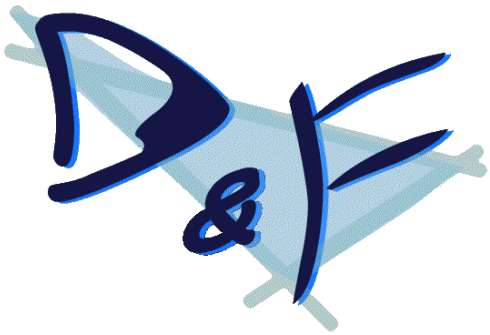
TESI TECNICA

ITIS P. PALEOCAPA

Workflow Management System



Autori

ARMATO DESIRÈE

CIGLIANO FABIO

A.S. 2006-07

Workflow Management System

di

Armato Desirèe

Cigliano Fabio

SOMMARIO

1. Introduzione

I **workflow management system** (WfMS) sono applicazioni che gestiscono flussi di lavoro. Qualsiasi azienda, amministrazione pubblica o ente commerciale segue dei protocolli per gestire processi produttivi o amministrativi basati sul passaggio di informazioni o documenti: questi iter possono essere schematizzati come flussi di lavoro a cui si riferiscono i dipendenti e/o gli utenti.

Nella gestione tipica di questi flussi di lavoro molte energie vengono spese inutilmente in due ambiti che non portano alcun risultato pratico. Da una parte si deve curare il passaggio del lavoro o dei documenti, perdendo tempo e risorse nel cercare di capire che cosa si debba fare di un certo oggetto o a quale ufficio debba essere mandato un incartamento. Dall'altra parte molte delle attività svolte sono ripetitive e automaticamente gestibili da un'applicazione software anziché da un essere umano.

I WfMS si propongono di automatizzare queste fasi dispendiose e non redditizie dei flussi di lavoro con l'aiuto delle tecnologie informatiche. Il guadagno in efficienza portato dai WfMS è proporzionale alla complessità del processo modellato e alla percentuale di attività che possono essere rese automatiche di cui è composto.

Il progetto affrontato si propone di sviluppare un WfMS che sia compatibile con gli standard in atto e di semplice utilizzo.

Nel **primo capitolo** si affrontano le questioni principali del progetto: i workflow. Si spiega cosa sono i workflow, i WfMS, come e dove vengono impiegati e quali sono i problemi che li caratterizzano.

Nel **secondo capitolo** si definiscono gli obiettivi del progetto. Si presenta il modello di riferimento per i WfMS seguito per il progetto, si delineano le applicazioni software realizzate, infine si danno alcune linee guida seguite nello sviluppo.

Nel **terzo capitolo** si introducono gli strumenti software utilizzati per il progetto: il workflow editor Jawe e il linguaggio Xpdl, l' ambiente AMP XAMPP e il linguaggio PHP. Jawe viene presentato mostrandone le caratteristiche salienti. Di XAMPP se ne descrive la struttura, il funzionamento e le caratteristiche che lo rendono adatto per il progetto.

Nel **quarto capitolo** si passa ad affrontare l'analisi del progetto. Dopo aver introdotto i concetti fondamentali, si descrive il progetto con degli use case che ne sintetizzano le funzionalità previste.

2. Workflow

In questo capitolo si introduce il concetto di workflow: si spiega cosa sono i workflow e come si classificano, passando poi ad introdurre i workflow management system.

2.1. Cosa sono i workflow

Si definisce il **workflow** come "*l'automazione di una parte o dell'intero processo commerciale, dove documenti, informazioni e compiti vengono passati da un attore ad un altro per ricevere un qualche tipo di azione, seguendo un determinato insieme di regole*".

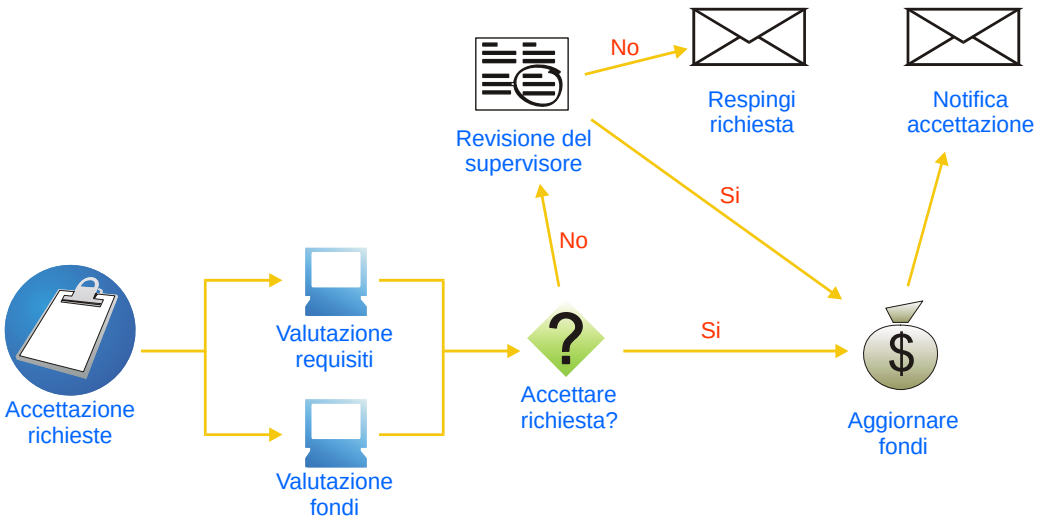


Figura 1: un esempio di processo

Dunque un workflow riassume una serie di procedure che definiscono un contesto di lavoro complesso, dove più unità lavorative, siano esse risorse umane, gruppi di lavoro o sistemi computerizzati, lavorano per un determinato fine. Ogni workflow è dunque composto di un certo numero di **processi**, cioè procedure da seguire per ottenere un certo risultato da determinate condizioni di partenza.

Ogni processo si può definire attraverso la sequenza di azioni che devono essere svolte dai partecipanti, e attraverso l'evoluzione dello stato degli oggetti che lo attraversano. Un

processo, ad esempio, può essere l'iter per una richiesta di acquisto dell'università, o le procedure per ottenere una licenza edilizia nell'ambito della pubblica amministrazione (Figura 1)

Le azioni che devono essere svolte nel processo sono chiamate **attività** del processo. Ogni processo viene descritto dall'insieme di attività e dalla sequenza con cui vengono svolte.

Un'attività descrive un singolo passo del processo da eseguire: ad esempio prendere una decisione, apporre una firma ad un documento, o stampare un elenco. Il processo ha sempre una sola attività di inizio, in cui "entrano" gli oggetti che devono essere elaborati, ed un'attività di fine da cui "escono" i prodotti del processo.

La sequenza di passi di un processo non è limitata ad essere una semplice sequenza lineare. Si possono infatti avere ramificazioni e ricongiungimenti, a rappresentare attività che devono essere svolte in parallelo o attività che devono essere svolte in alternativa le une alle altre. Ad esempio in un'amministrazione comunale un certo incartamento può intraprendere un percorso attraverso vari uffici, trovarsi ad essere elaborato contemporaneamente da due uffici di controllo, per poi dover scegliere se essere mandato fra le pratiche urgenti o fra quelle a bassa priorità, e così via.

Ogni processo è un'entità statica, una sorta di definizione di procedure e iter da intraprendere: può essere eseguito più volte, cioè può avere più istanze, anche contemporanee.

Ogni **istanza di processo** ha una propria identità distinta, anche se condivide con le altre istanze di processo la stessa sequenza di azioni da intraprendere. Un'istanza di processo è dunque un'entità dinamica: ha una data e orario di nascita, dati registrati che cambiano di attività in attività e, infine, ha anche una fine: la sua archiviazione.

2.2. Tipi di workflow

I workflow sono divisi in tre tipi per funzioni e capacità: produttivi, amministrativi e collaborativi.

I workflow **produttivi** sono orientati alla produzione su grande scala. Gestiscono grossi numeri di attività simili, ripetitive, prevedibili e il loro fine è ottimizzare la produttività intesa come pezzi per unità di tempo. Un esempio di workflow produttivo può essere il processo di richiesta di prestito alle banche o la presentazione di reclami alle assicurazioni. Tutti i

processi del workflow sono orientati ad avere un alto livello di automazione, per trarre il massimo vantaggio dall'assenza di componente umana.

I workflow **amministrativi** sono caratterizzati dall'avere facilità nel definire processi diversi. Spesso si possono avere processi molto eterogenei che evolvono secondo le necessità e l'esperienza acquisita nell'utilizzo. Ad esempio il workflow relativo alla gestione di uffici comunali, dove i processi devono seguire l'evoluzione delle normative ministeriali. In questi workflow la flessibilità è molto più importante della produttività.

I workflow **collaborativi** si focalizzano sugli strumenti di comunicazione e scambio di informazioni fra gruppi di lavoro (grandi o piccoli che siano). Non c'è necessità di avere un'alta produttività e i processi sono tipicamente solo vagamente definiti. Quel che conta è avere strumenti di confronto e condivisione. Come esempio si può prendere l'iter di risoluzione dei conflitti, o le procedure di negoziazione. Portato questo concetto agli estremi si ha un workflow caratterizzato dalla più bassa produttività e dalla massima flessibilità nel definire procedure: siamo nella situazione in cui ogni caso di lavoro richiede una procedura diversa, creata ad hoc.

L'ordine con cui sono stati presentati i tipi di workflow (produttivo, amministrativo e collaborativo) rappresenta anche l'ordine decrescente del grado di efficienza che il workflow permette di raggiungere. Infatti il massimo guadagno di efficienza viene ottenuto in quei processi quasi completamente automatizzabili, come i processi produttivi. Via via che aumentiamo l'impatto delle risorse umane e della flessibilità dei processi, andiamo a perdere sempre più uno dei principali vantaggi del workflow: l'automazione delle attività. Flessibile, infatti, significa mutevole, non formalizzato, non definitivo e dunque non formalmente modellabile come attività automatica.

2.3. Cosa sono i workflow management system

Si definisce il workflow management system (WfMS) come "*un sistema che definisce, crea e gestisce l'esecuzione di workflow attraverso l'uso di software, coinvolgendo uno o più motori di workflow; e che è in grado di interpretare definizioni di processo, interagire con i partecipanti del workflow e, se richiesto, invocare l'uso di applicazioni e strumenti dell'information technology*".

L’utilizzo di strumenti di workflow nell’ambito industriale e/o amministrativo è sempre più diffuso, specie se si ha a che fare con processi coinvolgenti le nuove tecnologie. Questo perché l’uso di workflow porta due diversi vantaggi: da una parte si assegna efficientemente il lavoro a chi di dovere, dall’altra si automatizza il lavoro stesso.

La questione dell’**assegnare i compiti** si propone di risolvere le questioni del tipo “chi deve fare cosa e quando”. Ogni lavoratore all’interno del workflow sa che, quando gli viene sottoposto un lavoro, avrà a disposizione tutte le informazioni e gli strumenti necessari al compimento della sua attività. Non gli mancherà nessuna delle componenti di cui avrà bisogno. Questa gestione del lavoro può produrre grossi incrementi d’efficienza.

L’**automazione del lavoro** produce un ancor maggiore guadagno d’efficienza. Quelle attività che sono puramente meccaniche e automatiche, che non hanno bisogno di alcuna componente umana nello svolgimento, possono essere affidate agli strumenti di esecuzione sotto il controllo diretto del workflow. Inviare una e-mail di conferma, stampare un certificato, aggiornare un data-base sono azioni automatiche che non hanno bisogno dell’uso della risorsa più costosa: l’uomo. Inoltre le attività automatiche godono della possibilità di poter essere più facilmente svolte in parallelo, il che comporta ulteriori vantaggi.

A fianco di questi vantaggi tangibili vi sono anche tutta una serie di vantaggi che a prima vista possono sfuggire. Infatti con l’utilizzo di workflow:

- si ha una maggiore qualità del servizio, dovuta ai minore errori nelle attività svolte automaticamente e al maggiore tempo disponibile per le attività non automatiche;

- si semplifica la pianificazione e si possono prendere decisioni più oculate, potendo monitorare più efficientemente e profondamente l’andamento dei processi del workflow;

- si valorizzano le risorse umane, giacché gli aspetti più ripetitivi e monotoni dell’attività del processo vengono svolte automaticamente.

3. Obiettivi

Il nostro obiettivo è realizzare un'applicazione per gestire un flusso di lavoro il più possibile automatizzata; secondo quest'ottica gli utenti che partecipano al workflow dovranno esclusivamente preoccuparsi della loro attività da svolgere. Tutto questo è permesso da un sistema mail che li avviserà dell'inizio del loro compito e del tempo che hanno a disposizione, ma anche da un responsabile che, in caso di ritardi o errori particolari del sistema, provveda a far ripartire il processo.

Per rendere più chiaro questo meccanismo di automazione abbiamo citato la struttura abituale di un WfMS nella prima sezione di questo capitolo.

Nella seconda sezione si definiscono gli strumenti software utilizzati e alcuni particolari del progetto realizzato.

Nella terza sezione si definiscono le linee guida seguite nello sviluppo del progetto.

3.1. Modello di riferimento

Nel seguente modello di riferimento si distinguono tre diverse aree funzionali (Figura 2):

- **progettazione**: rappresenta l'insieme delle funzionalità che descrivono i processi da trattare. In fase di progettazione perciò si compone la definizione del processo di lavoro. In concreto rappresenta il file dove verranno memorizzate la attività del processo.
- **run-time**: rappresenta l'insieme delle funzionalità che gestiscono l'evoluzione nel tempo del flusso di lavoro. A tempo di esecuzione queste funzioni fanno riferimento alla definizione del processo effettuata in fase di progettazione.
- **interfaccia**: rappresenta l'insieme delle funzionalità preposte all'interazione con l'utente e il WfMS. E' rappresentato dall'interfaccia amministrativa (CMS) nel nostro specifico applicativo.

