XFlow: an Xml-Based Document-Centric Workflow

Andrea Marchetti[1], Maurizio Tesconi[1], Salvatore Minutoli[1]

[1]CNR, IIT Department, Via Moruzzi 1, I-56124, Pisa, Italy
{andrea.marchetti, salvatore.minutoli, maurizio.tesconi}@iit.cnr.it

Abstract. This paper aims at investigating on an appropriate framework that allows the definition of workflows for collaborative document procedures. In this framework, called XFlow and largely based on XSLT Processing Model, the workflows are described by means of a new XML application called XFlowML (XFlow Markup Language). XFlowML describes the document workflow using an agent-based approach. Each agent can participate to the workflow with one or more roles, defined as XPath expressions, based on a hierarchical role chart. An XFlowML document contains as many templates as agent roles participating to the workflow. The document workflow engine constitutes the run-time execution support for the document processing by implementing the XFlowML constructs. A prototype of XFlow has been implemented with an extensive use of XML technologies (XSLT, XPath, XForms, SVG) and open-source tools (Cocoon, Tomcat, mySQL).

1 Introduction

The problem of processing documents has long been recognized to be a critical aspect in the enterprise productivity ([3], [6], [7], [8], [9]). The management of documents becomes more difficult when it involves different actors, possibly in a decentralized working environment, with different tasks, roles and responsibilities in different document sections.
Many enterprise day-to-day operations can be viewed as a series of steps involving the filling out of appropriate forms by different actors, sometimes with a concurrent processing. We can expect that implementing an effective system that supports these forms flows will result in considerable cost savings for enterprises. We propose a solution based on a complete independence between workflow definition and workflow engine which supports the management of simultaneous document workflows by largely using XML technologies.

2 Overview: Definitions, Concepts, Terminology

Before illustrating our approach, we give some remarks on the terminology used.
A *workflow* is “the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules”.

As *Document-centric Workflow* or *Document Workflow* (DW) we refer to a particular workflow in which all activities, made by the agents, turn out to documents compilation. It can be viewed as the automation and administration of particular documents procedures ([1],[3],[9]). In other words, a DW can be seen as a process of cooperative authoring where the document can be the goal of the process or just a side effect of the cooperation.
Through a DW, a document life-cycle is tracked and supervised, continually providing document compilation actions control. In this environment a document travels among *agents* who essentially carry out the pipeline receive-process-send activity. There are two types of agents: *external agents* are human or software actors which perform activities dependent from the particular DW, and *internal agents* are software actors providing general-purpose activities useful for any DW and, for this reason, implemented directly into the system. An external agent *executes* some processing using the document content and possibly other data, *updates* the document inserting the results of the preceding processing, *signs* the updating and finally *sends* the document to the next agent(s).
Figure 1 illustrates a generic document workflow diagram where external and internal agents cooperate exchanging documents according to defined procedural rules.

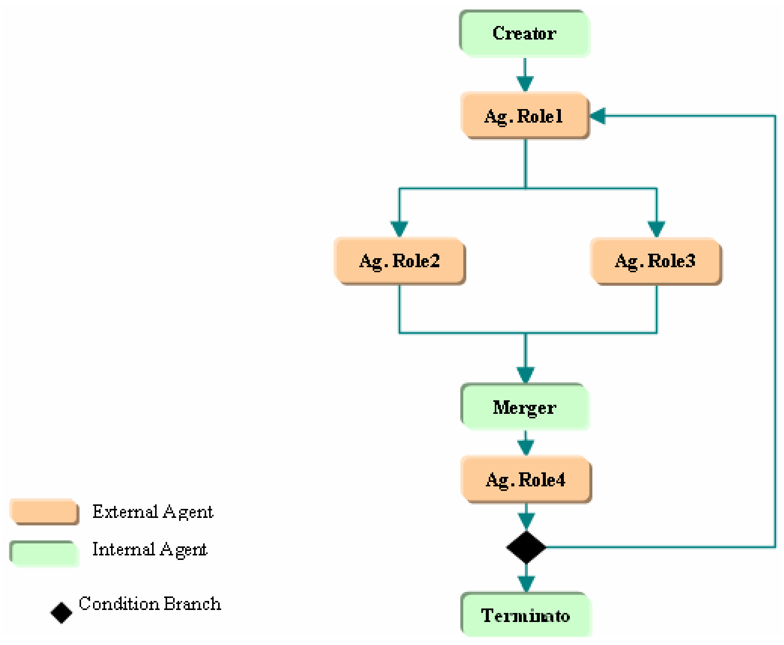


Fig. 1. A generic document workflow

Internal agents perform general functionalities such as *creating* a document belonging to a particular DW, populating it with some initial data, *duplicating* a document to be sent to multiple agents, *splitting* a document and sending partitions to different agents, *merging* duplicated documents coming from multiple agents, *aggregating* document fragments, *terminating* operations on the document.

