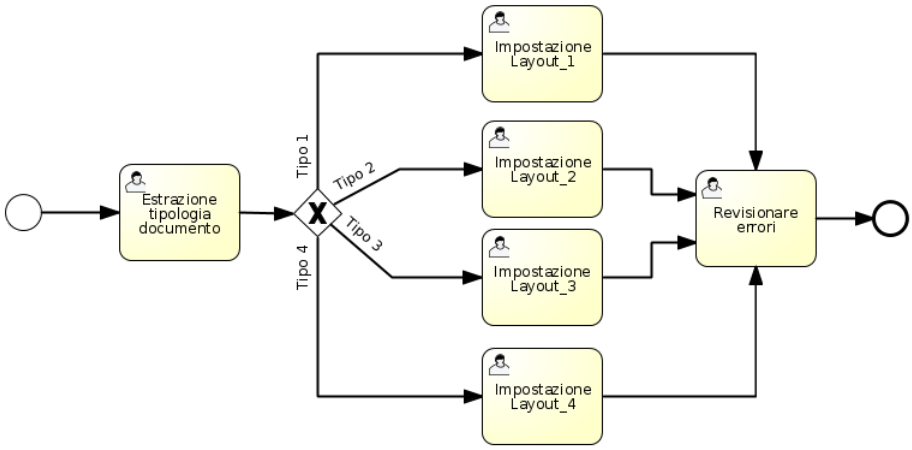


Figura 5.4: *BPD Impostazione layout offerta*

È stato utilizzato un form anche nel processo "Impostazione layout offerta". Questo sottoprocesso è stato creato solo a scopo descrittivo, nel senso che il flusso di questo processo non rispecchia completamente quello reale. Il task "Estrazione tipologia documento" rappresenta il lavoro che la segreteria svolge nell'identificare di che tipo sia il documento. Viene impostato un layout particolare che varia a seconda che il documento sia un'offerta, un'ordine interno o altro.

Il form *layout.html* è così definito:

- **layout.html**:

```
<html>
  <body>
    <h1>Impostazione layout offerta</h1>
    Seleziona il tipo di layout: </br>
    <select name="layout">
      <option value="tipo1">Tipo 1</option>
      <option value="tipo2">Tipo 2</option>
      <option value="tipo3">Tipo 3</option>
      <option value="tipo4">Tipo 4</option>
    </select>
  </body>
</html>
```