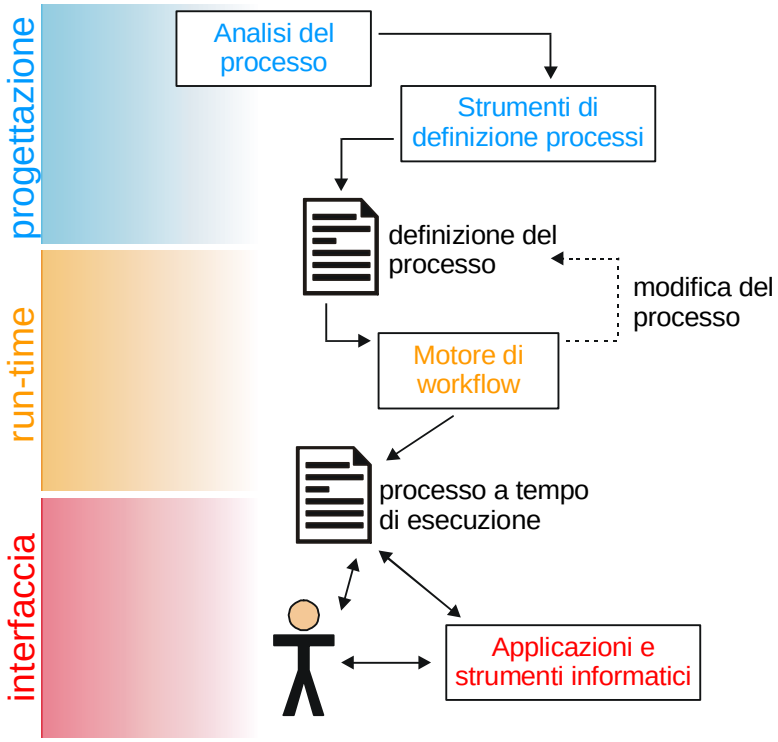


Figura 2: aree funzionali dei WfMS

3.1.1. Strumento di definizione di processi

Lo strumento di definizione di processi può essere suddiviso schematicamente in alcune entità (Figura 3).

La **descrizione del processo** costituisce l'elemento di riferimento dell'intero processo ed è quindi in relazione con ogni altro elemento. Fornisce diversi tipi di dati:

- dati puramente descrittivi: il nome del processo, una descrizione testuale, commenti e così via.
- dati attinenti all'amministrazione del processo: data di creazione, autore, e così via.
- dati da usare a tempo di esecuzione: parametri di inizializzazione, priorità, scadenze temporali, persone da avvisare, attributi per simulazioni, e così via.

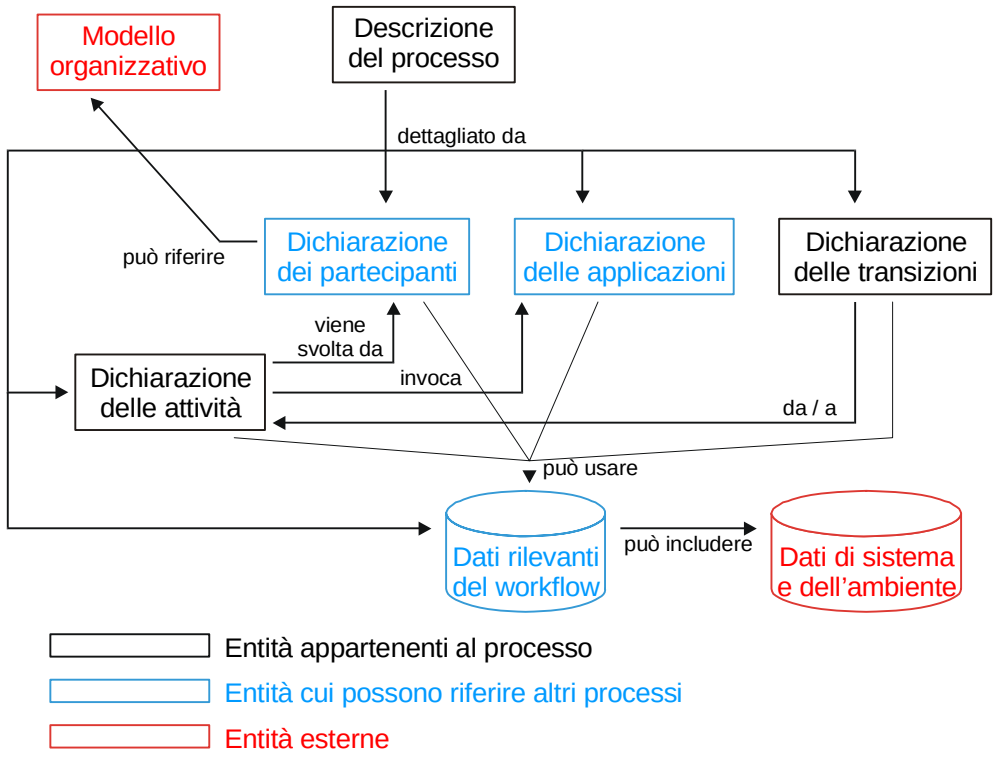


Figura 3: modello dello strumento di definizione dei processi

Un processo consiste di una o più **attività**. Ogni attività rappresenta un lavoro da svolgere all'interno del processo. Il lavoro verrà svolto dai partecipanti e/o da applicazioni informatiche, e potrà utilizzare i dati rilevanti del workflow. Un'attività di un processo può essere di quattro tipi fondamentali :

- atomica (o standard): rappresenta un'attività semplice, attività che contiene un'unità di lavoro che non può essere ulteriormente suddivisa in passi più semplici.
- sottoprocesso: rappresenta un'attività che dà vita ad un nuovo processo per lo svolgimento del lavoro. La chiamata a questo sottoprocesso è caratterizzata da eventuale passaggio di parametri e valori di ritorno.
- ciclo: rappresenta un'attività che viene ripetuta ciclicamente finché non vengono soddisfatte certe condizioni
- vuota: rappresenta un'attività che non svolge lavoro. Il suo utilizzo è funzionale solo alla gestione del flusso di lavoro del processo.

Le attività sono poste in relazione fra loro dalle **transizioni**. Ogni transizione mette in relazione due attività: una di provenienza e una di destinazione. Inoltre le transizioni possono

prevedere condizioni con cui abilitare o impedire il collegamento fra attività. Il flusso di lavoro del processo viene descritto attraverso le transizioni: percorsi paralleli e percorsi alternativi di attività si descrivono con più transizioni che partono o arrivano ad una attività.

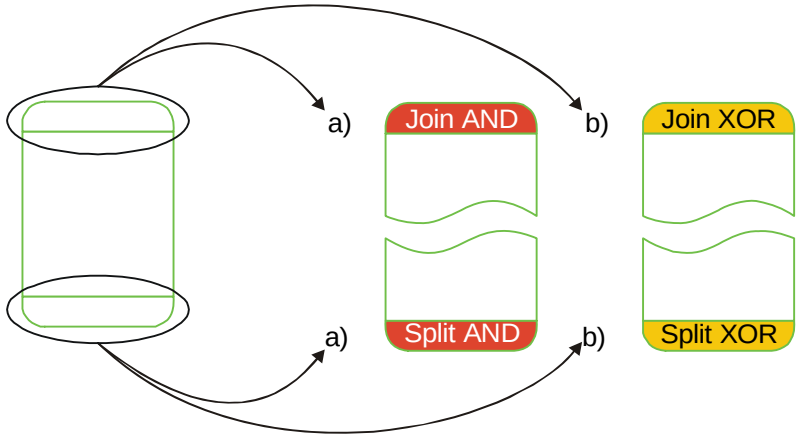


Figura 4: i tipi di split e join delle attività.

Dunque configurando attività e transizioni si possono ottenere flussi di lavoro qualsiasi. Il separarsi del flusso di lavoro (*split*) viene gestito come una forma di post-operazione del lavoro dell'attività. Viceversa il ricongiungersi del flusso di lavoro (*join*) viene gestito come una forma di pre-operazione del lavoro dell'attività (Figura 4).

Le due modalità di percorsi multipli rappresentabili sono:

- **percorsi paralleli**: descritti da più transizioni che partono da un'attività (*split and*). Per coerenza del flusso si hanno i ricongiungimenti dei percorsi paralleli in un'attività di rendez-vous in cui arrivano più transizioni (*join and*). Con riferimento alla Figura 5 la sequenza in cui si svolgono le attività è: A, B e C contemporaneamente, infine D.
- **percorsi alternativi**: descritti da più transizioni che partono da un'attività (*split xor*). Per coerenza del flusso si avranno i ricongiungimenti dei percorsi alternativi in un'attività in cui arrivano più transizioni (*join xor*). Saranno le attività a stabilire se la scelta del percorso da intraprendere sia da imputare al motore di workflow o all'utente. Nel primo caso, scelta automatica, si potrà ricorrere alle condizioni assegnate alle transizioni in uscita dell'attività di *split xor*. Nel secondo caso, scelta manuale, volta per volta si dovrà dichiarare al motore di workflow quale sia il percorso da seguire. Con riferimento alla Figura 5 la sequenza in cui si svolgono le attività è: A, B oppure C (non entrambe), infine D.

Sono le attività a mantenere le informazioni su quale sia il tipo di *split* o di *join* delle transizioni multiple in ingresso ed in uscita.

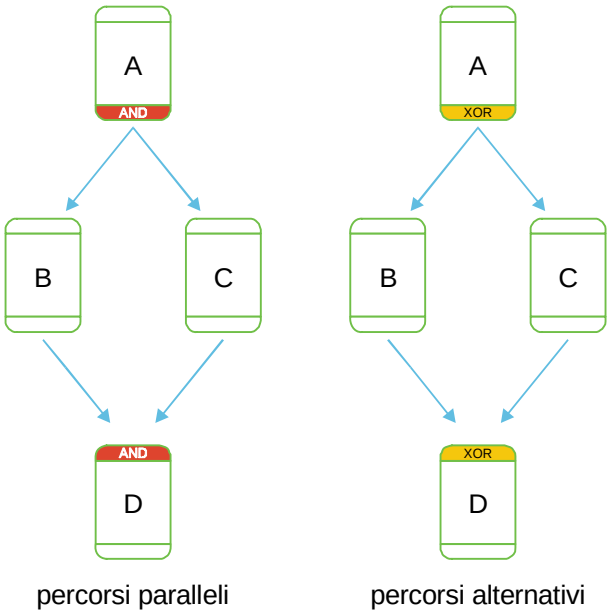


Figura 5: percorsi paralleli e alternativi.

La **dichiarazione dei partecipanti** descrive le risorse che operano sulle attività del processo. Per partecipante non si intende solamente un individuo, una risorsa umana, ma anche gruppi di lavoro (tipicamente suddivisi per competenze) o sistemi informatici.

Il **modello organizzativo** descrive le relazioni e/o l'organizzazione gerarchica interna in cui sono suddivisi i partecipanti. La dichiarazione dei partecipanti vi può fare riferimento in quei casi in cui il processo preveda di interagire con specifiche risorse. Ad esempio si potrebbe riferire l'unità amministrativa di un'azienda o il suo ufficio personale o la sezione di ricerca e sviluppo. La dichiarazione dei partecipanti e il modello organizzativo rappresentano due aspetti diversi delle risorse coinvolte nel processo: la prima fornisce una descrizione per l'uso interno al motore di workflow, la seconda una descrizione più esterna.

La **dichiarazione della applicazioni** descrive i software che le attività utilizzano per supportare o automatizzare del tutto lo svolgimento del lavoro assegnato. Le applicazioni possono essere generici strumenti di sviluppo, software specifici per una certa azienda o procedure implementate all'interno del WfMS. La dichiarazione delle applicazioni specifica anche l'eventuale passaggio di parametri che deve avvenire.

I **dati rilevanti del workflow** definiscono i dati che vengono creati e manipolati dal processo durante l'esecuzione. I dati sono messi a disposizione delle attività e delle loro applicazioni e possono essere utilizzati per valutare condizioni o anche per passare dati fra attività e attività.

I **dati di sistema e dell'ambiente** sono un'estensione dei dati rilevanti del workflow e includono tutti quei dati specifici del sistema operativo o del WfMS stesso. L'accesso a questi dati è possibile solo attraverso chiamate a funzioni definite nei dati rilevanti del workflow.

3.1.2. Applicazioni clienti e applicazioni esterne

Le applicazioni clienti e le applicazioni esterne rappresentano le applicazioni che interagiscono con il workflow nel suo normale funzionamento (Figura 6).

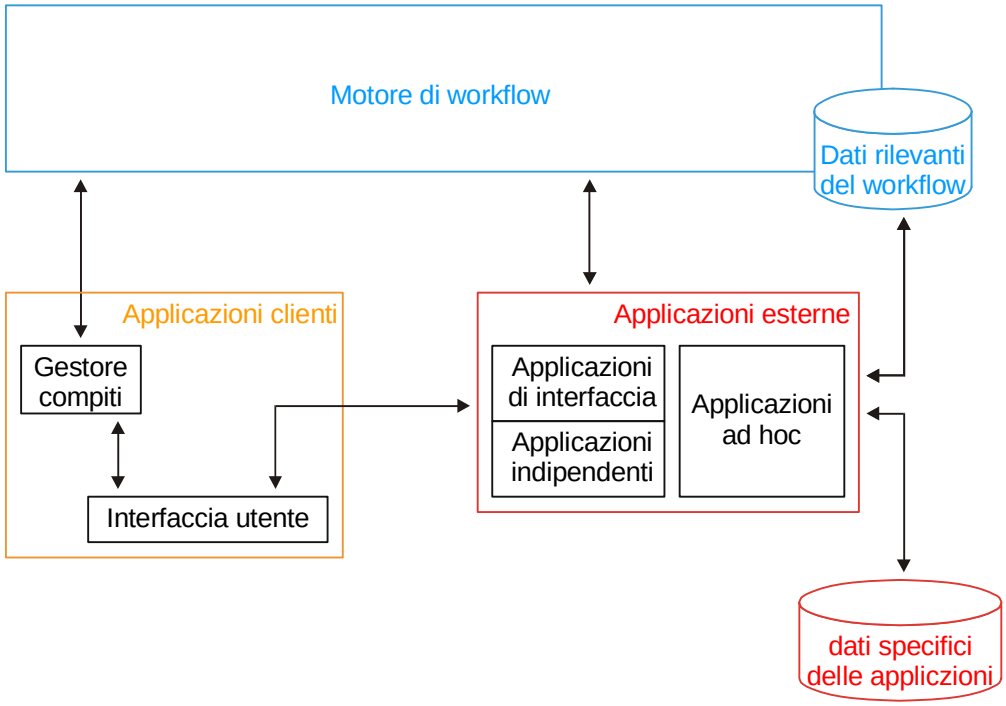


Figura 6: applicazioni clienti e applicazioni esterne.

Le **applicazioni clienti** si frappongono fra l'utente e il motore di workflow, fungendo da interfaccia. Si strutturano in due parti:

- la **gestione dei compiti** si occupa di interrogare il motore di workflow per conto dell'utente. I lavori che l'utente deve svolgere possono essergli assegnati dal motore di workflow.