3 Document Workflow Framework

Our document workflow framework is based on document-centric model where all the activities, made by the agents, turn out to document compilation.
During its life the document passes through several phases, from its creation to the end of its processing. The state diagram in figure 2 describes the different states of the document instances. At the starting point of the document instance life cycle there is a creation phase, in which the system raises a new instance of a document with several information attached (such as the requester agent data). The document instance goes into *pending* state. When an agent gets the document, it goes into *processing* state in which the agent compiles the parts of his concern. If the agent, for some reason, doesn't complete the instance elaboration, he can save the work performed until that moment and the document instance goes into *freezing* state. If the elaboration is completed (submitted), or cancelled, the instance goes back into *pending* state, waiting for a new elaboration.

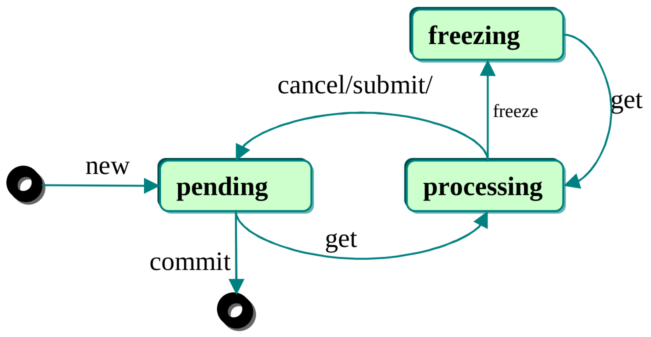


Fig. 2. State diagram of a document instance.

The *DW engine* has the task of managing all the functionalities to support agent activities. In order to design a *DW engine* independent from the single DW, it's necessary to isolate the information of each DW inside an XML document. We will see in detail the essential components necessary to describe a DW.
Summarizing, the *DW framework* is composed of three parts:

- The DW Environments (Agents participating to the DW)
- The DW Engine
- The DW Data (DW descriptions + Documents created by the DW)

3.1 Document Workflow Description

The description of a DW can be seen as an extension of the XML document class. A class of documents, created in a DW, share the schema of their structure, as well as the definition of the procedural rules driving the DW and the list of the agents attending to the DW. Therefore, in order to describe a DW, we need four components:

- a *schema* of the documents involved in the DW;
- the agent roles chart, called *role chart*, i.e. the set of the external and internal agents, operating on the document flow. Inside the role chart these agents are organized in roles and groups in order to define who has access to the document. This component constitutes the DW environment;
- a *document interface description* used by external agents to access the documents. This component also allows to check the access to the document resource;
- a *document workflow description* defining all the paths that a document can follow in its life-cycle, the activities and policies for each role.

Furthermore the system for keeping track of document instances history (including the agents that manipulated the document) and document state during its whole flow path needs respectively a *Log* and *Metadata* component. The Metadata component represents the document current state. Every time a document changes its state (see Figure 1) the Metadata is first saved into Log component and then updated. These last two documents are produced automatically by the DW engine. For each component we define a declarative language, using XML. Hence, each document belonging to a DW, during all its life-cycle ,will be associated with six documents, as indicated in figure 3.

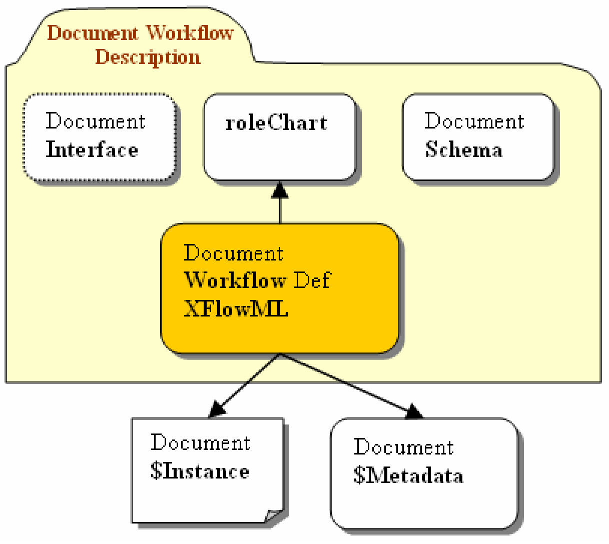


Fig. 3. Document Workflow Description

3.1.1 Document Schema (XML Schema)

Document schema describes the structure and the data-types of the documents participating to the flow. *Document schema* will be described using XML Schema.