5.4 Sottoprocesso di firma dell'offerta

```
<?xml version="1.0" encoding="UTF-8"?>
  <definitions xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
      xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
      xmlns:omgdc="http://www.omg.org/spec/DD/20100524/DC"
      xmlns:omgdi="http://www.omg.org/spec/DD/20100524/DI"
      xmlns:signavio="http://www.signavio.com"
      xmlns:activiti="http://activiti.org/bpmn"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      id="sid-f325f9ae-eb27-4d2d-8955-b69141c72aa0"
      targetNamespace="http://activiti.org/bpmn20"
      typeLanguage="http://www.w3.org/2001/XMLSchema"
      xsi:schemaLocation="http://www.omg.org/spec/BPMN/20100524/MODEL
          http://www.omg.org/spec/BPMN/2.0/20100501/BPMN20.xsd">
      <process id="signProcess" name="Firmare offerta">
        <startEvent id="startSign" name="">
          <documentation> Lettura e approvazione dell'offerta
                          da parte della direzione aziendale.
          </documentation>
          <outgoing>startSignTOwriteOffer</outgoing>
        </startEvent>
        <userTask id="writeOffer" name="Scrivi offerta"
            activiti:formKey="importo.html" activiti:candidateGroups="management">
          <incoming>exGtwMasterApprove1TOwriteOffer</incoming>
          <incoming>startSignTOwriteOffer</incoming>
          <incoming>exGtwManagerApproveTOwriteOffer</incoming>
          <incoming>exGtwMasterApprove2TOwriteOffer</incoming>
          <outgoing>writeOfferTOexGtwAmount</outgoing>
        </userTask>
        <exclusiveGateway gatewayDirection="Diverging"
            id="exGtwMasterApprove1" name="Offerta accettata?">
          <incoming>masterApprove1TOexGtwMasterApprove1</incoming>
          <outgoing>exGtwMasterApprove1TOwriteOffer</outgoing>
          <outgoing>exGtwMasterApprove1TOendApprove1</outgoing>
        </exclusiveGateway>
        <userTask id="masterApprove1" name="Approvazione e firma dirigente"
            activiti:formKey="approvazione.html" activiti:assignee="kermit">
          <incoming>exGtwAmountTOmasterApprove1</incoming>
          <outgoing>masterApprove1TOexGtwMasterApprove1</outgoing>
        </userTask>
        <exclusiveGateway gatewayDirection="Diverging" id="exGtwAmount"
            name="Importo?">
          <incoming>writeOfferTOexGtwAmount</incoming>
          <outgoing>exGtwAmountTOmasterApprove1</outgoing>
```

```
    <outgoing>exGtwAmountTOmanagerApprove</outgoing>
  </exclusiveGateway>
  <userTask id="managerApprove" name="Approvazione dirigente"
      activiti:formKey="approvazione.html" activiti:assignee="kermit">
    <incoming>exGtwAmountTOmanagerApprove</incoming>
    <outgoing>managerApproveTOexGtwManagerApprove</outgoing>
  </userTask>
  <exclusiveGateway gatewayDirection="Diverging"
      id="exGtwManagerApprove" name="Approvazione dirigente?">
    <incoming>managerApproveTOexGtwManagerApprove</incoming>
    <outgoing>exGtwManagerApproveTOwriteOffer</outgoing>
    <outgoing>exGtwManagerApproveTOmasterApprove2</outgoing>
  </exclusiveGateway>
  <userTask id="masterApprove2" name="Approvazione e firma direttore"
      activiti:formKey="approvazione.html" activiti:assignee="gonzo">
    <incoming>exGtwManagerApproveTOmasterApprove2</incoming>
    <outgoing>masterApprove2TOexGtwMasterApprove2</outgoing>
  </userTask>
  <exclusiveGateway gatewayDirection="Diverging"
      id="exGtwMasterApprove2" name="Approvazione direttore?">
    <incoming>masterApprove2TOexGtwMasterApprove2</incoming>
    <outgoing>exGtwMasterApprove2TOendApprove2</outgoing>
    <outgoing>exGtwMasterApprove2TOwriteOffer</outgoing>
  </exclusiveGateway>
  <endEvent id="endApprove2" name="">
    <incoming>exGtwMasterApprove2TOendApprove2</incoming>
  </endEvent>
  <endEvent id="endApprove1" name="">
    <incoming>exGtwMasterApprove1TOendApprove1</incoming>
  </endEvent>
  <sequenceFlow id="startSignTOwriteOffer" name=""
      sourceRef="startSign" targetRef="writeOffer"/>
  <sequenceFlow id="writeOfferTOexGtwAmount" name=""
      sourceRef="writeOffer" targetRef="exGtwAmount"/>
  <sequenceFlow id="managerApproveTOexGtwManagerApprove" name=""
      sourceRef="managerApprove" targetRef="exGtwManagerApprove"/>
  <sequenceFlow id="masterApprove1TOexGtwMasterApprove1" name=""
      sourceRef="masterApprove1" targetRef="exGtwMasterApprove1"/>
  <sequenceFlow id="exGtwAmountTOmasterApprove1"
      name="Importo &amp;lt; 20000 euro"
      sourceRef="exGtwAmount" targetRef="masterApprove1">
    <conditionExpression xsi:type="tFormalExpression">
      ${importo &amp;lt; 20000}
    </conditionExpression>
```

```
</sequenceFlow>
<sequenceFlow id="masterApprove2TOexGtwMasterApprove2" name=""
    sourceRef="masterApprove2" targetRef="exGtwMasterApprove2"/>
<sequenceFlow id="exGtwMasterApprove2TOendApprove2" name="Si"
    sourceRef="exGtwMasterApprove2" targetRef="endApprove2">
  <conditionExpression xsi:type="tFormalExpression">
    ${approvazione == "si"}
  </conditionExpression>
</sequenceFlow>
<sequenceFlow id="exGtwManagerApproveTOmasterApprove2" name="Si"
    sourceRef="exGtwManagerApprove" targetRef="masterApprove2">
  <conditionExpression xsi:type="tFormalExpression">
    ${approvazione == "si"}
  </conditionExpression>
</sequenceFlow>
<sequenceFlow id="exGtwManagerApproveTOwriteOffer" name="No"
    sourceRef="exGtwManagerApprove" targetRef="writeOffer">
  <conditionExpression xsi:type="tFormalExpression">
    ${approvazione == "no"}
  </conditionExpression>
</sequenceFlow>
<sequenceFlow id="exGtwMasterApprove2TOwriteOffer" name="No"
    sourceRef="exGtwMasterApprove2" targetRef="writeOffer">
  <conditionExpression xsi:type="tFormalExpression">
    ${approvazione == "no"}
  </conditionExpression>
</sequenceFlow>
<sequenceFlow id="exGtwMasterApprove1TOendApprove1" name="Si"
    sourceRef="exGtwMasterApprove1" targetRef="endApprove1">
  <conditionExpression xsi:type="tFormalExpression">
    ${approvazione == "si"}
  </conditionExpression>
</sequenceFlow>
<sequenceFlow id="exGtwMasterApprove1TOwriteOffer" name="No"
    sourceRef="exGtwMasterApprove1" targetRef="writeOffer">
  <conditionExpression xsi:type="tFormalExpression">
    ${approvazione == "no"}
  </conditionExpression>
</sequenceFlow>
<sequenceFlow id="exGtwAmountTOmanagerApprove"
    name="Importo &gt; 20000 euro"
    sourceRef="exGtwAmount" targetRef="managerApprove">
  <conditionExpression xsi:type="tFormalExpression">
    ${importo &gt; 20000}
```

```
        </conditionExpression>
      </sequenceFlow>
    </process>
</definitions>
```

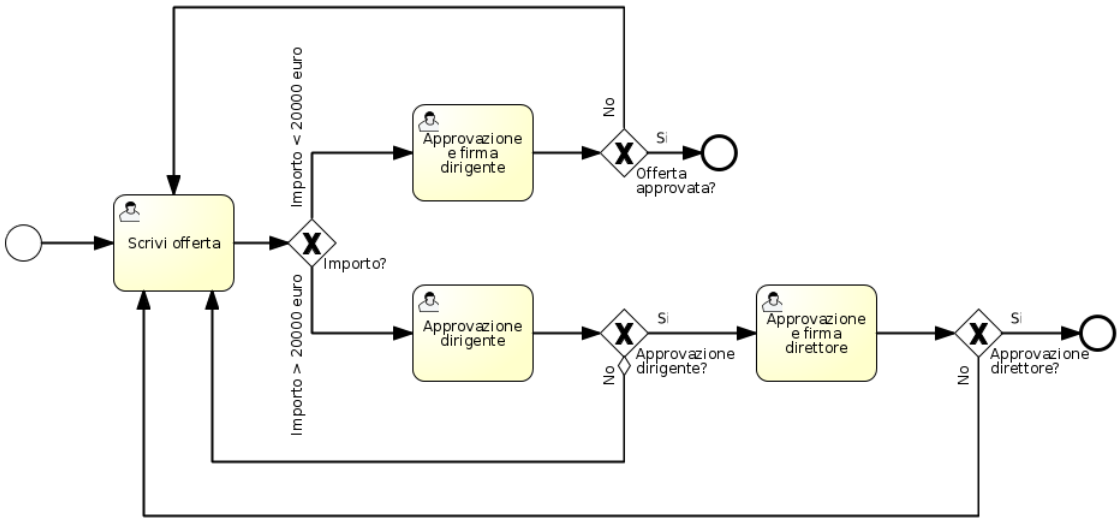


Figura 5.5: *BPD Firma*

I form utilizzati in questo sottoprocesso sono due: "*importo.html*" e "*approvazione.html*". Il secondo viene usato 3 volte, in corrispondenza dei gateway "*Offerta approvata?*", "*Approvazione dirigente?*" e "*Approvazione direttore?*". Sono stati così definiti:

- **importo.html**:

```
<html>
  <body>
    <h1>Estrazione importo offerta</h1>
    Importo:<br/>
    <input type="text" name="importo" value=""/>
    <input type="hidden" name="importo_type" value="Integer"/>
  </body>
</html>
```

- **approvazione.html**:

```
<html>
  <body>
    <h1>Approvazione offerta?</h1>
    <select name="approvazione">
      <option value="si">Si</option>
      <option value="no">No</option>
    </select>
  </body>
</html>
```

5.5 Sottoprocesso di valutazione degli approvvigionamenti

```
<?xml version="1.0" encoding="iso-8859-1"?>
  <definitions xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
      xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
      xmlns:omgdc="http://www.omg.org/spec/DD/20100524/DC"
      xmlns:omgdi="http://www.omg.org/spec/DD/20100524/DI"
      xmlns:signavio="http://www.signavio.com"
      xmlns:activiti="http://activiti.org/bpmn"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      targetNamespace="http://activiti.org/bpmn20"
      typeLanguage="http://www.w3.org/2001/XMLSchema"
      xsi:schemaLocation="http://www.omg.org/spec/BPMN/20100524/MODEL
          http://www.omg.org/spec/BPMN/2.0/20100501/BPMN20.xsd">
      <process id="supplyProcess" name="Approvvigionamenti">
        <startEvent id="startSupply" name="startSupplyProcess">
          <documentation> Effettuare studio di fattibilità,
                          progetto di massima e altre
                          valutazioni per stabilire quante,
                          quali e quanto costino le risorse necessarie.
          </documentation>
          <outgoing>startTOec_ev</outgoing>
        </startEvent>
        <userTask id="economicEvaluation" name="Valutazione quantitativa, economica
            e finanziaria" activiti:candidateGroups="management">
          <incoming>startTOec_ev</incoming>
          <outgoing>ec_evTOtime_ev</outgoing>
        </userTask>
        <userTask id="timeEvaluation" name="Valutazione tempi di consegna"
            activiti:candidateGroups="accountancy">
```

```
      <incoming>ec_evTOtime_ev</incoming>
      <outgoing>time_evTOwrite_q</outgoing>
    </userTask>
    <userTask id="writeQuote" name="Redazione preventivo"
        activiti:candidateGroups="accountancy">
      <incoming>time_evTOwrite_q</incoming>
      <outgoing>write_qTOend</outgoing>
    </userTask>
    <endEvent id="endSupply" name="endSupplyProcess">
      <incoming>write_qTOend</incoming>
    </endEvent>
    <sequenceFlow id="startTOec_ev" name=""
        sourceRef="startSupply" targetRef="economicEvaluation"/>
    <sequenceFlow id="ec_evTOtime_ev" name=""
        sourceRef="economicEvaluation" targetRef="timeEvaluation"/>
    <sequenceFlow id="time_evTOwrite_q" name=""
        sourceRef="timeEvaluation" targetRef="writeQuote"/>
    <sequenceFlow id="write_qTOend" name=""
        sourceRef="writeQuote" targetRef="endSupply"/>
  </process>
</definitions>
```

Figura 5.6: *BPD Approvvigionamenti*

5.6 Sottoprocesso di analisi del progetto

```
<?xml version="1.0" encoding="ISO-8859-1"?>
  <definitions xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
      xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
      xmlns:omgdc="http://www.omg.org/spec/DD/20100524/DC"
      xmlns:omgdi="http://www.omg.org/spec/DD/20100524/DI"
      xmlns:signavio="http://www.signavio.com"
      xmlns:activiti="http://activiti.org/bpmn"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      id="sid-894b2665-83de-402b-b501-f14284f008ae"
      targetNamespace="http://activiti.org/bpmn20"
      typeLanguage="http://www.w3.org/2001/XMLSchema"
```

```
        xsi:schemaLocation="http://www.omg.org/spec/BPMN/20100524/MODEL
            http://www.omg.org/spec/BPMN/2.0/20100501/BPMN20.xsd">
        <process id="analisysProcess" name="Analisi di fattibilità">
          <startEvent id="startAnalisys" name="">
            <outgoing>startTOrequirement</outgoing>
          </startEvent>
          <userTask id="requirementAnalisys" name="Raccolta e analisi requisiti"
              activiti:candidateGroups="accountancy">
            <incoming>startTOrequirement</incoming>
            <outgoing>requirementTOfeasibility</outgoing>
          </userTask>
          <userTask id="feasibilityStudy" name="Studio di fattibilità"
              activiti:candidateGroups="accountancy">
            <incoming>requirementTOfeasibility</incoming>
            <outgoing>feasibilityTOend</outgoing>
          </userTask>
          <endEvent id="endAnalisys" name="">
            <incoming>feasibilityTOend</incoming>
          </endEvent>
          <sequenceFlow id="startTOrequirement" name=""
              sourceRef="startAnalisys" targetRef="requirementAnalisys"/>
          <sequenceFlow id="requirementTOfeasibility" name=""
              sourceRef="requirementAnalisys" targetRef="feasibilityStudy"/>
          <sequenceFlow id="feasibilityTOend" name=""
              sourceRef="feasibilityStudy" targetRef="endAnalisys"/>
        </process>
    </definitions>
```