- l'**interfaccia utente** si occupa di interagire con l'utente: gli presenta il lavoro da svolgere e gli permette di interagire con le applicazioni esterne.

Le **applicazioni esterne** consentono di svolgere in parte o del tutto il lavoro previsto nelle attività del processo. Possono essere invocate dall'utente, tramite l'interfaccia utente, per consentirgli di svolgere i compiti assegnati; oppure possono essere invocate direttamente dal motore di workflow per quelle attività che non necessitano dell'intervento dell'utente. Possono essere di tre tipi:

- **applicazioni ad hoc**, scritte appositamente per lo specifico workflow: tool specifici di un'azienda, programmi personalizzati per specifici reparti e così via.
- **applicazioni indipendenti**, selezionate dal mondo delle applicazioni già esistenti: word processor commerciali, server web esistenti e così via.
- **applicazioni di interfaccia**, dedicate a far interagire il workflow o l'utente con le applicazioni indipendenti, qualora fosse necessario.

Bisogna distinguere i dati utilizzati e le applicazioni esterne. Da una parte si hanno i **dati rilevanti del workflow**, modificati dalle applicazioni nello svolgimento dei lavori previsti dal processo. Sono i dati cui si riferiscono tutte le entità del workflow.

Dall'altra si hanno i **dati specifici dell'applicazione** che nulla hanno a che fare con il workflow. Si tratta dei dati interni all'applicazione, necessari per il suo funzionamento ma al lavoro svolto dal processo.

3.1.3. Strumenti di amministrazione e monitor

Gli strumenti di amministrazione e monitor interagiscono con il motore di workflow osservando il flusso del lavoro nei processi. Il monitoraggio del workflow ha due aspetti.

Da una parte si ha il monitoraggio dello **stato del processo**: si devono poter osservare tutte le istanze del processo, il loro stato, le attività che hanno svolto, quelle che stanno svolgendo e quelle che svolgeranno.

Dall'altra si hanno i **rapporti descrittivi** sul funzionamento del workflow che descrivono in termini di statistici il lavoro svolto dal processo. A seconda del tipo di workflow implementato si avranno metriche diverse; ad esempio un workflow che modelli il flusso di lavoro di un'agenzia bancaria dovrà poter osservare il numero medio di clienti

giornaliero, il totale mensile di operazioni effettuate, il volume di scambi, i conti annualmente aperti e chiusi, e così via.

Si può prevedere inoltre l'aggiunta di feedback inviati al cliente per e-mail con la richiesta di esplicitare un giudizio sul servizio offerto.

3.1.4. Motore di workflow

Il motore di workflow è composto di più entità, come indicato in Figura 7.

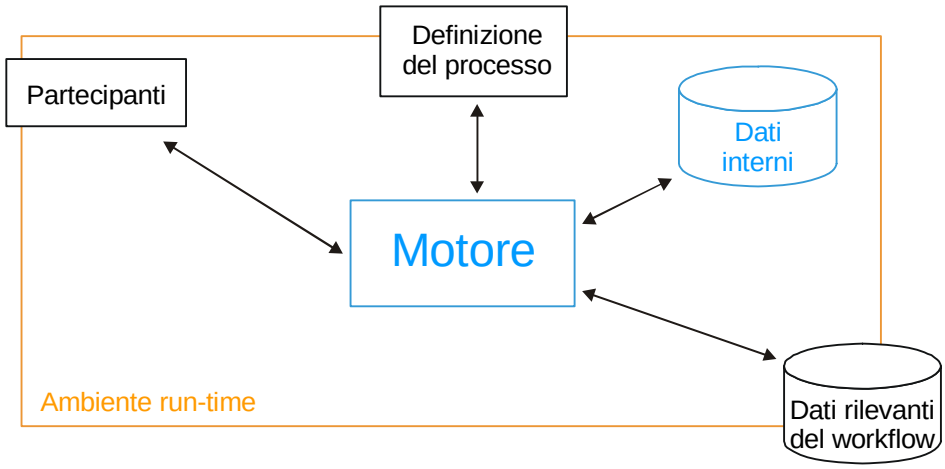


Figura 7: le entità che compongono il motore di workflow.

L'**ambiente run-time** descrive l'evoluzione dello stato delle istanze del workflow. Include tutti gli stati in cui si trova un'istanza e le funzionalità invocabili di stato in stato.

I **dati rilevanti del workflow** sono quei dati che vengono riferiti da tutti i componenti del workflow. Ad esempio le attività usano e modificano i dati attraverso le applicazioni, le condizioni delle transizioni utilizzano i dati per dirigere il flusso di lavoro, e così via. I dati rilevanti del workflow esulano dall'implementazione del WfMS, e dipendono dal processo descritto. I **dati interni** sono invece necessari al funzionamento del WfMS e sono dipendenti dall'implementazione.

I **partecipanti** sono le risorse che gestiscono le attività del workflow. Come si è già visto nella descrizione dei processi, i partecipanti possono essere individui, gruppi di utenti o sistemi informatici. Inoltre possono riferirsi ad una struttura organizzativa esterna al WfMS.

3.1.5. Interfaccia con altri WfMS

La parte di interfaccia con altri WfMS si propone di curare le comunicazioni fra WfMS. Questa entità è necessaria dove esistono più WfMS che lavorano in collaborazione.

Ad esempio un'azienda potrebbe decidere di utilizzare i WfMS per gestire i propri processi produttivi e amministrativi. Sarebbe improduttivo gestire con un solo WfMS entrambi i processi, date le loro diverse finalità. D'altro canto sarebbe anche improduttivo isolare i due processi gestendoli con due WfMS indipendenti, dato che in qualche modo appartengono alla stessa azienda e dunque riferiscono delle entità in comune. La soluzione è utilizzare due WfMS che possano comunicare l'uno con l'altro.

La comunicazione avviene ricorrendo ad API standard: le stesse con cui viene gestito il WfMS. Infatti tutte le interfacce fra le entità che compongono i WfMS (definizione dei processi, applicazioni clienti, e così via) hanno il medesimo formato e possono essere utilizzate fra WfMS e WfMS.

3.2. Strumenti software

Il progetto realizzato è un applicativo per gestire workflow compatibile agli standard delle specifiche dello standard xpdl. È composto da:

- uno strumento di definizione di processi;
- un motore di workflow;
- un monitor e un sistema di amministrazione.

3.2.1. Strumento di definizione di processi

Lo strumento di definizione di processi è in grado di definire le attività che compongono un processo, mettendole in relazione fra loro tramite transizioni che ne stabiliscano il flusso del lavoro. Le sue caratteristiche sono:

- possibilità di associare applicazioni alle attività;
- possibilità di disegnare attività automatiche;
- possibilità di modellare percorsi paralleli;

- possibilità di modellare percorsi alternativi;
- ottimizzazione dinamica dei processi.

Si può **assegnare un'applicazione ad ogni attività** del processo. L'applicazione permette all'utente di svolgere il lavoro dell'attività. Ad esempio, se un'attività prevede che si rediga un rapporto, l'applicazione dovrà invocare un word processor e metterlo a disposizione dell'utente. Le applicazioni sono il cuore del processo: rappresentano ciò che si fa di ogni attività. Possono essere di natura diversa: database relazionali, word processor, server di posta, applicazioni personalizzate e così via. Si può collegare qualsiasi strumento software come applicazione di un'attività.

Inoltre si può distinguere fra applicazioni che necessitano dell'intervento di un utente, e **applicazioni automatiche** che, come dice il nome, vengono eseguite automaticamente dal motore di workflow. Infatti un punto fondamentale dell'efficienza dei WfMS deriva proprio da questo automatizzare quanto possibile le attività del processo, per richiedere il minimo intervento umano.

Nel disegnare il processo si possono prevedere **esecuzioni parallele** o **alternative** di attività, con le relative ricongiunzioni. Questi punti di decisione nel flusso del processo valutano determinate condizioni dipendenti dalla singola esecuzione dell'attività, o dipendenti dall'insieme dei dati comuni del workflow.

Infine, il disegno del processo può svolto anche in condizioni di esecuzione del processo stesso per rendere possibili affinamenti e correzioni che possono evidenziarsi nell'utilizzo. Questa **ottimizzazione dinamica** dei processi incrementa la flessibilità del WfMS realizzato.

3.2.2. Motore di workflow

Il motore di workflow ha le seguenti funzionalità:

- invocare le applicazioni;
- gestire il flusso di lavoro;
- archiviare o distruggere il flusso di lavoro.

In ogni attività che l'utente seleziona, oppure in ogni attività automatica che riceve un'istanza di processo, il motore **invoca** la relativa **applicazione** (o script in PHP), con

l'eventuale passaggio di parametri. Terminato il proprio lavoro l'utente segnalerà al motore la conclusione dell'attività.

Ad ogni conclusione di attività è compito del motore **far procedere il flusso** del processo. Il flusso del processo prosegue nelle attività successive, eventualmente generando esecuzioni parallele di attività o scegliendo, in base alla definizione del processo, un percorso da seguire fra quelli proposti in esclusione reciproca.

Al termine dell'esecuzione dell'intero processo si può **rimuovere il flusso** del processo archiviandola o distruggendola definitivamente. Lasciare a disposizione del motore flussi di processo che hanno già concluso il loro ciclo di vita è inutile e può generare confusione.

3.2.3. Monitor e lo strumento di amministrazione

Lo strumento di amministrazione permette di osservare i processi sia in condizioni statiche (la struttura delle attività) sia dinamiche (il loro flusso).

Della **struttura di ogni processo** si possono visualizzare:

- i dettagli di ogni attività: l'applicazione associata, la descrizione, il performer e così via;
- la struttura delle attività, cioè i legami che stabiliscono il flusso del processo.

Osservare la **dinamica di un processo** si traduce nell'osservarne ogni flusso cioè:

- la storia passata: le attività che sono state completate;
- posizione attuale: quali attività sta svolgendo;
- lo stato corrente: documenti allegati, attributi vari, commenti e così via.

Per gestire le eccezioni, ossia situazioni impreviste dal normale flusso dell'istanza si deve gestire un'istanza di processo in maniera non prevista. I passi che si prevedono sono:

- sospendere l'esecuzione di un'istanza;
- gestire la condizione eccezionale;
- far ripartire l'istanza.

Un'istanza eccezionale, proprio perché ha bisogno di un intervento particolare che esula dalle capacità e/o competenze del processo, non può rimanere nel processo stesso: deve essere isolata.

3.3. Linee guida di sviluppo

Gli obiettivi esposti nella sezione precedente sono stati perseguiti seguendo certe linee guida:

- orientamento alla rete;
- aderenza a modelli standard di linguaggio.

Orientamento alla rete significa che l'intero utilizzo del software da parte dell'utente viene gestito tramite interfaccia web, tramite http e https. Questo permette di considerare l'applicazione realizzata come un'applicazione server che risponde ad applicazioni clienti tramite semplici richieste formattate http.

I **modelli standard di linguaggio** cui si riferisce lo sviluppo del progetto sono quelli elaborati dalla workflow management coalition (WfMC). Questa nasce nel 1996 come tavola rotonda fra esperti e studiosi di workflow management system. I suoi sforzi sono orientati a dare linee guida e standard per lo sviluppo di prodotti di workflow management system. La formalizzazione apportata dalla WfMC ha lo scopo ultimo di permettere a tutti i sistemi di workflow di essere compatibili gli uni con gli altri, in modo da permettere alle aziende, agli enti, di adottare modelli di workflow differenti per situazioni diverse senza dover erigere muri di incomunicabilità fra i dati trattati.

4. Strumenti di sviluppo utilizzati

Gli strumenti utilizzati nello sviluppo del progetto sono JaWE, XAMPP, PHP-Documentor (utilizzato per comporre la documentazione delle librerie), SciTE (Scintilla Text Editor).

Nella prima sezione del capitolo si introduce Jawe, un editor di workflow che permette di gestire semplicemente flowchart, e che offre la massima flessibilità di implementazione utilizzando il linguaggio Xpdl.

Nella seconda sezione del capitolo si descriverà l'utilizzo di XAMPP, che comprende diversi pacchetti software, tra cui Apache, PHP4 e 5, MySQL, Mercury Mail Transport System. Permette di implementare applicazioni dinamiche in PHP, elaborare dati Xml utilizzando lo standard DOM e la libreria SimpleXML ed altri vantaggi.

Nella terza sezione si introduce il CMS, lo strumento di amministrazione necessario per gestire i flussi di lavoro.

4.1. Jawe

Enhydra Jawe è un ambiente aperto per disegnare workflow scritto in Java gratuitamente reperibile sul web.

Le caratteristiche che rendono Jawe ottimale per sviluppare flussi sono molte: se ne presentano di seguito le principali.

Jawe è completamente **compatibile con le specifiche *Xpdl* (**XML Process Definition Language) della Workflow Management Coalition. Utilizza gli attributi «estesi», previsti dallo standard, per gestire informazioni relative al layout grafico.

Inotre consente la **memorizzazione su file** dei processi di workflow per un successivo uso, definiti in modalità grafica: tramite un motore di workflow esterno,così è possibile interpretarli in ambiente d'esecuzione.

Jawe permette una facile **integrazione** con altri prodotti, engine ed editor, grazie alla compatibilità con le specifiche *Xpdl*.

Jawe è **facile da utilizzare**: grazie ad una form che permette di editare in modo guidato gli elementi a livello sia di pacchetto che di processo. Per un livello di controllo maggiore, è sempre possibile passare alla vista Xpdl che mostra il documento Xml così come sarà salvato e consente di modificare direttamente i tag e gli attributi.

La maggiore **flessibilità** implica però maggiore controllo: quando si apre o si salva un file, effettua un controllo sintattico e semantico, avvertendo l'utente di eventuali incongruenze.

Il software è open source ed è scaricabile gratuitamente dal sito, ma è disponibile anche una versione commerciale, realizzata dalla società di Vienna, Together Teamlösungen.

4.1.1. XPDL: XML Process Definition Language

XPDL è uno standard ideato dalla *WFMC* (*WorkFlow Management Coalition*) che si basa sul linguaggio XML, linguaggio di markup estensibile (meta-linguaggio) che consente di definire tag personalizzati e anche altri linguaggi o strumenti, quale ad esempio XSLT.

L'XPDL definisce un schema standard per la rappresentazione di un processo di business (*workflow*). L'obiettivo di tale standardizzazione è quello di fornire un punto di riferimento unico per la rappresentazione dei processi di lavoro e per garantire l'interoperabilità tra le applicazioni software che gestiscono tali processi.