3.1.2 Document Interface (XForms, Web Service)

This document describes, for each agent role, the interface to the document. The *document interface* for *external human agents* relies upon Web Modules technologies [2], and *external software agents* make use of Web Services technologies (WSDL, SOAP). In the first solutions we adopted XForms technology, promoted by W3C.

3.1.3 Role Chart: Agent Role Declaration

The *role chart* is an XML document containing the description of all actors (agents) that participate to the workflow. Each actor has a *role* and a *unique identifier*. Roles are organized in the role chart hierarchically. Each agent can participate to the workflow with one or more roles, therefore it can appear in one or more role chart positions. The role chart schema is depicted in figure 4.

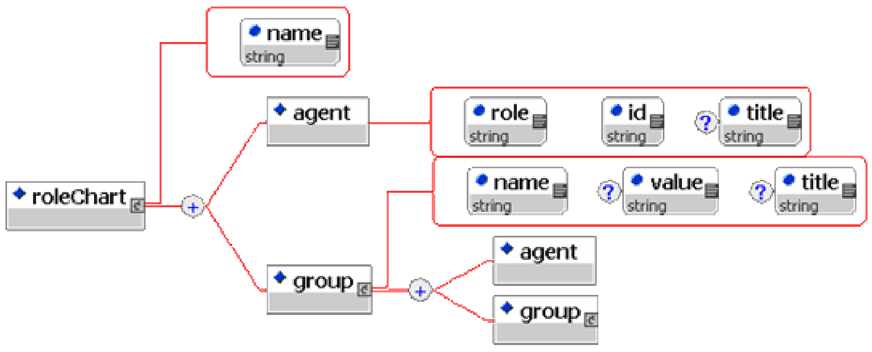


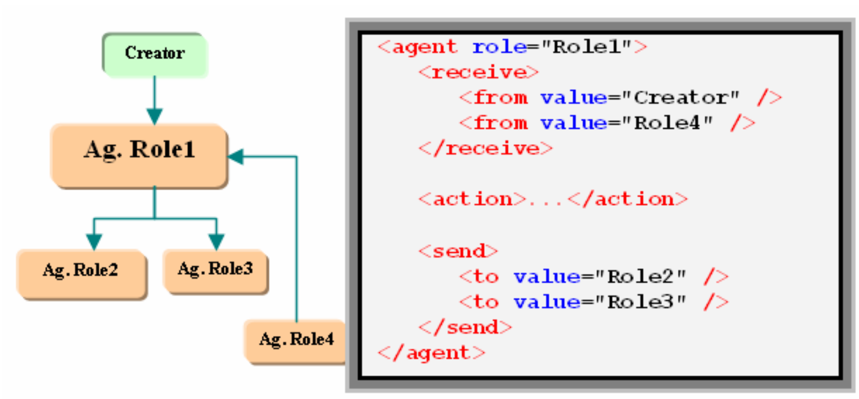
Fig. 4. The Role Chart Schema

Finally the *document workflow description* is a document based on a new XML application (XFlowML Xml document workFlow Markup Language) suitably defined for this purpose.

3.1.4 XFlow: Document Workflow Definition (XFlowML)

For the definition of a language to describe complex document flows we analyzed several syntaxes and approaches. A possible solution was to use a notation similar to concurrent languages, using statements like fork and join to describe flows. Another choice was to describe the document flow from the *point of view of the agents*. To describe a document flow it is sufficient to accurately describe all the agents and all the operations any agent can perform on the document instance. This way of describing the flow resembles XSL syntax, where actions performed by various

agents are similar to the templates to apply to the elements of an XML document. Our basic decision to *represent flows as XML documents*, led us to choose the second option, since with XML, due to its intrinsic hierarchical notation, it is more straightforward to represent lists rather than graphs (other approaches which emphasize the role of XML can be found in [4], [5], [10], [11], [12]). Taking as a simple example the generic flow depicted in Figure 1, we will have to supply as many descriptions as the agents roles involved in the process. For instance, in the description of the external agent with role1 (*Ag. Role1*), we must specify that it can receive documents from *Creator* or *Ag.Role4*, and send it to *Ag.Role2* and *Ag.Role3*.
Figure 5 shows both the graphical representation and the XML notation of the Agent Role1.



```
<agent role="Role1">
   <receive>
      <from value="Creator" />
      <from value="Role4" />
   </receive>

   <action>...</action>

   <send>
      <to value="Role2" />
      <to value="Role3" />
   </send>
</agent>
```