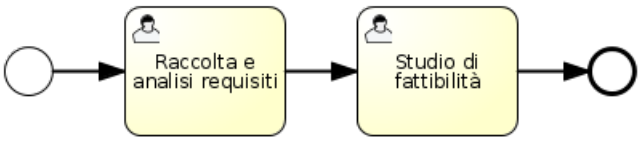


Figura 5.7: *BPD Analisi*

5.7 Sottoprocesso di sviluppo del progetto

```
<?xml version="1.0" encoding="UTF-8"?>
    <definitions xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
        xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
        xmlns:omgdc="http://www.omg.org/spec/DD/20100524/DC"
        xmlns:omgdi="http://www.omg.org/spec/DD/20100524/DI"
```

```
    xmlns:signavio="http://www.signavio.com"
    xmlns:activiti="http://activiti.org/bpmn"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    id="sid-6866f12a-dfaa-4d4a-9122-5f528b5fa2a3"
    targetNamespace="http://activiti.org/bpmn20"
    typeLanguage="http://www.w3.org/2001/XMLSchema"
    xsi:schemaLocation="http://www.omg.org/spec/BPMN/20100524/MODEL
        http://www.omg.org/spec/BPMN/2.0/20100501/BPMN20.xsd">
    <process id="projectDevelopmentProcess" name="Sviluppo progetto">
      <startEvent id="startDevelopment" name="">
        <outgoing>startTodesign</outgoing>
      </startEvent>
      <userTask id="design" name="Progettazione"
          activiti:candidateGroups="accountancy">
        <incoming>startTodesign</incoming>
        <outgoing>designTOrealization</outgoing>
      </userTask>
      <userTask id="realization" name="Realizzazione"
          activiti:candidateGroups="accountancy">
        <incoming>designTOrealization</incoming>
        <outgoing>realizationTOtest</outgoing>
      </userTask>
      <userTask id="test" name="Validazione e collaudo"
          activiti:candidateGroups="accountancy">
        <incoming>realizationTOtest</incoming>
        <outgoing>testTOendDevelopment</outgoing>
      </userTask>
      <endEvent id="endDevelopment" name="">
        <incoming>testTOendDevelopment</incoming>
      </endEvent>
      <sequenceFlow id="designTOrealization" name=""
          sourceRef="design" targetRef="realization"/>
      <sequenceFlow id="realizationTOtest" name=""
          sourceRef="realization" targetRef="test"/>
      <sequenceFlow id="testTOendDevelopment" name=""
          sourceRef="test" targetRef="endDevelopment"/>
      <sequenceFlow id="startTodesign" name=""
          sourceRef="startDevelopment" targetRef="design"/>
    </process>
</definitions>
```

Figura 5.8: *BPD Sviluppo Progetto*

A questo punto bisogna mettere in funzione il tutto. Per farlo Activiti Probe mette a disposizione una funzionalità apposita, detta **Deployments**. È possibile caricare un file alla volta, compresi i form, oppure creare un file .zip che li raggruppa tutti. Activiti Probe è anche in grado di leggere file compressi in formato .bar, ottenibili semplicemente modificando l'estensione di un file .zip. In questo caso, si mette in funzione il file test.zip come rappresentato in figura 5.9:

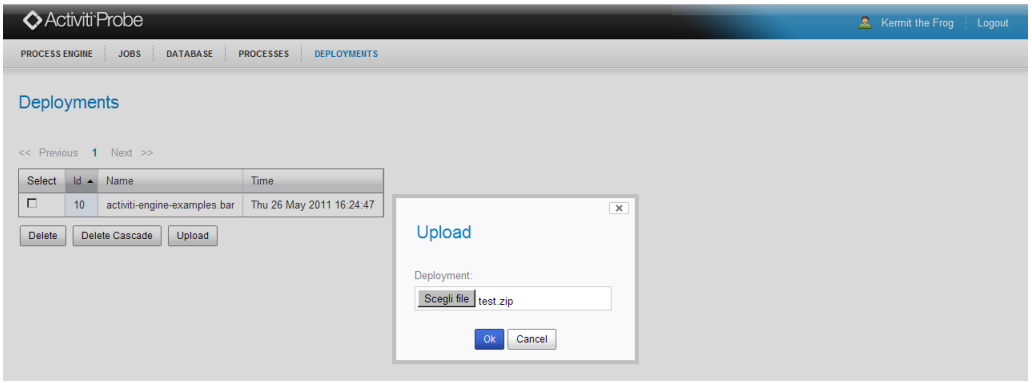


Figura 5.9: *Deployment di un processo con Activiti Probe*

Ora troveremo nel database tutto quanto riguarda le nuove definizioni di processo appena caricate. Per verificare basta entrare nella sezione database e sfogliare tra le varie tabelle.Accedendo ad Activiti Explorer si può invece eseguire il processo, cliccando sul bottone "Start instance" relativo al processo che si vuole far partire; eseguendo il processo principale, "Processo di gestione dell'offerta", verranno automaticamente eseguiti anche i suoi sottprocessi qualora il flusso del processo lo richieda. Se fosse necessario, si ha la possibilità di eseguire separatamente ogni singolo sottoprocesso.