L'XML, acronimo di eXtensible Markup Language, ovvero «Linguaggio di marcatura estensibile» è un metalinguaggio creato e gestito dal World Wide Web Consortium (W3C). È una semplificazione e adattamento dell'SGML, da cui è nato nel 1988. La formattazione di questo linguaggio è resa possibile dallo strumento XSLT; come un file .css per pagine HTML, questo strumento definisce la formattazione del file XML rendendo anche possibile l'utilizzo di tag XHTML (definendo i namespaces di riferimento).

La funzionalità di questo linguaggio estensibile è lo scambio dei dati, quindi di back-office (al contrario del linguaggio HTML che ha funzionalità di front-office).

Un esempio di file XPDL è il seguente:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type='text/xsl' href='foglio-formattazione.xsl' version='1.0' ?>
<Package xmlns:xpdl="http://www.wfmc.org/2002/XPDL1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" Id="workflow_prova"
    xsi:schemaLocation="http://www.wfmc.org/2002/XPDL1.0 http://wfmc.org/standards/docs/TC-
    1025_schema_10_xpdl.xsd">
  <PackageHeader>
    <XPDLVersion>1.0</XPDLVersion>
    <Description>Descrizione del gruppo di workflows</Description>
  </PackageHeader>
  <RedefinableHeader PublicationStatus="UNDER_TEST">
    <Author>Armato&Cigliano</Author>
  </RedefinableHeader>
  <Partecipants>
    <Partecipant Id="cliente" Name="nome_cliente">
      <PartecipantType Type="ROLE"/>
    </Partecipant>
```

```
    <Partecipant Id="venditore" Name="nome_venditore">
      <PartecipantType Type="ROLE"/>
    </Partecipant>
  </Partecipants>
  <WorkflowProcesses>
    <WorkflowProcess>
      <ProcessHeader>
        <Description>Descrizione del workflow</Description>
      </ProcessHeader>
      <Activites>
        <Activity Id="uno">
          <Description>Questa e' la descrizione della prima attivita'</Description>
          <Performer>cliente</Performer>


          <SimulationInformation>
            <TimeEstimation>10s</TimeEstimation>
          </SimulationInformation>
          <ExtendedAttributes>
            <ExtendedAttribute Name="funzione" Value="uno.php"/>
            <ExtendedAttribute Name="variabile" Value="desy.mail"/>
            <ExtendedAttribute Name="variabile" Value="fabio.residenza"/>
            <ExtendedAttribute Name="variabile" Value="desy.eta"/>
          </ExtendedAttributes>
        </Activity>
        <Activity Id="due">
          <Description/>
          <Performer>venditore</Performer>
          <SimulationInformation>
            <TimeEstimation>10s</TimeEstimation>
          </SimulationInformation>
          <ExtendedAttributes>
            <ExtendedAttribute Name="funzione" Value="due.php"/>
          </ExtendedAttributes>
        </Activity>
      </Activites>
      <Transitions>
        <Transition From="uno" Id="Transition1" To="due">
          <ExtendedAttributes>
            <ExtendedAttribute Name="trans1" Value=" CONDITION "/>
```

```
            </ExtendedAttributes>
          </Transition>
          <Transition From="due" Id="Transition1" To="tre">
            <ExtendedAttributes>
              <ExtendedAttribute Name="trans2" Value=”CONDITION "/>
            </ExtendedAttributes>
          </Transition>
        </Transitions>
        <ExtendedAttributes>
          <ExtendedAttribute Name="variabile_workflow" Value="20"/>
        </ExtendedAttributes>
      </WorkflowProcess>
    </WorkflowProcesses>