Fig. 5. Description of Agent Role1 (see Fig.1)

To describe document flow we adopted a XML dialect, called XFlowML, largely based on XSL-Syntax.
A XFlowML document is composed of a list of *internal* or *external* `agent`s. Each `agent` has a mandatory attribute `role`, containing a XPath expression referring to the *rolechart*. Other optional attributes specify if the agent has to sign the document (`sign`), and the maximum time the agent is allowed to keep the document (`timeout`).
When an agent requests a document, the DW Engine matches the agent's role on XFlow document and processes the three section: receive, action and send.
In the `receive` section the `from` elements identify from which agent roles the document can be received. The roles of the agents are coded as XPath expressions. The `receive` section is optional because it's necessary only to verify if the agent can really receive the current document.
In the `action` section there are one or more `permission` elements defining the access policies to the document fields.
The `send` section contains all the possible receivers of the document. The document can be sent simultaneously to several agents by using a sequence of `to` elements.

In order to increase the flexibility and power of the language, thus allowing for an easy definition of more complex DWs, we introduced the conditional `<if>` and `<choose>` statements which adopt the XSLT syntax and can be specified in any agent section (i.e. `receive`, `action`, `send`). Test attributes can contain any XPath expression which returns a Boolean, and it is possible to refer document *Metadata* or document *instance*. For distinguishing the referred document the 'test' XPath expression will begin with two different prefixes: respectively $Metadata and $Instance.
A typical use of these conditional statements is in `send` element, when we have to send the document to two different agent depending on the value of a document field previously filled out (see Figure 6).

```
<send>
  <xsl:choose >
    <xsl:when test="$instance//financeReview[approved='true']">
      <to select="//agent[@role='manager']"
    <xsl:when>
    <xsl:otherwise>
      <to value = "//agent[@role='employee'][@id=$instance//employee/@id]">
    <xsl:otherwise>
  </xsl:choose>
</send>
```

Fig. 6. The send section with XSLT conditional statements

3.1.5 Metadata and Log Components

For each document instance that participates to a specific DW, additional information (that we have called metadata) is stored together with information needed to reconstruct the document history (Log) that consists of all the document transitions during its processing, including information about actors involved. Log permits to undo actions. Every time a document is submitted by an agent, the differences with the previous version of the document are generated and saved. Only when the document life cycle is completed and it is archived all this data is deleted.

3.2 The Document Workflow Engine (DWE)

The document workflow engine constitutes the run-time support for the DW, it implements the *internal agents,* the support for *agent's activities,* and some *system modules* that the external agents have to use to interact with the DW system. Also, the engine is responsible for two kinds of documents useful for each document flow: the *documents system logs* and the *documents system metadata.*