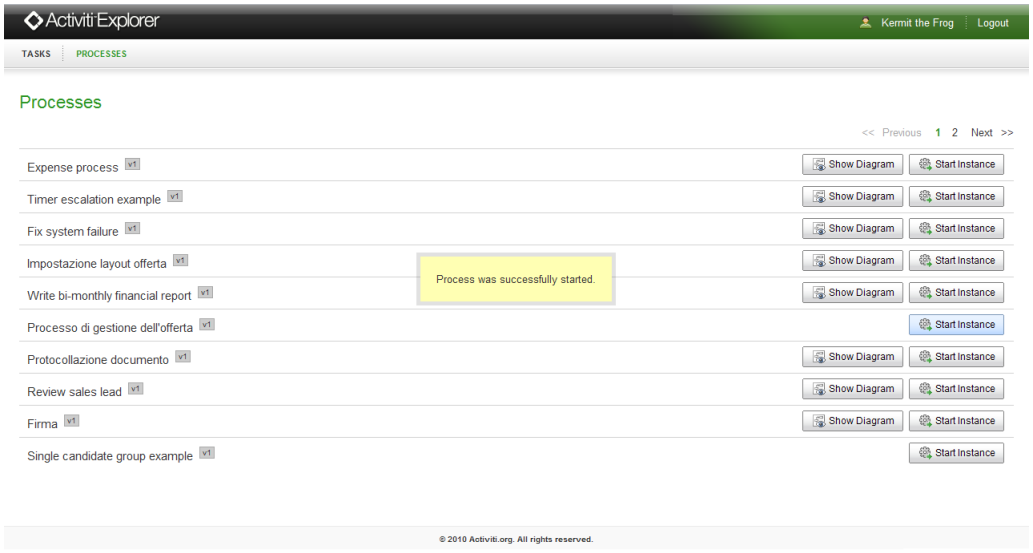


Figura 5.10: *Esecuzione di un processo con Activiti Explorer*

Nella sezione Tasks, tra quelli non assegnati, si troverà il primo task del processo appena avviato ed eventualmente tutti i task non assegnati, di altri processi precedentemente avviati.
Per ciò che riguarda le call activity denominate "*Processo approvvigionamenti*" e "*Processo amministrativo*" (presenti nel processo di gestione dell'offerta), non vengono fornite le rispettive definizioni di processo, poichè sono state modellate solo a titolo d'esempio, inserendo molto semplicemente uno start event seguito da un paio di user task e un end event. Questo perchè il tempo a disposizione non sarebbe stato sufficiente, dal momento che tali processi avrebbero meritato un'analisi separata e più approfondita.

6 Integrazione Alfresco Share con Activiti

L'ultimo passo è l'integrazione dei due strumenti precedentemente descritti.
Recentemente la community di sviluppo di Alfresco e quella di Activiti, si sono unite proprio per portare avanti un progetto in questa direzione. L'ultima versione di Alfresco Community, la 3.4.e, infatti comprende uno strumento per la creazione e gestione degli workflow, associati ai documenti contenuti nel repository, basato sul process engine di Activiti.
Anche con le versioni precedenti dei due software è possibile ottenere un'integrazione, scrivendo una process definition in BPMN 2.0, tramite Activiti, e caricandola poi nel repository di Alfresco, seguendo una procedura che renderà visibile la definizione tra quelle base di Alfresco, disponibili di default.
Alfresco individuerà, all'interno della definizione, quali sono i task demandati all'utente e quali i task che verranno eseguiti in background, come ad esempio quelli che invocano parti di codice Java o che dialogano con web services.

6.1 Sviluppi futuri

A partire dalla nuova versione di Activiti c'è una variazione fondamentale nel linguaggio di riferimento, ovvero il passaggio da BPMN 2.0 a BPEL (Business Process Execution Language), una combinazione tra WSDL (Web Service Definition Language) e XML. Il concetto di business process come insieme di attività automatizzate e manuali, in questo modo viene rappresentato adeguatamente.
Lo svantaggio è che ciò richiede una conversione di tutto quanto creato precedentemente. Poichè lo strumento di conversione automatica è presente solo nella versione Enterprise di Alfresco, quest'ultimo passo rimane da completare. Il tempo rimasto è stato però sufficiente per iniziare a documentarsi sui linguaggi necessari a implementare i processi aziendali con la nuova tecnologia, della quale si da una panoramica nei prossimi paragrafi.

6.1.1 WSDL - Web Service Definition Language

> *WSDL si definisce un formato XML per descrivere servizi di rete come un insieme di punti terminali operanti su messaggi contenenti informazione di tipo "documentale" o "procedurale".*

WSDL è utilizzato per definire interfacce tramite le quali i client sulla rete possono richiedere un servizio ad un web service posto sul web. L'interazione avviene tramite lo scambio di messaggi, definiti dal protocollo SOAP e da XML Schema per il type system. Un documento WSDL contiene le informazioni sulle operazioni fornite dal servizio, sui protocolli da utilizzare per accedere al servizio, su come definire i messaggi in input e in output e per i relativi dati e infine sull'indirizzo URI al quale si accede al servizio.
WSDL permette di separare gli aspetti astratti da quelli concreti introducendo i seguenti vantaggi:

- lo stesso servizio può avere implementazioni differenti basandosi sulla stessa descrizione astratta;
- le descrizioni astratte possono essere riutilizzate, interamente o in parte, nella creazione di nuovi servizi.

Nel livello astratto, il servizio viene definito riferendosi alle interfacce e al tipo di mesaggi scambiati per ciascuna di esse; nel livello concreto le descrizioni astratte vengono istanziate legandole a una implementazione reale.
La **descrizione astratta** è composta:

1. dalla definizione del *tipo* di *messaggi scambiati* tra client e server durante la fruizione del servizio;
2. dalla definizione delle *operazioni* tramite le quali si interagisce col servizio;
3. dalla definizione delle *porte astratte*, insiemi di operazioni (interfacce) esposte dal servizio.

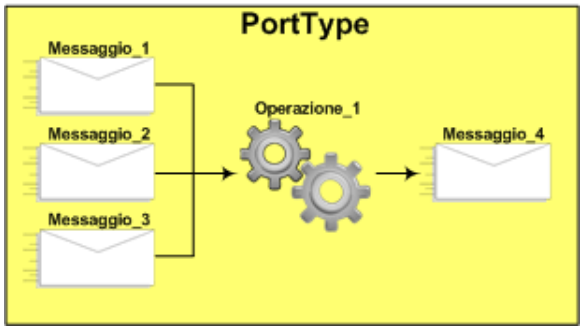


Figura 6.1: *Rappresentazione di una porta astratta, che definisce il tipo di servizio offerto e per questo chiamata* **portType**.

La **descrizione concreta** è composta:

1. dai *binding* che legano ciascuna porta astratta a un protocollo e ad un formato dati per i messaggi scambiati;

2. dalle *porte*, che legano i portType a indirizzi di rete reali (WSDL vede i web services come insiemi di terminali di rete, detti **porte**);

3. dai *servizi*, che spesso permettono di accedere allo stesso servizio secondo diverse modalità.