    <ExtendedAttributes>
      <ExtendedAttribute Name="variabile_genrale" Value="10"/>
    </ExtendedAttributes>
</Package>
```

I principali *tag* con cui abbiamo a che fare sono i seguenti:

- <*?xml-stylesheet type=.. href=.. ?*>, elemento che definisce il riferimento al foglio di stile.
- <*Package xmlns:xpdl=.. >*, *xmlns* è la sintassi per definire l’URL di riferimento allo spazio dei nomi a cui si atterrà la parola “xpdl” nel file. Questo tag è la root del file, che conterrà i prossimi elementi.
- <*Partecipants*>, che comprende tutti i tags <Partecipant> caratterizzati da un *Id* e *Name;* saranno i performer delle Activity.
- <*WorkflowProcesses*>, che racchiude i tags <WorkflowProcess> che costituiscono ogni singolo workflow.
- **<***Activites***>,** all’interno di tale tag sono definiti tutti gli attributi necessari per lo svolgimento dell’attività.
- **<***Transitions***>** ha la funzione di stabilire il percorso che dovrà seguire il workflow.

- <ExtendedAttributes> definiscono delle variabili riferite all'attività in cui sono state specificate.

4.2. XAMPP: Apache web server, MySQL, PHP, Perl.

XAMPP include diversi pacchetti software in un unica soluzione. Alcuni pacchetti più importanti riguardano Apache, MySQL, FileZilla, phpMyAdmin e tutti gli strumenti necessari per programmare con PHP e Perl,. Il programma è rilasciato sotto la *GNU General Public License*. Mediante XAMPP è possibile avere un web server capace di interpretare pagine dinamiche PHP.

XAMPP è uno strumento utile per creare applicazioni per il web: applicazioni che si trovano su un server e che rendono disponibili le loro interfacce attraverso il protocollo http. Tramite XAMPP si possono, creare e mantenere i servizi di complessi siti dinamici.

4.2.1. Componenti di XAMPP

I principali strumenti che compongono XAMPP sono:

- un server web Apache [vers. 2.2];
- un database MySQL [vers. 5.0];
- moduli per il supporto di PHP [vers. 5.2.1] e Perl [vers. 5.8.8];
- FileZilla server FTP;
- phpMyAdmin;
- Mercury Mail Transport System.

XAMPP mette a disposizione innumerevoli strumenti. Ad esempio Webalizer, programma per l'analisi dei file di log, per tenere sotto controllo il lavoro del proprio web server; oppure PHP Switch per testare gli script con PHP 4 o PHP 5.

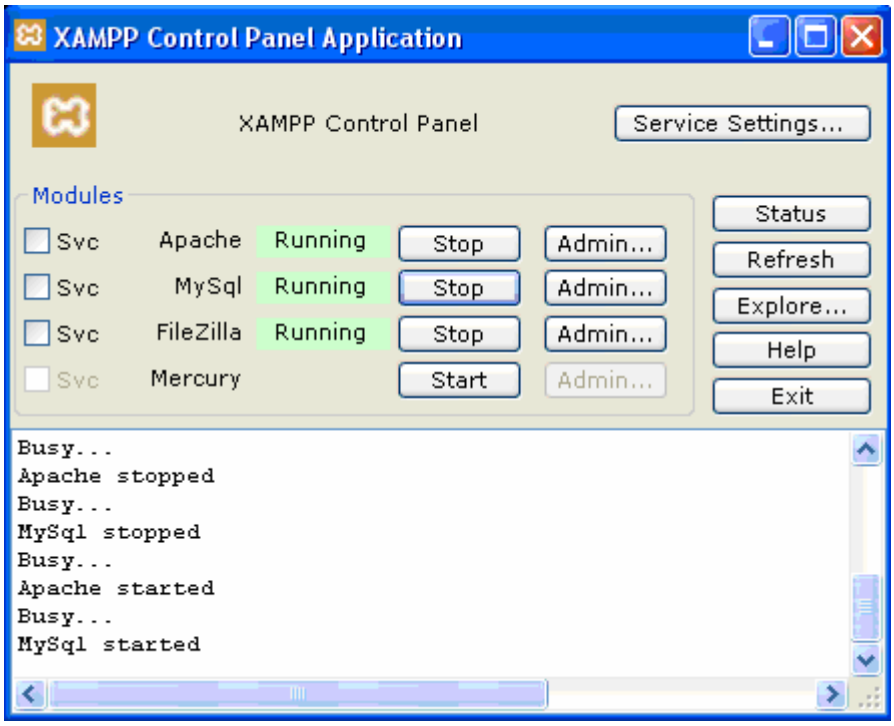


Figura 8: la form di gestione XAMPP.

4.2.2. Come funziona XAMPP

Il funzionamento di XAMPP ruota intorno al suo server web: Apache. Si tratta della piattaforma server Web più diffusa in grado di operare su sistemi operativi UNIX-Linux e Microsoft. Quando il server Apache di XAMPP viene contattato con una richiesta http (da un web browser sulla porta :80), il server individua il file utilizzando l'URL della richiesta nella cartella di root che può essere configurata nel file php.ini (nella cartella *apache/bin*). Una volta che il file è stato trovato il server interpreterà il linguaggio, con i parametri estratti dall'URL della richiesta.

4.2.3. PHP: Hypertext Preprocessor

PHP è un linguaggio di scripting interpretato , con licenza open source, originariamente concepito per la realizzazione di pagine web dinamiche. Attualmente è utilizzato principalmente per sviluppare applicazioni web lato server ma può essere usato anche per scrivere script a linea di comando o applicazioni standalone con interfaccia grafica. Le caratteristiche principali sono le seguenti:

- Dalla versione 5 migliora il supporto al paradigma di **programmazione ad oggetti**.
- Mette a disposizione **moltissime API**, oltre 3000 funzioni del nucleo base; a titolo di esempio ne fornisce una specifica per interagire con Apache.
- Supporta numerose tecnologie, come **XML**, **SOAP**, IMAP, FTP, CORBA.
- Si integra anche con altri linguaggi/piattaforme quali Java e .NET.

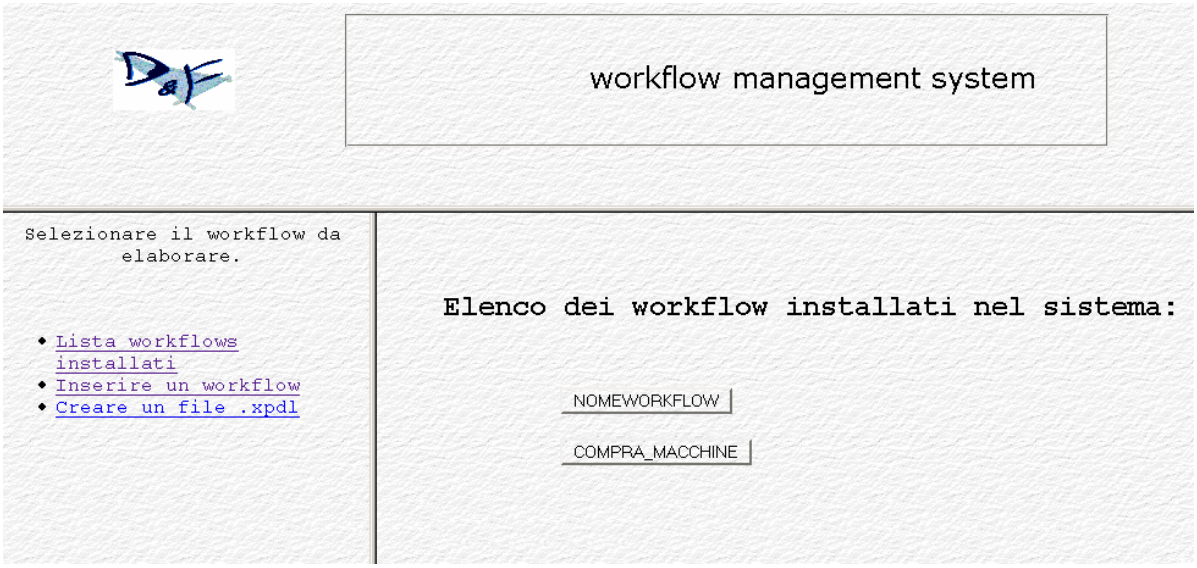
5. Implementazione del progetto

Nella prima sezione del capitolo si introduce il concetto di CMS, analizzandone la struttura e la funzionalità.

Nella seconda sezione si descrive il progetto in generale, sottolineando i concetti principali.

Nella terza sezione si descrivono le specifiche del progetto secondo una visione ad use case: si introducono gli attori e se ne descrivono i relativi compiti.

5.1. CMS: Content Management System



Un sistema di gestione del contenuto è una parte di software per il lato server che consente di creare, pubblicare e gestire del contenuto in modo rapido ed efficace. Di solito è composto dai seguenti componenti:

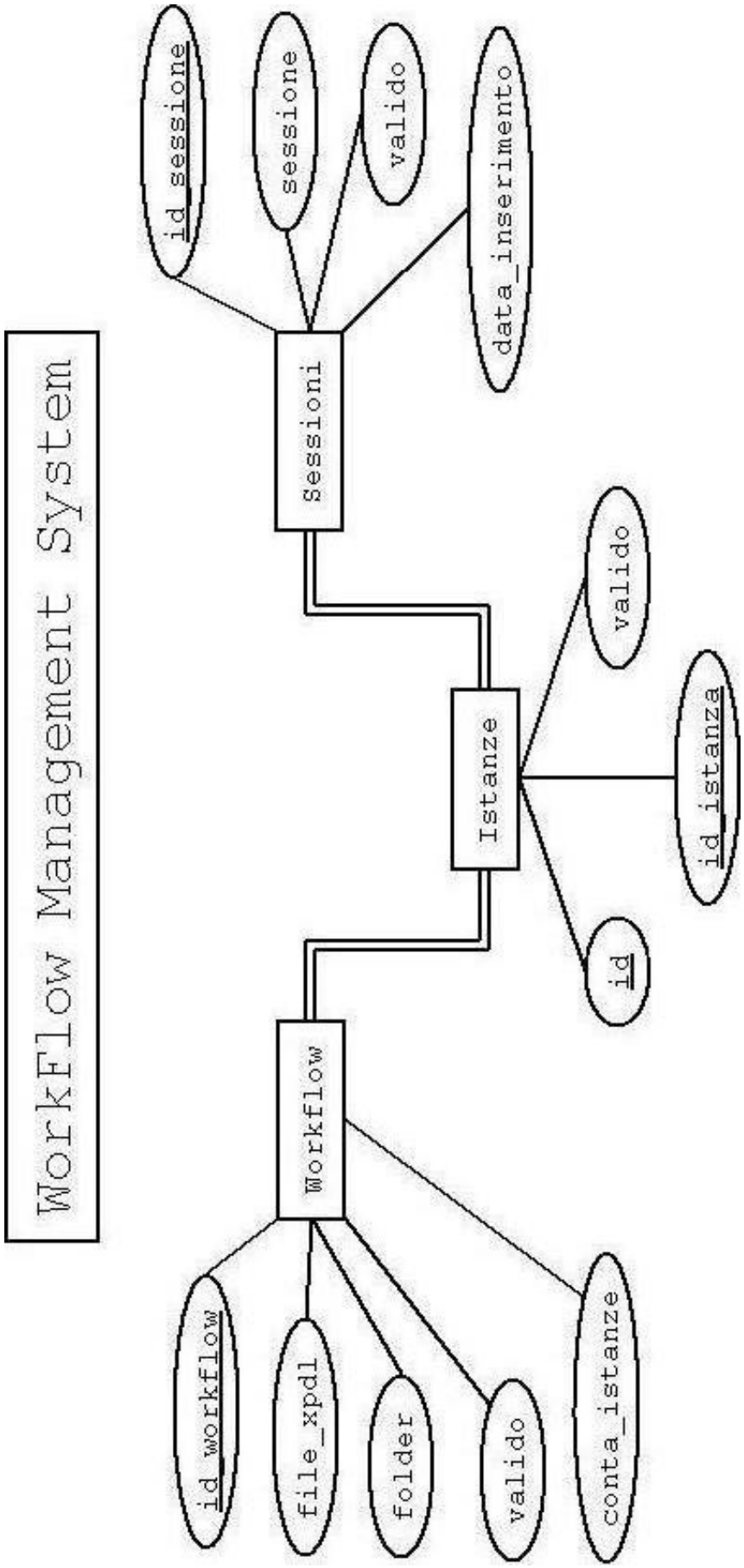
- Un componente di **back-end dei dati** (che comprende tabelle XML o di database) contenente tutti i contenuti necessari al flusso di lavoro.
- Un componente di **visualizzazione dei dati**, di solito modelli o altre pagine, in cui il contenuto viene "disegnato" dal CMS per essere visualizzato all'utente.
- Un componente di **amministrazione dei dati**. Di solito comprende form HTML di facile utilizzo che consentono agli amministratori di sito di creare, modificare ed eliminare elementi.

5.1.1. Back-end dei dati

La registrazione dei dati è una parte fondamentale del progetto. Per l'implementazione abbiamo utilizzato lo strumento MySQL messo a disposizione dal pacchetto XAMPP.

Sono state necessarie tre tabelle per gestire i flussi di lavoro (workflow), le singole istanze e le sessioni dedicate agli utenti. Ad ogni workflow sono associate più istanze e ad ogni istanza sono associate più sessioni.

Nella pagina seguente vi è il modello E/R (entità e relazioni) che mostra la struttura delle tabelle utilizzate dal sistema.

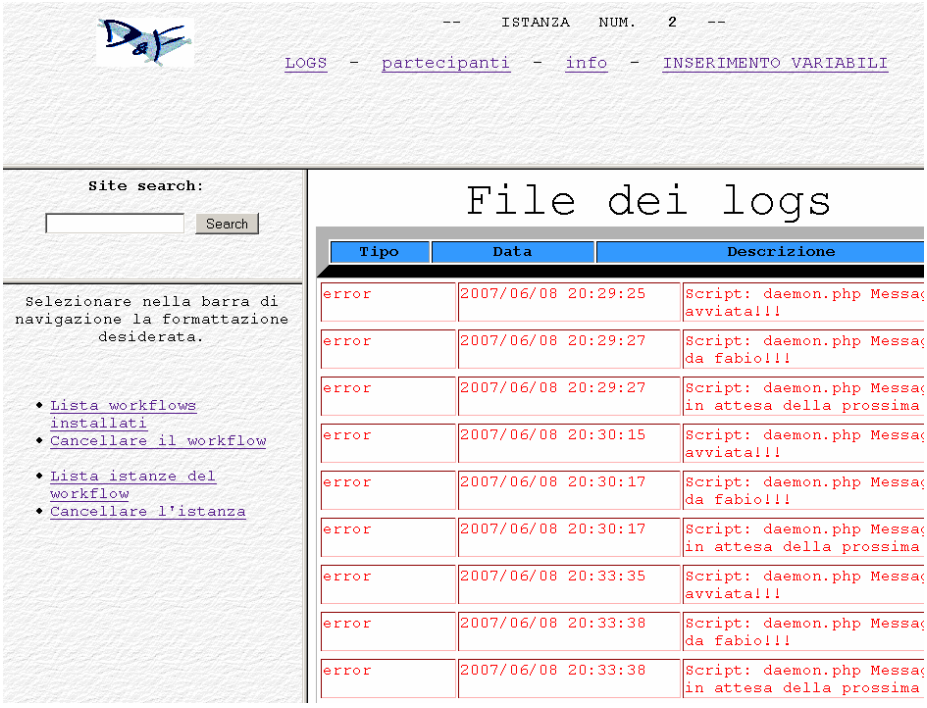

WorkFlow Management System
Sessioni
id_sessione
sessione
valido
data_inserimento
Istanze
valido
id_istanza
id
Workflow
id_workflow
file_xpdl
folder
valido
conta_istanze

5.1.2. Visualizzazione dei dati

I files .xpdl che definiscono le attività del workflow e i files .xml che definiscono gli attributi specifici di ogni istanza possono essere visualizzati tramite fogli di stile sviluppati in XSLT. Vengono sviluppate tabelle apposite per evidenziare gli attributi più significativi, come possono essere gli script e le variabili associati.

5.1.3. Amministrazione dei dati

E' superfluo precisare che l'interfaccia amministrativa deve essere sicura per evitare che chiunque possa accedere al CMS ed eliminare il contenuto, apportare modifiche non autorizzate al contenuto esistente.



La sicurezza del flusso di dati deve tener conto perciò dell'autenticazione degli utenti che dovranno interagire con il sistema.

L'**autenticazione** consiste nel riconoscere un utente attraverso una login e una password. Un altro sistema di autenticazione più sicuro è permettere l'accesso solo attraverso sessioni inviate per e-mail, che permetteranno di eseguire singole fasi del flusso di lavoro.

Grazie a questa politica di gestione degli utenti permettiamo esclusivamente agli amministratori di sistema di accedere al CMS, per poter inserire, modificare ed eliminare elementi del processo.

Gli utenti secondo questa logica devono essere distinti in *amministratore del sistema*, *cliente* e *venditore*: i venditori saranno inseriti in tabelle back-end, i clienti nei files .xml che caratterizzano ogni istanza avviata.

È infatti importante distinguere chi deve gestire l'applicazione e chi ne deve fruire.

5.2. Descrizione generale

Il progetto implementa un WfMS gestito attraverso un motore di workflow (*daemon.php*), un *CMS* (*Content Management System*), un monitor (*monitor.php*). Il sistema è basato fondamentalmente su PHP, il linguaggio XML e MySQL.

Nel progetto dobbiamo dunque da affrontare diverse fasi di esecuzione, di seguito si elencano le principali:

- *Definizione del flusso di lavoro*: questa fase è a carico dell'amministratore. Si occupererà di disegnare il processo di workflow tramite l'editor apposito (*Jawe*). Sarà così disponibile il file (*workflow.xpdl*) su cui si baserà l'intero processo.
- *Installazione del workflow*: lo strumento di amministrazione permette di installare un processo. Per questa operazione sono necessari i seguenti passi:
  - Inserimento nome del Workflow (viene controllata la presenza di un workflow con lo stesso nome).
  - Verrà creata la cartella radice che conterrà tutti i files inerenti al workflow.
  - Caricamento del file xpdl che definisce il Workflow (questo file verrà poi messo nella root del Workflow, precedentemente creata).
  - Il Workflow verrà registrato nel database del sistema.
  - Caricamento degli scripts che compongono il Workflow.
  - Caricamento delle query SQL che creano le tabelle utilizzate dal Workflow (verrà creato un database con lo stesso nome del Workflow in questione).
  - Caricamento ed esecuzione dello script in PHP per 'popolare' le tabelle appena create.
- *Interpretazione workflow.xpdl*: verranno estratte informazioni utili all'esecuzione del processo dall'intersezione del file xpdl con il file dell'istanza che si sta analizzando (in

formato xml). Analizzerà anche lo *status* dell'attività per tenere sotto controllo il processo: potrà essere avviata, ferma, eseguita, ect ..

- *Richiesta del servizio*: l'utente accederà al portale per richiedere il servizio. Dovrà inserire le informazioni necessarie per lo svolgimento di tutto il processo, come l' e-mail.
- *Ogni performer completa la sua attività*: nell'e-mail inviatagli sarà disponibile il link al server web per accedere all' attività da svolgere. Si dovrà preoccupare di svolgere il suo compito nel tempo stabilito nel file .xpdl *(<TimeEstimation>*); altrimenti il sistema lo inviterà più volte a presentare il suo lavoro, contattando anche l'amministratore per avvisare dell'inconveniente.
- *Finito il ciclo di workflow:* il sistema avviserà il responsabile della conclusione con esito positivo, attraverso il sistema e-mail; il monitor si occuperà di chiudere il processo, che verrà archiviato nel database del sistema.
- *Indagine statistica sulle preferenze dei clienti:* il sistema prima di archiviare i dati inerenti al processo si occuperà dell'invio di feed-back al cliente, con lo scopo di migliorare il servizio offerto.

5.3. Use case

5.3.1. Attori

Nell'analisi del progetto si deve tener conto di tre tipi di attori:

- *Amministratore o manager*: ricopre il ruolo di amministratore. In generale egli è incaricato di gestire il sistema in tutti i suoi aspetti decisionali di alto livello (stesura delle mappe dei processi, assegnazione delle Role agli User, ecc).
- *Agente* o *employee*: partecipante del sistema a cui viene assegnata una o più *activities* del workflow ed è incaricato di svolgerle. Ad esempio l'ufficio vendite di un' azienda.
- *Cliente o customer:* la persona che compilando un modulo di richiesta, farà avviare un'istanza di workflow.

5.3.2. Use case per il manager

Gli use case di un *manager* sono riportati in Figura 9.

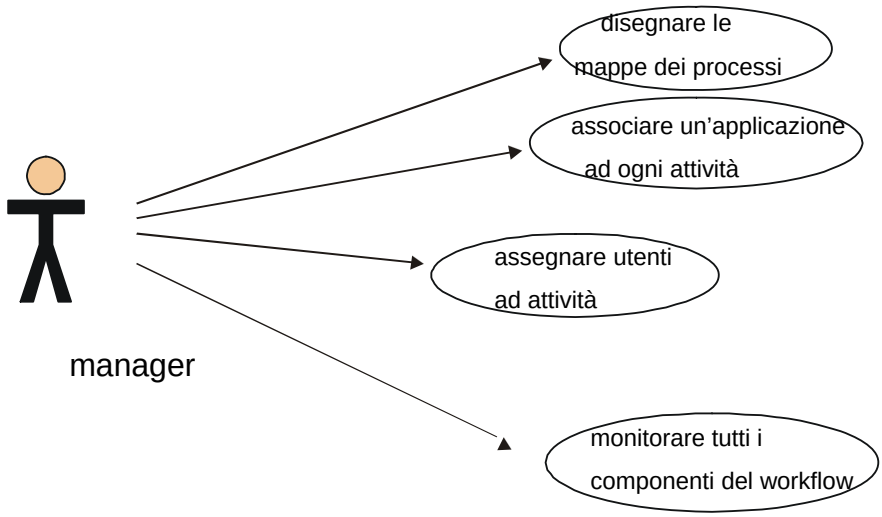


Figura 9: use case per il manager

Per **disegnare le mappe dei processi** del workflow il *manager* deve creare un *process* che raccolga le *activity* e *transition* che definiscono la mappa. Per ogni attività del processo da disegnare il *manager* crea una *activity*. Per collegare le attività fra loro il *manager* crea le *transition* configurandole in modo da rispettare i collegamenti fra le attività del processo. Una delle caratteristiche determinanti per la flessibilità del sistema è la possibilità di modificare i processi a piacere in qualsiasi momento, anche mentre ci sono delle istanze di processo "vive", cioè anche in situazioni dinamiche. Nel caso si modifichino delle attività (cancellandole, ad esempio) che avevano una coda di istanze di processo da smaltire, tutte le istanze vengono automaticamente a cadere nello stato di "*fallout*" per poter essere gestite come eccezioni.

Per **associare un'applicazione ad una attività** il *manager* deve configurare il campo *application* dell'*activity* relativa. In questo campo si deve riportare un URL che deve essere invocato ogni qualvolta che un'istanza di processo deve eseguire l'applicazione dell'attività.

Eventuali parametri da passare all'applicazione possono essere messi nel campo *parameters*. I parametri verranno aggiunti automaticamente all'URL invocato.

Per **assegnare agli utenti le attività** il *manager* deve assegnare ad ogni *activity* i *performer* che la porteranno a termine nel file .xpdl.