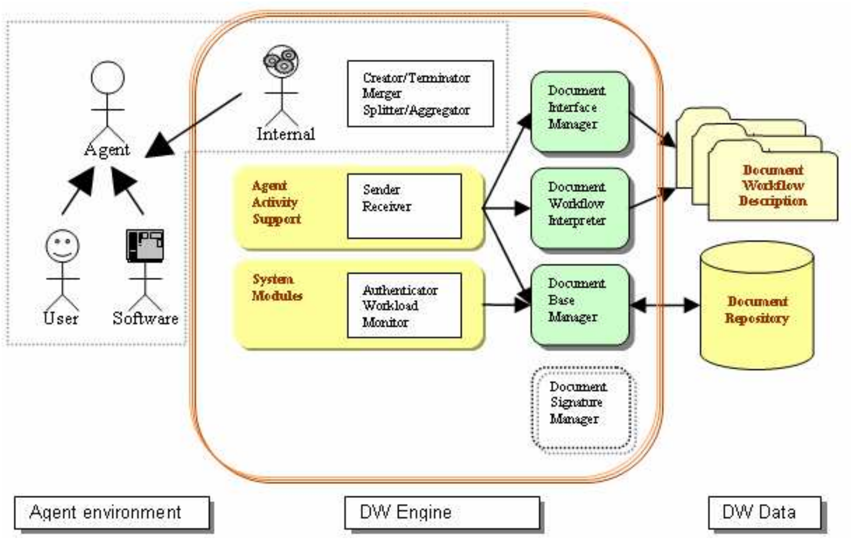


Fig. 7. Document Workflow Framework

3.2.1 Agent's Activities Support

These are the two modules, called *Sender* and *Receiver*, supporting the activities of sending to, and receiving from the current agent1.
The *Sender* has to prepare and send the document (identified by an URN) requested by an external agent. It checks the agent rights, verifies if the document instance is still available, analyzes/interprets the workflow description to generate an adapted document, using the agent's role and access rights to determine which are the parts of the stored document to be included (using XForms for a human agent and SOAP for a software agent).
The *Receiver* gets the document from the handling agent in consequence of a submit, freeze or cancel command. It determines the roles of the next agents to whom the document must be sent.
Both modules use the *DW Interpreter* which transforms the XFlow document into a XSLT stylesheet. The generated stylesheet is applied to the Rolechart document producing the role agent's activities.

4 Case Study: a Travel Request

To illustrate the usefulness of our system, a case study is set up and briefly outlined.
One typical activity in a research institute is the participation in conferences or seminars. In this case, the employee (researcher) must obtain the proper authorization,

[1] The name of these modules are given with the point of view of DWE

involving the approval of the office manager, administrative verification, and final approval by the director.

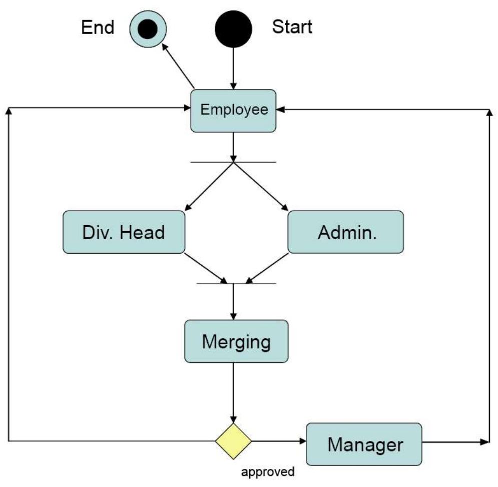


Fig. 8. Travel Request Graph

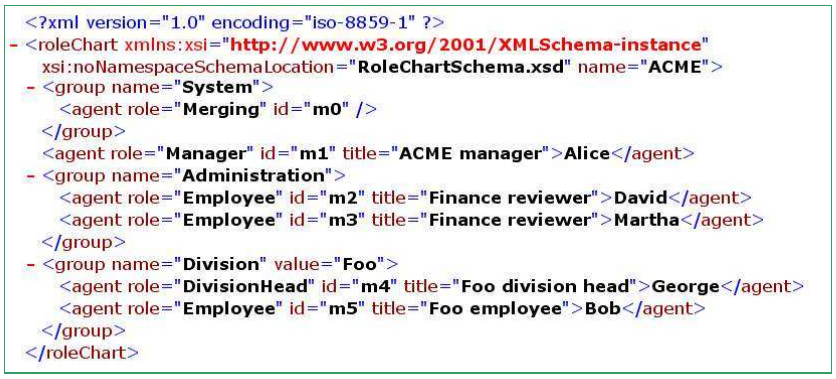
In terms of workflow, this consists of filling out several mandatory fields (such as purpose, destination, duration and dates of the trip and estimated daily traveling allowance) by the employee. Some employee's data, such as name and division can be pre-filled by the application.
Then the document must be approved by the office manager and by the administration.
These two activities are independent and can be performed concurrently, therefore the document is duplicated in different copies, each one sent to the appropriate actor.
Afterwards the document is properly recomposed from the merging agent and sent to the director for authorization.
Here each actor involved fills out a different part of the form. The method used to visualize the document consists of an adaptive user interface that shows the fields to be filled out.
Note that the Administration is actually a role, designating a set of individuals who can perform the task, while each of the other actors is a specific person.

4.1 Travel Request Flow Description

In order to describe this flow it's necessary to create a schema as well as a rolechart and a xflow document.