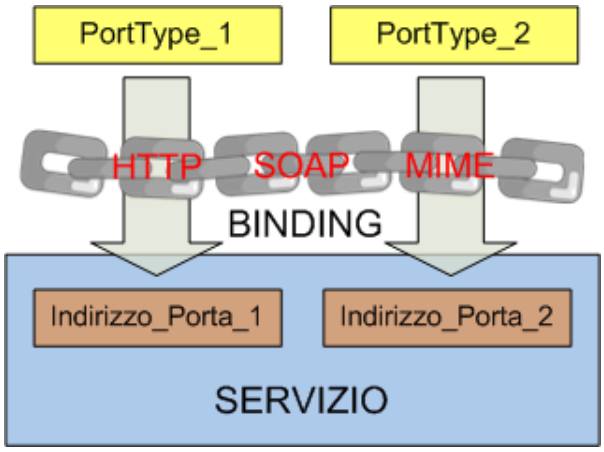


Figura 6.2: *Il* **binding** *definisce il tipo di collegamento tra un portType e un indirizzo di rete reale.*

Ogni messaggio scambiato nell'ambito delle operazioni WSDL deve essere tipizzato, come detto precedentemente, utilizzando il type system XML Schema.

Per definire i tipi si usa l'elemento *types*, all'interno del quale può essere inserito un intero schema XML, o qualunque altra definizione XML valida. Si possono definire due tipi fondamentali:

1. *simpleType;*
2. *complexType.*

L'elemento fondamentale per la creazione di un web service sono i **messaggi**, descritti dall'oggetto *message*, ognuno dei quali identificato da un nome univoco. I messaggi sono costituiti da una o più parti, ciascuna con un nome e un tipo distinti, il quale si riferisce ad un tipo definito o importato nella sezione *types*.
Le funzionalità esposte dall'interfaccia del servizio, le **operazioni**, vengono definite dall'oggetto *operation*, all'interno del quale si devono definire gli oggetti *input*, *output* e *fault* che rappresentano i messaggi scambiati durante l'operazione. L'oggetto *operation* è contenuto nell'oggetto portType.
WSDL definisce quattro tipi di operazioni:

1. **one-way**: il client spedisce un messaggio al servizio;
2. **request-response**: il client spedisce un messaggio al servizio e riceve una risosta;
3. **notification**: il servizio invia un messaggio al client;
4. **solicit-response**: il servizio invia un messaggio a un client e questo risponde.

Completata la definizione di un *portType*, si deve istanziare la porta astratta utilizzando l'oggetto *binding*, nel quale vanno inseriti l'attributo *type* che si riferisce al *portType* istanziato e il tipo di binding utilizzato, ossia a quale protocollo si fa riferimento per lo scambio di messaggi. I tipi di binding opssibili sono: SOAP, HTTP oppure MIME (Multipurpose Internet Mail Extensions).
Ad esempio se si tratta di un SOAP binding, si deve utilizzare l'elemento *soap:binding*, all'interno del quale dovranno essere specificati gli attributi *style*, per lo **stile di codifica**, e *transport*, per il **tipo di trasporto** utilizzato.
In ciascuna *operation* dovrà essere inserito un elemento *soap:operation*, che specifica lo stile di codifica dell'operazione (solo se diverso da quello dichiarato nel *soap:binding*) mediante l'attributo *style*, e la **SOAPAction** associata, solo se il trasporto è HTTP, mediante l'attributo *soapAction*.
Per una completa definizione di un messaggio si dovranno dichiarare quali informazioni verranno mappate nell'header del messaggio e quali nel body.

Il contenuto del corpo del messaggio SOAP è definito dall'elemento *soap:body*, caratterizzato dagli attributi:

- *parts*, che descrive quali parti aggiungere al corpo;
- *use* e *encodingStyle*, che descrivono la codifica dei tipi;
- *namespace*, che specifica il namespace di provenienza delle parti.

Eventualmente, utilizzando l'elemento *soap:header*, si mappano parti di messaggio nell'-header, specificando gli attributi:

- *message*, che indica il messaggio di riferimento;
- *part*, che indica la parte da includere;
- *use*, *econdingStyle* e *namespace*

Se necessario si possono specificare anche i messaggi che saranno restituiti in caso di errore nell'eleborazione del body o dell'header:

- *soap:fault*, ha la stessa sintassi di *soap:body* e va incluso nell'oggetto fault associato alla relativa *operation*;
- *soap:headerfault*, ha la stessa sintassi di *soap:header* e va incluso negli elementi *soap:header* stessi.

Riassumendo, un esempio completo di un documento WSDL per la descrizione di un web service assume la forma seguente:

```
<definitions name="nome_servizio">
  <!- Lista dei tipi utilizzati ->
  <types>
    ...
  </types>
  <!- Lista dei messaggi ->
  <message name="nome_messaggio">
    <part name="nome_parte" type="nome_tipo"/>
  </message>
  <!- Lista delle porte astratte ->
  <portType name="nome_tipo_servizio">
    <!- Elenco delle operation associate ->
    <operation name="nome_operazione">
      <input name="nome_messaggio_input" message="nome_messaggio_di_riferimento" />
      <output name="nome_messaggio_output" message="nome_messaggio_di_riferimento" />
    </operation>
  </portType>
```

```
  <!- Lista dei binding ->
  <binding name="nome_tipo_servizio" type="nome_portType_istanziato">
    <soap:binding style="rpc/document" transport="http://schemas.xmlsoap.org/soap/http" />
    <operation name="nome_operazione">
      <soap:operation soapAction="nome_operazione_di_riferimento" style="rpc/document" />
        <input name="nome_messaggio_input">
          <soap:body use="encoded" namespace="..."
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
        </input>
        <output name="nome_messaggio_output">
          <soap:body use="encoded" namespace="..."
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
        </output>
    </operation>
  </binding>
  <!- Lista dei servizi ->
  <service name="nome_servizio">
    <port name="nome_port" binding="nome_binding_di_riferimento">
      <soap:address location="indirizzo_porta"/>
    </port>
  </service>
</definitions>
```