Per **monitorare il workflow** in tutte le sue parti il *manager* può accedere alle pagine del CMS, per mezzo di una login e password. Saranno disponibili l'elenco dei *workflows*, delle istanze; inoltre sono disponibili tabelle che riassumono le caratteristiche principali del file .xpdl di ogni workflow e di ogni istanza.

5.3.3 Use case per l'employee

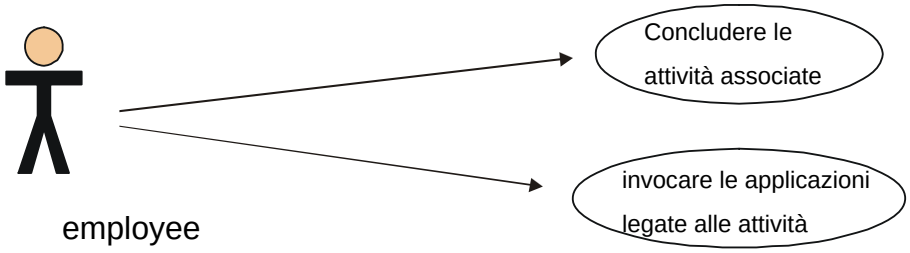


Figura 10: use case per l'employee.

Gli use case per l'employee sono riportati in Figura 10.

Per invocare l'applicazione di una attività l'employee (l'utente generico che dovrà eseguire delle attività) avrà a disposizione una pagina di gestione dell'attività: il motore invocherà automaticamente l'applicazione associata alla *activity*.

5.3.4. Use case per il customer

Gli use case per il customer sono riportati in Figura 11.

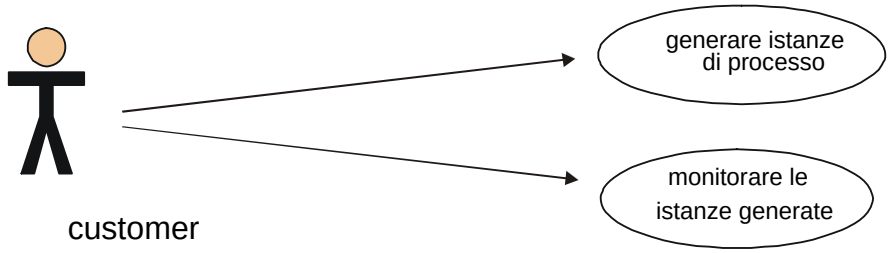


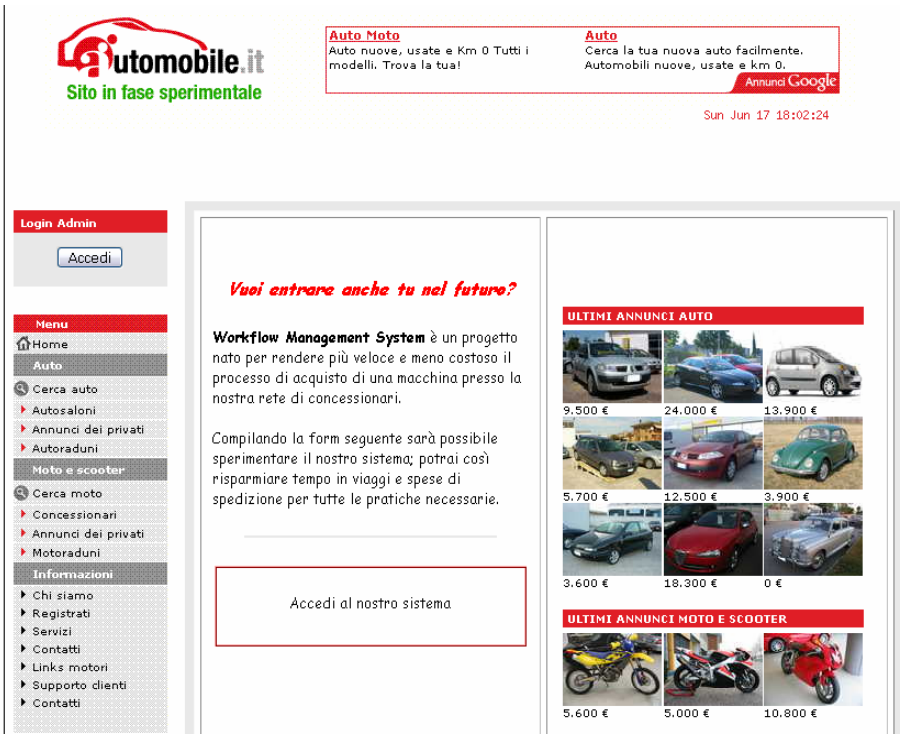
Figura 11: use case per il customer.

Per **generare un'istanza di processo** il customer dovrà accedere alla pagina di gestione delle richieste di servizi degli utenti. Una volta registratosi e inviata la richiesta del servizio, il customer non dovrà preoccuparsi di nient'altro. Sarà l'applicativo che avvierà un'istanza di processo. Se, ad esempio, il processo gestisce le richieste di acquisto di un centro di ricerche, un'interfaccia web permetterà agli utenti (i *customers*) di compilare il modulo di richiesta, inserendo obbligatoriamente l' e-mail, e l'applicativo avvierà l'istanza del workflow specifico.

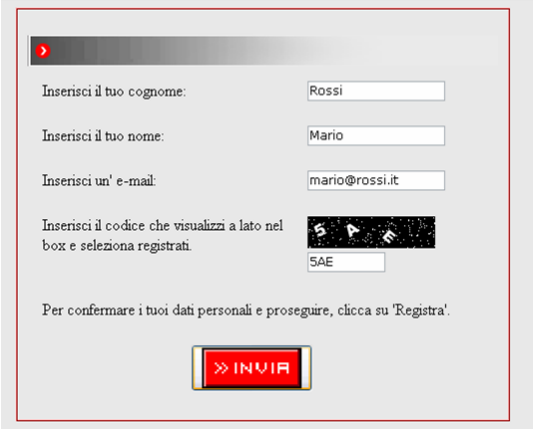
6. Esempio di applicazione: il COMPRA MACCHINE

6.1. Il portale dell'autoconcessionaria

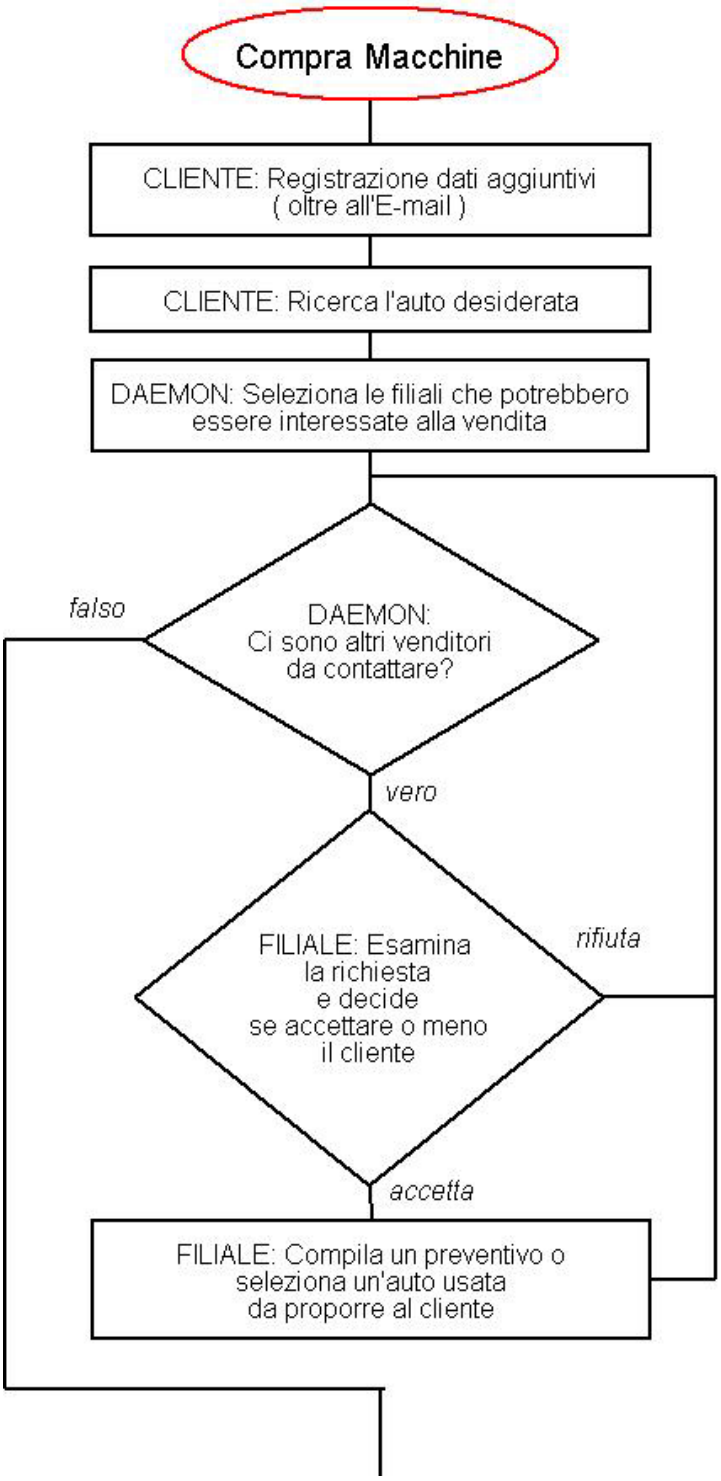
" **Vuoi entrare anche tu nel futuro?** **Workflow Management System** è un progetto nato per rendere più veloce e meno costoso il processo di acquisto di una macchina presso la nostra rete di concessionari. Compilando la form seguente sarà possibile sperimentare il nostro sistema; potrai così risparmiare tempo in viaggi e spese di spedizione per tutte le pratiche necessarie. "

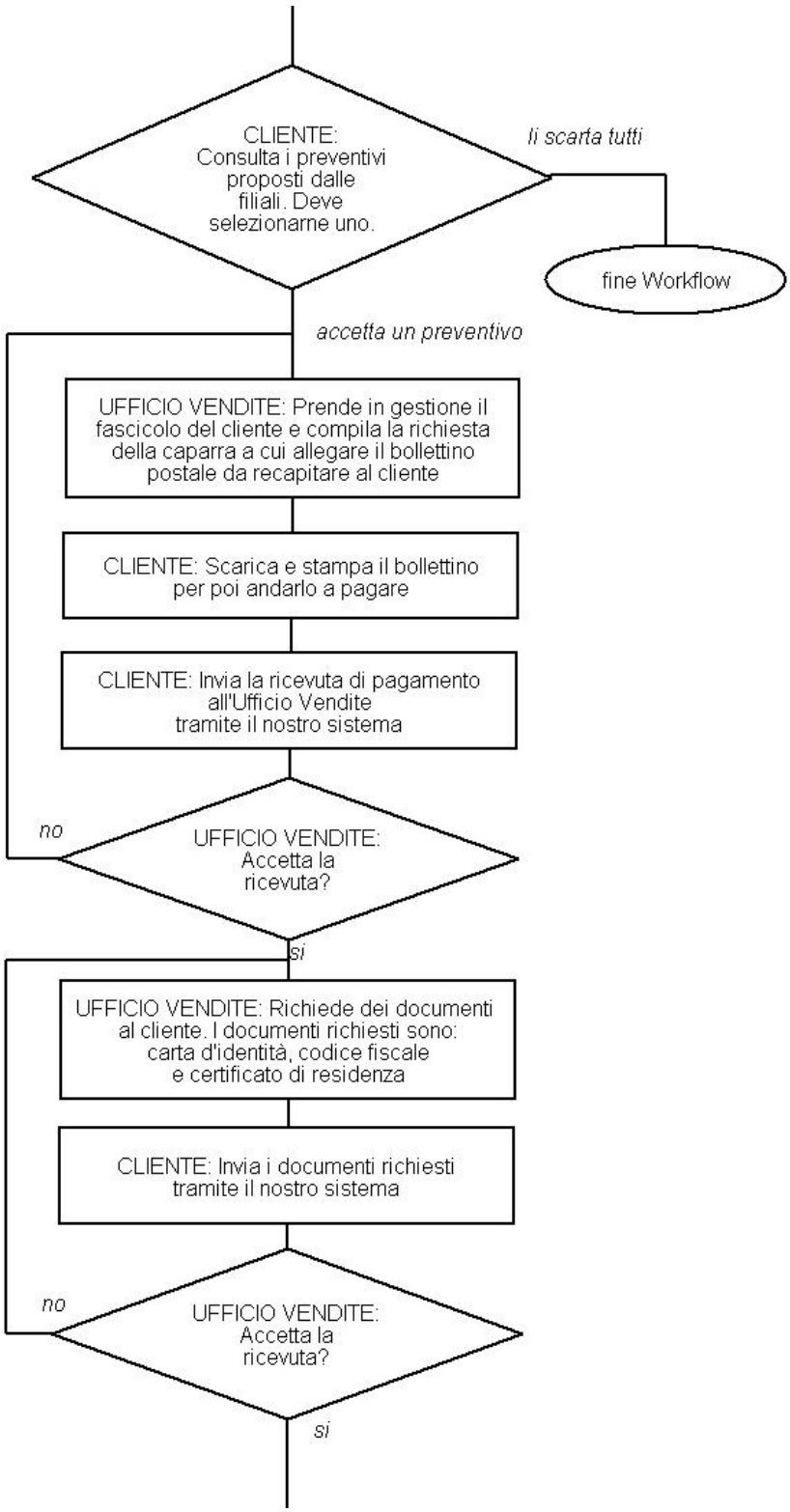


Questa è la presentazione del nostro progetto sul portale dell'Autoconcessionaria in una ipotetica applicazione. Il cliente registrandosi avvia un'istanza del workflow COMPRA MACCHINE installato nel nostro sistema.



6.2. Il flow chart




CLIENTE:
Consulta i preventivi
proposti dalle
filiali. Deve
selezionarne uno.
li scarta tutti
fine Workflow
accetta un preventivo
UFFICIO VENDITE: Prende in gestione il
fascicolo del cliente e compila la richiesta
della caparra a cui allegare il bollettino
postale da recapitare al cliente
CLIENTE: Scarica e stampa il bollettino
per poi andarlo a pagare
CLIENTE: Invia la ricevuta di pagamento
all'Ufficio Vendite
tramite il nostro sistema
no
UFFICIO VENDITE:
Accetta la
ricevuta?
si
UFFICIO VENDITE: Richiede dei documenti
al cliente. I documenti richiesti sono:
carta d'identità, codice fiscale
e certificato di residenza
CLIENTE: Invia i documenti richiesti
tramite il nostro sistema
no
UFFICIO VENDITE:
Accetta la
ricevuta?
si


si
UFFICIO VENDITE: Contatta il P.R.A. per richiedere l'immatricolazione dell'auto
P.R.A: Risponde con i dati di immatricolazione
UFFICIO VENDITE: Registra i dati e propone al cliente un termine per il ritiro dell'autovettura
CLIENTE: Conferma la data del ritiro dell'autovettura
SISTEMA: Archivia il fascicolo
fine Workflow

6.3. l'Indagine statistica

Abbiamo predisposto così l'indagine statistica:

1) **Pianificazione**

   Obiettivo:

   - Rendere ottimale il servizio rivolto ai clienti.
   - Monitoraggio del grado di soddisfazione dei clienti al fine di migliorare il servizio.

   Oggetto:

   - Tutti i clienti che hanno utilizzato il servizio.

   Oggetto della rilevazione:

   - Servizio di vendita di autoveicoli attraverso un processo automatizzato (detto Workflow).

   Modo di rilevazione:

   - Questionario rivolto ai clienti che hanno testato il processo.

2) **Raccolta dati**

Poche, chiare e semplici domande per rendere più rapida la compilazione da parte dell'utente.

Richiesta di dati generici: età, sesso, professione, titolo di studio.

L'interfaccia che ha avuto a disposizione è stata semplice e di facile comprensione?	Per niente | Poco | Abbastanza | Molto
Ha ritenuto accettabile il tempo impiegato per le varie fasi?	Per niente | Poco | Abbastanza | Molto

I preventivi proposti sono inerenti alle Vostre aspettative?	Per niente | Poco | Abbastanza | Molto
Ritiene questo servizio efficiente e utile per un effettivo risparmio di tempo?	Per niente | Poco | Abbastanza | Molto

3) Elaborazione dati

Dopo la compilazione del questionario via Web, il sistema aggiornerà le rilevazioni immagazzinate nel database.

Tramite l'interfaccia dell'amministratore, sarà possibile consultare una griglia riassuntiva delle rilevazioni e i grafici corrispondenti.

7. Conclusioni personali

In questi mesi di lavoro abbiamo affrontato diverse problematiche. Innanzitutto non è sempre facile il lavoro di gruppo, perché è una attività molto complessa, anche se il gruppo è composto da sole due persone.

Alcune questioni dovrebbero essere considerate attentamente prima di iniziare a lavorare insieme: prima di tutto, è bene che ci si chieda esattamente quale è l'obiettivo da raggiungere. Ed è inoltre importante che tutti i membri del gruppo abbiano capito l'obiettivo.

Questo permette poi al gruppo di lavoro di individuare e condividere anche come raggiungere il nostro obiettivo, ovvero il metodo di lavoro, come organizziamo i tempi, come ci suddividiamo i ruoli e le responsabilità, quali sono le competenze di ogni membro del gruppo, quali sono le differenze.

Lavorare in gruppo non significa focalizzare l'attenzione solo sull'obiettivo da raggiungere ma anche sul "come" si lavora insieme.

7.1. Percorso affrontato

Abbiamo affrontato il lavoro di questi 3-4 mesi in questo modo:

1. Una prima parte è stata impiegata per l'analisi e la stesura delle varie librerie che vengono utilizzate dalle varie parti del sistema.
2. La fase successiva è stata quella di implementare il workflow esemplificativo (vedi il punto 6.2) nel linguaggio XPDL.
3. In seguito siamo passati alla realizzazione dell'automa che gestisce l'assegnazione dei compiti e la transazione fra le varie attività (DAEMON) e, in parallelo, la costruzione del CMS come interfaccia principale dell'amministratore di tutto il sistema.
4. Infine siamo passati alla fase di stesura della relazione e costruzione della documentazione delle parti sviluppate.
5. Nel tempo restante ci siamo dedicati allo sviluppo del Monitor, il secondo automa, che controlla i vari processi attivi e i loro tempi di esecuzione avvertendo l'amministratore tramite il CMS e i Feed-RSS a lui recapitati.

Le difficoltà più in particolare…

- XAMPP non è stato il primo pacchetto software che abbiamo utilizzato per realizzare il nostro progetto. Il primo fu Portable Apache MySQL PHP Application (PAMPA) che, però, non si rivelò un'ottima scelta.
- Lo standard XPDL e l'applicativo per definire i workflow si rivelarono non proprio "comodi": un particolare nell'intestazione dei files impediva la giusta formattazione.
- Il linguaggio XSLT ha dei difetti non poco rilevanti: analizzando un file xml formattato con XSLT, cambiando il browser si può osservare come sia interpretato in maniera differente.

Le ultime considerazioni…

- XAMPP è stato utile per poter disporre, in tempi brevi, di un Server di posta (Mercury Mail Exchanger), PHP da riga di comando (PHP-CLI version) e le librerie PEAR.

- Il server di posta è stato utilizzato per gestire meglio le comunicazioni fra il sistema e gli Agent esterni ad esso in sede d'esame per avere delle transazioni più veloci di quelle ottenibili utilizzando un server di posta esterno (per esempio quello di Libero, Alice o altri..)
- Le librerie PEAR e, in particolare la sezione di PHP-Documentor, ci hanno permesso di costruire in tempi brevi la documentazione delle varie librerie; a discapito del tempo impiegato per imparare la sintassi con cui commentare le varie righe di codice.

8. Bibliografia

JaWE	http://jawe.enhydra.org
XAMPP	http://www.apachefriends.org/xampp-en.html
XPDL	http://www.wfmc.org/standards/xpdl.htm
PHP	http://www.php.net/

date.inc

Library Date.inc

funzioni che gestiscono le date e durate

- **Package** lib
- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Copyright** D&F 2007
- **Since** 2007/05/06

date function converti_data($xpdl_datetime) *[line 104]*

Function Parameters:

- *string* **$xpdl_datetime** data fornita in formato xpdl descritto qui sotto: "[XXG] [XXM] [XXY] [XXh] [XXm] [XXs]"

1. XX - valore numerico
2. G - giorni
3. M - mesi
4. Y - anni
5. h - ore
6. m - minuti
7. s - secondi

- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Deprecated** converto la durata dal formato del file xpdl al formato datetime

- **Since** 2007/06/07

date function converti_datetime($datetime) *[line 64]*
Function Parameters:

- *date* **$datetime** data fornita in formato DATETIME

- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Deprecated** converte una data nel formato DATETIME (Y/m/d H:i:s) di MySQL al formato italiano (da noi stabilito) per la visualizzazione)
- **Since** 2007/06/07

date function datetime([$quando = 0]) *[line 44]*
Function Parameters:

- *int* **$quando** è opzionale, contiene la data che si vuole convertire in formato strtotime (numero di secondi) info sulla funzione strtotime(...): http://it.php.net/manual/it/function.strtotime.php

- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Deprecated** ritorna la data in formato DATETIME
- **Since** 2007/06/07

date function get_data_corrente([$err = stringa dell'errore da passare alla funzione errore(..)]) *[line 29]*
Function Parameters:

- *string* **$err** stringa dell'errore da passare alla funzione errore(..)

- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Deprecated** ritorna la data corrente in formato DATETIME usato in MySql
- **Since** 2007/06/07

int function quanti_secondi($datetime) *[line 145]*
Function Parameters:

- *date* **$datetime** data nel formato DATETIME (Y/m/d H:i:s)

- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Deprecated** converto una durata, in formato datetime, nel quantitativo corrispondente di secondi
- **Since** 2007/06/07

bool function verifica_scadenza($data_avvio_activity, $durata_max) *[line 209]*
Function Parameters:

- *date* **$data_avvio_activity** - presa nello script chiamante dal file dello stato.xml nel campo Istance->Status->Date
- *string* **$durata_max** - durata massima dell'attività nel formato della xpdl-durata del file xpdl

- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Deprecated** verifica se la sessione passata è scaduta, oppure no.

- **Since** 2007/06/07

email.inc

Library Email.inc

libreria email support funzioni per gestire la comunicazione via email

- **Package** lib
- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Copyright** D&F 2007
- **Since** 2007/06/14

function email_sessione($mail, $codice) *[line 59]*

Function Parameters:

- **$mail**
- **$codice**

****

Funzioni **********

feed-rss.inc

Library Feed-RSS.inc

pubblicazione dei feed-rss per l'amministratore

- **Package** lib
- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Copyright** D&F 2007
- **Deprecated**

1. DEFINIZIONI:
   - RDF Site Summary ed anche di Really Simple Syndication) è uno dei più popolari formati per la distribuzione di contenuti Web. E' basato su XML, da cui ha ereditato la semplicità, l'estensibilità e la flessibilità.
   - RSS definisce una struttura adatta a contenere un insieme di notizie, ciascuna delle quali sarà composta da vari campi (nome autore, titolo, testo, riassunto, ...). Quando si pubblicano delle notizie in formato RSS, la struttura viene aggiornata con i nuovi dati; visto che il formato è predefinito, un qualunque lettore RSS potrà presentare in una maniera omogenea notizie provenienti dalle fonti più diverse.
2. FRUIZIONE DI UN FEED RSS:
   La fruizione di un documento RSS è un processo molto semplice.
   Le modalità più diffuse sono due:
   - attraverso appositi software che interpretano un feed permettendo agli utenti di visualizzarne i contenuti
   - integrando i contenuti del feed all'interno di un sito Web.