Esempio 6.1.1: *esempio di definizione di un web service con WSDL.*

6.1.2 BPEL - Buiness Process Execution Languge

> *BPEL è un linguaggio basato sull'XML, costruito per descrivere formalmente i processi commerciali e industriali, in modo da permettere una suddivisione dei compiti tra attori diversi.*

Lo standard che definisce l'uso di BPEL nelle interazioni tra web services definiti in WSDL, è **BPEL4WS** (*BPEL for Web Service*). L'interazione è astratta, nel senso che avviene tra portType e non tra le porte vere e proprie.
BPEL permette di descrivere un business process, come BPMN, mediante una serie di **attività semplici** (invoke, receive, reply, assign, throw, wait, empty, exit) o **strutturate** (if, while, repeatUntil, foreach, pick, flow, sequence, scope). È adatto soprattutto per la rappresentazione di processi completamente auotmatizzati, ma grazie all'estensione *BPEL4People* è possibile rappresentare anche attività che richiedono l'interazione umana, utilizzando un nuovo tipo di attività semplice detta *humanTask*.
Alcune delle sue caratteristiche principali sono:

- fornitura di un modello dei dati universale per lo scambio di messaggi;
- gestione dei dati per una corretta interazione tra i vari attori;
- supporto dell'esecuzione parallela delle attività;

- gestione delle eccezzioni ed implementazione di logiche di compensazione.

Per creare un processo BPEL è necessario disporre dei documenti WSDL dei servizi da invocare e del processo stesso, poichè ogni WSDL ci consente di creare un *partnerLink*, il quale può implementare due ruoli: *myRole* per identificare il processo stesso, e *partnerRole* per indicare il ruolo di un partecipante esterno.

6.1.2.1 Attività semplici

Receive La receive è l'attività che ha il compito di istanziare il processo. Viene associata ad un elemento partnerLink, ovvero ciò che rappresenta un partecipante al processo, ad un'operazione e ad una variabile di input. Esegue una wait che attende l'arrivo di una richiesta (messaggio).

Reply L'attività di reply, anch'essa associata ad un partnerLink, un'operazione ed una variabile di output, può dare una risposta normale oppure una risposta con fault. Viene utilizzata nei processi sincroni, per rispondere alla richiesta giunta tramite la receive. Tutti i processi sincroni prevedono la combinazione delle attività receive e reply.

Invoke L'attività di invoke è usata per chiamare una determinata operazione di un partnerLink. È associata ad un partnerLink, un'operazione, una variabile di input e una variabile di output, le quali devono essere dello stesso tipo di input/output gestiti dalle rispettive operazioni.

Assign Attività che consente di copiare, mediante l'operazione *copy* (costituita da un *from* e da un *to*), il contenuto di una variabile semplice o di campi di variabili complesse, all'interno di un'altra variabile o campo di variabile. L'individuazione del dato da copiare può avvenire in due modi:

- *semplice*, per copiare dati tra variabili o tra parti di variabili, dello stesso tipo:

```
<assign name="nome">
  <copy>
    <from variabile="input"/>
    <to variabile="output"/>
  </copy>
</assign>
```

- *XPath*, per copiare dati tra campi, dello stesso tipo, di variabili complesse

```
<assign name="nome">
  <copy>
    <from>$input.part1/ns0:nome</from>
    <to>$output.part1/ns1:nome</to>
  </copy>
</assign>
```

I dati possono essere assegnati anche in modo misto, ad esempio utilizzando il modo semplice nel tag *from* e il modo XPath nel tag *to*. Per assegnare il valore di una variabile in modo manuale si utilizza il costrutto <*literal*>.

6.1.2.2 Attività strutturate

Sequence Consente l'esecuzione sequenziale delle attività definite al suo interno.

Flow Consente l'esecuzione parallela delle attività definite al suo interno, utilizzando i costrutti *fork* e *join*.

While e repeatUntil Permettono l'esecuzione ciclica di attività, basate su condizioni booleane.

If Implementa il costrutto if-then-else, permettendo l'esecuzione condizionata delle attività definite al suo interno.

Scope Costrutto che crea gruppi di attività, tra le quali si identifica un'attività primaria (sequence o flow).
All'interno di uno scope si possono definire i cosiddetti *faultHandlers*, ciascuno dei quali conterrà uno o più costrutti *catch*, che catturano eventi specifici, ed al più un costrutto *catchAll*, con lo scopo di catturare eccezioni di ogni. Ovviamente il catch è sempre definito in seguito ad un *throw*.

Throw Il costrutto throw ha il compito di gestire un eventuale errore nel processo BPEL, causato dal lancio esplicito di un'eccezione, da una chiamata errata ad un servizio esterno, oppure da errori interni all'ambiente di esecuzione.

Un esempio di processo di business descritto in BPEL è il seguente:

```
<?xml version = "1.0" encoding = "UTF-8" ?>
<process name="BPELProva" targetNamespace="..."
        xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/">
    <partnerLinks>
      <partnerLink name="client" partnerLinkType="client:BPELProva"
        myRole="BPELProvaErogatore" partnerRole="BPELProvaRichiedente" />
    </partnerLinks>
    <variables>
      < variable name="input" messageType="client:BPELProvaRequestMessage" />
      < variable name="output" messageType="client:BPELProvaResponseMessage" />
    </variables>
    <sequence name="main">
      <receive name="receiveInput" partnerLink="client" portType="client:BPELProva"
          operation="start" variable="input" createInstance="yes" />
      <invoke name="callbackClient" partnerLink="client" portType="client:BPELProvaCallback"
          operation="finish" input="output" />
    </sequence>
</process>
```