```
1        <   link rel="alternate"           type="application/rss+xml"
title="RSS"           href="http://www.repubblica.it/rss/homepage/rss2.0.xml"
/>
```

3. ESEMPIO DI FEED-RSS di "www.repubblica.it":

```
1        <?   xml version="1.0"           encoding="iso-8859-1"           ?>
2        <rss version="2.0">
3          <channel>
4            <title>Repubblica.it &#62; Homepage</title>
5            <link>http://www.repubblica.it</link>
6            <description>Repubblica.it: il quotidiano online in tempo
reale.</description>
7            <copyright>Copyright 2007 - Gruppo Editoriale
l'Espresso</copyright>
8            <language>it-IT</language>
9            <managingEditor>repubblicawww@repubblica.it (per segnalazioni sui
contenuti del servizio)</managingEditor>
10           <webMaster>repubblicawww@repubblica.it (per segnalazione di
malfunzionamenti del servizio)</webMaster>
11           <lastBuildDate>Tue, 10 Apr 2007 09:26:51 GMT</lastBuildDate>
12           <category
domain="http://www.repubblica.it">news</category>
13           <category
```

```
domain="http://www.repubblica.it">politica</category>
14           <generator>Atex Presweb Builder 4.3</generator>
15           <docs>http://blogs.law.harvard.edu/tech/rss</docs>
16           <ttl>30</ttl>
17           <image>
18             <url>http://www.repubblica.it/images/logo_repubblica.gif</url>
19             <title>Repubblica.it</title>
20             <link>http://www.repubblica.it</link>
21             <width>134</width>
22             <height>19</height>
23           </image>
24           <item>
25             <enclosure
url="http://www.repubblica.it/2007/04/ARCHIVE/homepage/images/sezioni/esteri/afghanis
tan-25/attacco-cdl_HM/ap_10152567_45050.jpg" length="8012"
type="image/jpeg" />
26             <link>http://www.repubblica.it/2007/04/sezioni/esteri/afghanistan-
25/attacco-cdl/attacco-cdl.html</link>
27             <guid
isPermaLink="true">http://www.repubblica.it/2007/04/sezioni/esteri/afghanista
n-25/attacco-cdl/attacco-cdl.html</guid>
28             <title>
29               <![CDATA[
30                 Afghanistan, la Cdl attacca il governo poi Berlusconi ci ripensa:
"Prima l'Italia"
31               ]]>
32             </title>
33             <description>
34               <![CDATA[
35                 <i>La Lega chiede l'impeachment del presidente del Consiglio,
Bonaiuti incalza: "Riferisca alle Camere"<br>Ma alla fine il Cavaliere
richiama tutti all'ordine: "Basta, pensiamo all'interesse
nazionale"<br></i>
36                 <b> Afghanistan, la Cdl attacca il governo poi Berlusconi ci
ripensa: "Prima l'Italia"</b>
37                 <br>
38                 Prodi comunque aveva respinto le accuse: "Utilizzati gli stessi
strumenti del precedente esecutivo"
39                 <br><br>
40                 (22:03 09/04/2007)
41                 ]]>
42             </description>
43             <author>repubblicawww@repubblica.it</author>
44             <category
domain="http://www.repubblica.it">esteri</category>
45             <pubDate>Mon, 09 Apr 2007 20:03:08 GMT</pubDate>
date("r")
46           </item>
47         </channel>
48       </rss>
```

- **Since** 2007/05/06

void function pubblica_feed() *[line 125]*

- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Deprecated** pubblica i vari log sul feed rss in modo da avvisare tempestivamente l'amministratore
- **Since** 2007/06/14

Library Genera_Indirizzo.inc

gestione delle sessioni tramite chiavi alfanumeriche

- **Package** lib
- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Copyright** D&F 2007
- **Since** 2007/07/06

string function crea_chiave() *[line 67]*

- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Deprecated** crea una chiave esadecimale di lunghezza random fra _LSTRINGAMIN e _LSTRINGAMAX. il quoziente e il resto vengono inseriti in coda alla successione creata dopo la loro corrispondente codifica (_STRINGARESTO e _STRINGAQUOZIENTE) infine viene restituita una stringa di questo tipo: 5FFPA388101D0DFC4E10FO5855E70408110NC79F1D0... in cui è codificata il numero del record (id_sessione) dove è registrata la sessione con i dati che servono al demone per iniziare/riprendere l'esecuizione dell'attività.
- **Since** 2006/07/06

int function decodifica($ind) *[line 180]*

Function Parameters:

- *string* **$ind** chiave alfanumerica che identifica la sessione

- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Deprecated** decodifica una chiave creata con la funzione qui sopra per ricavare il numero di record nella tabella sessioni. esempio:
  $ind=5FFPA388101D0DFC4E10FO5855E70408110NC79F1D08767I88E0D4W0FDWQOP31ABOP811

  return 10
- **Since** 2006/07/06

_LSTRINGAMAX = 50 *[line 47]*

- **Var** int
- **Deprecated** Lunghezza massima della stringa alfanumerica generata dalla libreria.
- **Name** _LSTRINGAMAX

_LSTRINGAMIN = 40 *[line 40]*

- **Var** int
- **Deprecated** Lunghezza minima della stringa alfanumerica generata dalla libreria.
- **Name** _LSTRINGAMIN

_NUMCIFRE = 3 *[line 54]*

- **Var** int
- **Deprecated** Numero di cifre per i valori del resto e del quoziente che vengono segnati nella stringa alfanumerica
- **Name** _NUMCIFRE

_STRINGAQUOZIENTE = "1AE" *[line 33]*

- **Var** int
- **Deprecated** Codice che precede il valore esadecimale del quoziente
- **Name** _STRINGAQUOZIENTE

_STRINGARESTO = "0FD" *[line 26]*

- **Var** int
- **Deprecated** Codice che precede il valore esadecimale del resto
- **Name** _STRINGARESTO

logs.inc

Library Logs.inc

funzioni riguardanti i log delle istanze e del workflow in generale

- **Package** lib
- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Copyright** D&F 2007
- **Since** 2007/05/06

void function errore($err) *[line 140]*

Function Parameters:

- *string* **$err** stringa contenente le informazioni dell'errore.
  E' normalmente formattata con:
  "Workflow::nomescript_chiamante: informazioni sull'errore"

- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Deprecated** si occupa di segnare l'errore scegliendo il file dei log esatto.
  La scelta viene fatta fra i log di sistema e quelli riguardanti
  le istanze dei processi (o Workflow).
- **Since** 2007/06/14

include_once **"xml_files.inc"** *[line 22]*

Includo la libreria xml_files perchè è necessaria al funzionamento delle funzioni di

questa libreria.

void function log_evento(&$tmp, $messaggio) *[line 68]*

Function Parameters:

- *object* **&$tmp** - SimpleXMLObject in cui inserire la notifica del nuovo evento
- *string* **$messaggio** - da inserire nella descrizione

- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Deprecated** modifica l'oggetto passato per indirizzo
  inserendo i dati estratti dal secondo parametro (messaggio)

  ESEMPIO DI LOG
  <p>
  <p>

```
1     <   Log>
2       ...
3       <   Event type="error"          >
4         <   Date>   1/04</   Date>
5         <   Description>   il primo evento</   Description>
6         <   Priority>   4</   Priority>
7       </   Event>
8       ...
9     </   Log>
```

- **Since** 2007/06/07

void function notifica($err) *[line 36]*

Function Parameters:

- *string* **$err** stringa dell'errore da passare alla funzione errore(..)

- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Deprecated** richiama la funzione che gestisce gli errori: è solo un alias di errore(err)

- **Since** 2007/06/07

sql.inc

Library Sql.inc

gestione del database sql e le varie query

- **Package** lib
- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Copyright** D&F 2007
- **Since** 2007/05/06

void function sql_connetti() *[line 35]*

- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Deprecated** si connette al dbms senza selezionare un database
- **Since** 2007/06/14

string function sql_create_table($nome_tabella, $query) *[line 169]*

Function Parameters:

- *string* **$nome_tabella** nome della tabella da creare
- *string* **$query** query vera e proprio che la descrive

- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0

- **Deprecated** esegue una query di creazione di una tabella
- **Since** 20070614

void function sql_disconnetti() *[line 55]*

- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Deprecated** si disconnette dal dbms
- **Since** 2007/06/14

object il function sql_query($q) *[line 78]*
Function Parameters:

- *string* **$q** stringa della query SQL

- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Deprecated** esegue una query al database aggiungendo la clausola VALIDO = 1 per verificare se i record analizzati sono ancora validi (a livello logico).
- **Since** 2007/06/14

string function sql_tabellarisultati($risultato) *[line 134]*
Function Parameters:

- *object* **$risultato** - resource della query

- **Author** Fabio Cigliano & Desirèe Armato

- **Version** 1.0
- **Deprecated** costruisce una tabella intestata dei record estratti con la query precedentemente eseguita
- **Since** 2007/06/14

upload.inc

Library Upload.inc

gestione dell'upload di file al nostro server

- **Package** lib
- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Copyright** D&F 2007
- **Link** http://it.php.net/manual/it/features.file-upload.php
- **Deprecated** BASI TEORICHE: METODO POST PER CARICAMENTO DI FILE:
  Questa caratteristica permette di caricare sia file di testo
  che binari. Utilizzando le funzioni di PHP per l\"autenticazione
  e manipolazione dei file, è possibile avere pieno controllo su chi
  ha i permessi per caricare un file e su ciò che deve essere fatto
  una volta che il file è stato caricato.
  Note relative alla configurazione:
  Si vedano i parametri file_uploads, upload_max_filesize,
  upload_tmp_dir, post_max_size e max_input_time nel php.ini
  A partire dal PHP 4.1.0 esiste la variabile globale $_FILES.
  (Utilizzare $HTTP_POST_FILES nelle versioni precedenti).
  Queste matrici contengono tutte le informazioni sui file inviati.
  Di seguito verrà illustrato il contenuto di $_FILES nel caso
  dell\"esempio precedente. Si noti che si assume come nome del
  file inviato userfile, come nell\"esempio precedente.
  Nella realtà questo può assurmere nome.
  $_FILES[\"userfile\"][\"name\"] - Il nome originale del file
  sulla macchina dell\"utente.
  $_FILES[\"userfile\"][\"type\"] - Il mime-type del file, se il
  browser fornisce questa informazione.
  Un esempio potrebbe essere "image/gif".
  $_FILES[\"userfile\"][\"size\"] - La dimensione, in bytes, del file
  caricato.
  $_FILES[\"userfile\"][\"tmp_name\"] - Il nome del file temporaneo in cui
  il file caricato è salvato sul server.
  $_FILES[\"userfile\"][\"error\"] - Il codice di errore associato all\"
  upload di questo file.

```
1        <?php
2          $uploaddir = "     /var/www/uploads/\";
3          $uploadfile = $uploaddir .
basename($_FILES[\"     userfile\"][\"          name\"]);
4          if (move_uploaded_file($_FILES[\"     userfile\"][\"          tmp_name\"],
$uploadfile)) {
5              echo "     File is valid, and was successfully uploaded.n";
6          } else {
7              echo "     Possibile attacco tramite file upload!n"     ;
8          }
```

```
9           echo \"     Alcune infromazioni di debug:\";
10          print_r($_FILES);
```

- **Since** 2007/06/14

array function dirlist([$dir = './']) *[line 214]*

Function Parameters:

- *string* **$dir** nome della directory da analizzare

- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Deprecated** elenca, ricorsivamente, i files e le cartelle presenti nella directory e sotto-directory specificate
- **Since** 2007/06/14

string function form_carica_file($nomescript_ricevente, [$numero_file = 1], [$opzione = 0], [$nomeopzione = "op"], [$estensione = ""], [$target = "_blank"]) *[line 88]*

Function Parameters:

- *string* **$nomescript_ricevente** script da richiamare in cui si ricevono i file in upload
- *int* **$numero_file** numero dei files da ricevere
- *int* **$opzione** opzione (se necessaria) da specificare nello script ricevente
- *string* **$nomeopzione** nome del campo che specifica l'opzione nello script ricevente
- *string* **$estensione** estensione dei file da caricare
- *string* **$target** metodo con cui aprire lo script ricevente (_blank , _self , _parente , <frame>)

- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0

- **Deprecated** ritorna la form per la selezione dei file da uploaddare
- **Since** 2007/06/14

string function indirizzo_assoluto($path) *[line 249]*

Function Parameters:

- *string* **$path** indirizzo relativo

- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Deprecated** passando l'indirizzo relativo di un file, viene ricostruito l'indirizzo assoluto
- **Since** 2007/06/14

string function ricevi_file([$filename = ""]) *[line 137]*

Function Parameters:

- *string* **$filename** nome con cui salvare il file ricevuto

- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Deprecated** riceve il file caricato (con indice $files_pointer) la funzione può essere richiamata più volte nello script ricevente per ricevere tutti i files richiesti dalla form precedentemente stampata.
- **Since** 2007/06/14

vars.inc

Library Vars.inc

le variabili globali del sistema

- **Package** lib
- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Copyright** D&F 2007
- **Since** 2007/05/29

bool function in_debug() *[line 74]*

- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Deprecated** debug = 1 vero. debug = 0 modalità normale.
- **Since** 2007/04/28

Library verifica_sessione.php

gestione delle sessioni e estrazione dei dati che mi permettono di accedere al workflow.

- **Package** lib
- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Copyright** D&F 2007
- **Since** 2007/05/06

xml_files.inc

Library xml_files.inc

gestione dei file xml con simplexml

- **Package** lib
- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Copyright** D&F 2007
- **Since** 2007/05/06

string function archivia_tabella($sql_result, $xml_destinazione) *[line 204]*

Function Parameters:

- *object* **$sql_result** resource di php-mysql
- *string* **$xml_destinazione** pathname del file in cui archiviare i file

- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Deprecated** archivia i record passati nel file xml specificato
- **Since** 2007/06/11

void function cambia_stile_filexml($filexml, $file_stile) *[line 64]*

Function Parameters:

- *string* **$filexml** percorso del file in cui modificare il tag stylesheet

- *string* **$file_stile** nome dei file xsl contenuto nella cartella del server

    - **Author** Fabio Cigliano & Desirèe Armato
    - **Version** 1.0
    - **Deprecated** cambia il campo ?xml-stylesheet del file xml passato con il file dello stile richiesto (contenuto nella cartella Styles/). Esempio: cambia_stile_filexml("directory/stato.xml","logs.xsl");
    - **Since** 2007/06/14

void function indenta_filexml($filexml, [$modo = 1]) *[line 163]*

Function Parameters:

- *string* **$filexml** pathname del file xml da indentare
- *int* **$modo** modo di formattazione da passare alla classe xml_viewer

  N.B. di default è 1.
  - Per fare in modo che la funzione imposti i nomi dei tag con la prima lettera maiuscola (vedi ucwords(...)) settare a 0.
  - Per fare in modo che la funzione restituisca tutti i tag in minuscolo settare a 1.
    </pre>

    - **Author** Fabio Cigliano & Desirèe Armato
    - **Version** 1.0
    - **Deprecated** indenta un file xml secondo le specifiche dello standard XML ver.1.0
    - **Since** 2007/06/07

string function status_get_stato() *[line 26]*

    - **Author** Fabio Cigliano & Desirèe Armato

- **Version** 1.0
- **Deprecated** ritorna lo stato dell'activity corrente, contenuto nel file stato.xml
- **Since** 2007/06/14

void function status_salva_filexml([$file = ""]) *[line 41]*

Function Parameters:

- **$file**

- **Author** Fabio Cigliano & Desirèe Armato
- **Version** 1.0
- **Deprecated** salva il file xml per come è stato modificato e richiama la funzione per indentarlo
- **Since** 2007/06/14

Package lib Classes

Class Agent
[line 1110]

- **Package** lib

void function Agent::call($aa_sfunc, $aa_cfunc, $aa_sfunc_args) *[line 1111]*
Function Parameters:

- **$aa_sfunc**
- **$aa_cfunc**
- **$aa_sfunc_args**

void function Agent::init() *[line 1150]*
void function Agent::listen($aa_event, $aa_cfunc, $aa_sfunc_args) *[line 1145]*
Function Parameters:

- **$aa_event**
- **$aa_cfunc**
- **$aa_sfunc_args**

Class Services_JSON

[line 397]

Converts to and from JSON format.

Brief example of use:

```
1       // create a new instance of Services_JSON
2       $json = new Services_JSON();
3
4       // convert a complexe value to JSON notation, and send it to the browser
5       $value = array('foo', 'bar', array(1, 2, 'baz'), array(3, array(4)));
6       $output = $json->   encode($value);
7
8       print($output);
9       // prints: ["foo","bar",[1,2,"baz"],[3,[4]]]
10
11      // accept incoming POST data, assumed to be in JSON notation
12      $input = file_get_contents('php://input', 1000000);
13      $value = $json->   decode($input);
```

- **Package** lib

Constructor *void* function Services_JSON::Services_JSON([$use = 0]) *[line 415]*

Function Parameters:

- *int* **$use** object behavior flags; combine with boolean-OR

  possible values:
  - SERVICES_JSON_LOOSE_TYPE: loose typing.
    "{...}" syntax creates associative arrays
    instead of objects in decode().
  - SERVICES_JSON_SUPPRESS_ERRORS: error suppression.
    Values which can't be encoded (e.g. resources)
    appear as NULL instead of throwing errors.
    By default, a deeply-nested resource will
    bubble up with an error, so all return values
    from encode() should be checked with isError()

constructs a new JSON instance

mixed function Services_JSON::decode($str) *[line 766]*

Function Parameters:

- *string* **$str** JSON-formatted string

decodes a JSON string into appropriate variable

- **Access** public

mixed function Services_JSON::encode($var) *[line 519]*

Function Parameters:

- *mixed* **$var** any number, boolean, string, array, or object to be encoded. see argument 1 to Services_JSON() above for array-parsing behavior. if var is a strng, note that encode() always expects it to be in ASCII or UTF-8 format!

encodes an arbitrary variable into JSON format

- **Access** public

void function Services_JSON::isError($data, [$code = null]) *[line 1047]*

Function Parameters:

- **$data**
- **$code**

Class Services_JSON_Error
[line 1062]

- **Package** lib

Constructor *void* function Services_JSON_Error::Services_JSON_Error([$message = 'unknown error'], [$code = null], [$mode = null], [$options = null], [$userinfo = null]) *[line 1064]*

Function Parameters:

- **$message**
- **$code**
- **$mode**
- **$options**
- **$userinfo**

Class xml_viewer
[line 17]

Library xml_viewer.inc

Teinos srl - classe_xml.inc v.1.3

visualizzatore di file xml (parsing)

- **Package** lib
- **Author** Fabio Cigliano
- **Version** 1.3
- **Copyright** Teinos srl 2006

- **Since** 2006/07/07

xml_viewer::$depth

mixed = [line 24]

xml_viewer::$flag

mixed = [line 27]

xml_viewer::$modo

mixed = [line 28]

xml_viewer::$pointer

mixed = [line 25]

xml_viewer::$ris

mixed = [line 26]

xml_viewer::$xml_codifica

mixed = [line 23]

xml_viewer::$xml_file

mixed = [line 22]

xml_viewer::$xml_parser

mixed = [line 21]

Constructor *void* function xml_viewer::xml_viewer([$file = ""], [$codifica = "UTF-8"]) *[line 41]*

Function Parameters:

- *string* **$file** se definito, richiama la funzione di visualizzazione
- *string* **$codifica**

- **Author** Fabio Cigliano
- **Version** 1.0
- **Deprecated** costruttore della sottoclasse xml_viewer
- **Since** 2006/07/07

void function xml_viewer::spaziatura() *[line 243]*

- **Author** Fabio Cigliano
- **Version** 1.0
- **Deprecated** stampa una spaziatura di n tabulazioni
- **Since** 2006/07/06

void function xml_viewer::visualizza([$modo = 1]) *[line 68]*

Function Parameters:

- *int* **$modo** modo = 0 cambia "case" altrimenti lascia invariato

- **Author** Fabio Cigliano
- **Version** 2.0
- **Deprecated** visualizza il file xml aprendolo con i gestori definiti in seguito
- **Since** 2006/07/07

void function xml_viewer::xml_apre($xml, $tag, $attributi) *[line 114]*

Function Parameters:

- *object* **$xml** parser xml
- *string* **$tag** tag di apertura
- *string* **$attributi** - proprietà del tag di apertura letto

- **Author** Fabio Cigliano

- **Version** 1.0
- **Deprecated** gestore del tag di apertura del file xml
- **Since** 2006/07/06

void function xml_viewer::xml_chiude($xml, $tag) *[line 170]*
Function Parameters:

- *object* **$xml** parser xml
- *string* **$tag** tag di apertura

- **Author** Fabio Cigliano
- **Version** 1.0
- **Deprecated** gestore del tag di chiusura del file xml
- **Since** 2006/07/06

void function xml_viewer::xml_dato($xml, $dato) *[line 218]*
Function Parameters:

- *object* **$xml** parser xml
- *string* **$dato** gestore del dato contenuto fra i tag

- **Author** Fabio Cigliano
- **Version** 1.0
- **Deprecated** gestore del dato contenuto fra i tag di apertura e chiusura,
- **Since** 2006/07/06

Appendices

Appendix A - Class Trees

Package lib

Agent

- Agent

Services_JSON

- Services_JSON

Services_JSON_Error

- PEAR_Error
  - Services_JSON_Error

xml_viewer

- xml_viewer

Index

A

C

D

E

F

G

I

L

N

P

Q

R

S

U

V

X

__