Esempio 6.1.2: esempio di processo BPEL

7 Conclusioni

La fase di test, come era prevedibile, verrà effettuata solo su una parte di quanto creato, ossia sul processo *Firma.bpmn20.xml*, poichè la messa in esecuzione del processo completo sarebbe troppo complessa e avrebbe un impatto troppo forte sull'organizzazione aziendale. In più si rende necessario un periodo di formazione del personale, per imparare ad utilizzare i nuovi strumenti in modo efficiente.
In questo progetto tutti i task sono stati modellati come *userTask*, così da rendere possibile, attraverso Activiti Explorer, l'esecuzione del processo completo. Alcuni di questi, in realtà, sono task che si potrebbero automatizzare, ad esempio scrivendo degli script o del codice java, invocandoli rispettivamente tramite *scriptTask* e *javaServiceTask*. In questo modo si raggiungerebbe uno degli obiettivi della dematerializzazione, cioè quello di semplificazione dei processi aziendali attraverso una riduzione delle fasi che li compongono.
A livello operativo, il personale dell'azienda che viene coinvolto dal processo aziendale lavorerà con Alfresco Share, mentre solo gli amministratori e gli incaricati al monitoraggio del processo lavoreranno anche con Activiti.
Un ulteriore obiettivo che resta da raggiungere è quello riguardante la visualizzazione dei metadati di un documento. Si vorrebbe che un documento associato ad un processo di business, venisse visualizzato insieme ai suoi metadati per rendere più diretta la lettura di alcune informazioni in esso contenute, come ad esempio il suo numero di protocollo. Alfresco Share supporta la visualizzazione dei metadati di un documento, tipo autore, date di creazione e modifica, e altre informazioni, ma non riferiti al suo contenuto. Fortunatamente il fatto che sia open-source, permette di modificarne il codice, aggiungendo una funzionalità (per esempio scritta in Java) che estrae dal documento, e rende visibili, le sue informazioni identificative, al momento del suo caricamento nel repository. Un esempio pratico potrebbe riguardare un documento di autorizzazione preventiva per un'attività fuori sede. Alcuni metadati potrebbero essere la matricola del richiedente, il numero e la data della commessa, gli estremi dell'attività da svolgere (ente ospitante, località, date di inizio e fine), il mezzo di trasporto con cui il richiedente si muoverà e altre informazioni.

Detto ciò, il processo di dematerializzazione è un processo complesso che richiede tempo e risorse per essere completato, perchè l'utilizzo di una piattaforma come quella descritta, necessita di numerosi adattamenti e personalizzazioni.
Una volta avviato il processo di dematerializzazione, si potranno notare dei piccoli vantaggi che con il tempo diventeranno sempre più rilevanti. Questi non saranno solo a livello economico per il risparmio di materiale cartaceo, ma anche a livello temporale, in quanto i tempi di esecuzione dei processi di business si ridurrà notevolmente. Inizialmente, la produzione di documenti digitali potrebbe causare un aumento dei costi, il quale non dev'essere considerato come un fattore negativo, come possono essere invece la perdita di autenticità di un documento. Infatti la dematerializzazione, per produrre risultati consistenti, deve includere il controllo sulla corretta formazione del documento e il controllo del suo intero ciclo di vita, compresa la sua conservazione.

Ringraziamenti

Dopo essere stato ad un passo dal lasciare gli studi, eccomi qua a ringraziare chi mi ha convinto a non farlo, e non solo.
Desidero ringraziare innanzitutto i miei genitori, Claudio e Sonia, che mi hanno supportato economicamente e soprattutto moralmente, nei momenti difficili che si sono presentati nel mio percorso universitario, con la loro infinita pazienza. Spero che questo traguardo sia motivo di soddisfazione, oltre che per me, anche per loro. Vorrei ringraziare anche tutti i parenti che mi sono stati vicino in questi anni, non facendomi mai mancare il loro affetto.
Ringrazio il Professor Federico Filira che ha accettato, nonostante i numerosi laureandi che doveva seguire, di essere il mio relatore.
Un grazie alla "Venis - Venezia Informatica e Sistemi S.p.A." tutta, per avermi dato la possibilità di svolgere il tirocinio presso le loro strutture, per i consigli che mi saranno senz'altro utili in futuro, e per la disponibilità e gentilezza dimostrate nei miei confronti.
Per evitare di dimenticare qualcuno, rivolgo un grazie speciale a TUTTI gli amici della "compagnia" e non, con i quali ho passato bellissimi momenti, e che, forse senza accorgersene, mi hanno aiutato ad arrivare fin qua.
Infine, grazie anche a tutti coloro che ho conosciuto durante gli studi, con i quali ho condiviso "gioie e dolori" fin dal primo anno di Università.

Bibliografia

[1] David Caruana, John Newton, Michael Farman, Michael G. Uzquiano, Kevin Roast (Maggio 2010), "*Professional Alfresco-Practical Solution for Enterprise Content Management*", Paperback, London

[2] Business Process Modeling and Notation (BPMN), `http://www.omg.org/spec/BPMN/2.0`

[3] Martin Owen, Jog Raj, "BPMN and Business Process Management - Introducing to the New Business Process Modeling Standard", Popkin Software

[4] Giuseppe Della Penna, Università degli Studi di L'Aquila, "Web Service Definition Language (WSDL)", `http://dellapenna.univaq.it`

[5] Stefano Iannucci, Università degli Studi di Roma "Tor Vergata", "Processi BPEL"

[6] Cesare Pautasso, Università di Lugano, "BPEL for REST"

[7] Tijs Rademakers, Ron van Liempd, "Activiti in action, Executable business process in BPMN 2.0", MEAP-Manning Early Access Program Edition

[8] Fabio Adezio, "Il software Open Source per la gestione documentale"

[9] Daniele Canini, "Un modello dinamico per la descrizione del processo di dematerializzazione nella pubblica amministrazione italiana"

[10] Maria Guercio, "La dematerializzazione dei documenti - l'efficienza del digitale e i suoi limiti"

[11] `http://activiti.org`

[12] `http://www.alfresco.com`

[13] `http://docs.alfresco.com/3.4/index.jsp`

[14] `http://www.column2.com/2010/05/open-source-bpm-with-alfrescos-activiti/`