4.1.1 The Rolechart Document

A rolechart document contains all the agents organized by groups and roles. The roles are: employee, division head, administrator and manager.



```
  <?xml version="1.0" encoding="iso-8859-1" ?>
- <roleChart xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="RoleChartSchema.xsd" name="ACME">
  - <group name="System">
      <agent role="Merging" id="m0" />
    </group>
    <agent role="Manager" id="m1" title="ACME manager">Alice</agent>
  - <group name="Administration">
      <agent role="Employee" id="m2" title="Finance reviewer">David</agent>
      <agent role="Employee" id="m3" title="Finance reviewer">Martha</agent>
    </group>
  - <group name="Division" value="Foo">
      <agent role="DivisionHead" id="m4" title="Foo division head">George</agent>
      <agent role="Employee" id="m5" title="Foo employee">Bob</agent>
    </group>
  </roleChart>
```

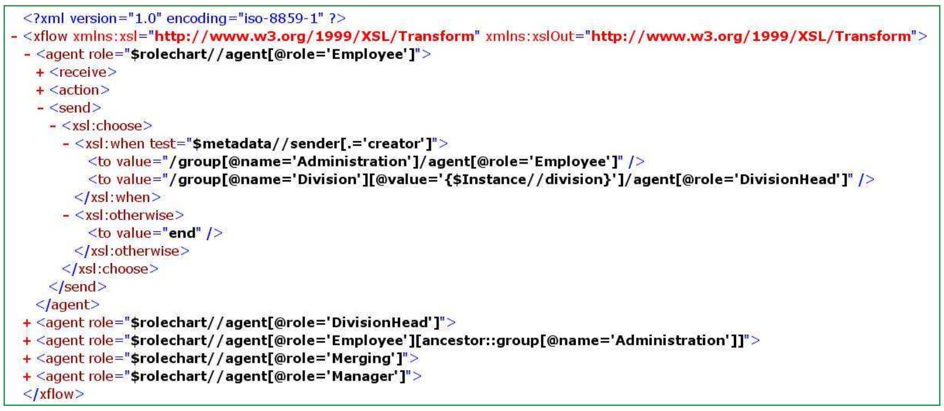
Fig. 9. Rolechart document.

Let's suppose that in our firm ACME there is only the division "Foo" where George is the division head and Bob is the only employee. Two employees work in administration. The manager of ACME's firm is Alice. Figure 9 describes this situation.

4.1.2 The XFlow Document

The travel request graph is codified in xflow document. It's organized in 5 sections corresponding to the 5 agents roles: employee, division head, administration employee, merging (internal agent necessary for merging the document coming from division head and administration, see Figure 8) and the manager.

In the figure 10 we can see the *send* section of the "Employee" agent. In this section it is declared that a document submitted by an Employee can follow two paths. If the document has just been created, it will be duplicated and sent to an administration employee and to the division head of the employee, otherwise the document will be archived because the process is completed.



```
  <?xml version="1.0" encoding="iso-8859-1" ?>
- <xflow xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:xslOut="http://www.w3.org/1999/XSL/Transform">
  - <agent role="$rolechart//agent[@role='Employee']">
    + <receive>
    + <action>
    - <send>
      - <xsl:choose>
        - <xsl:when test="$metadata//sender[.='creator']">
            <to value="/group[@name='Administration']/agent[@role='Employee']" />
            <to value="/group[@name='Division'][@value='{$Instance//division}']/agent[@role='DivisionHead']" />
          </xsl:when>
        - <xsl:otherwise>
            <to value="end" />
          </xsl:otherwise>
        </xsl:choose>
      </send>
    </agent>
  + <agent role="$rolechart//agent[@role='DivisionHead']">
  + <agent role="$rolechart//agent[@role='Employee'][ancestor::group[@name='Administration']]">
  + <agent role="$rolechart//agent[@role='Merging']">
  + <agent role="$rolechart//agent[@role='Manager']">
  </xflow>
```

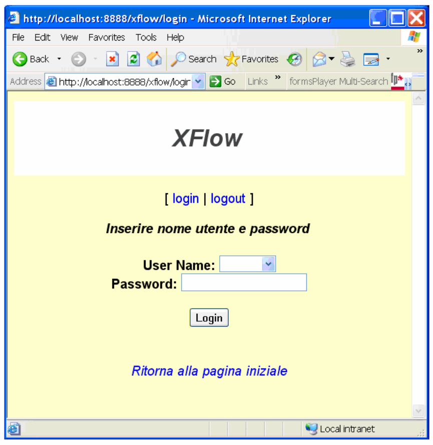
Fig. 10. XFlow document

5 Implementation Overview

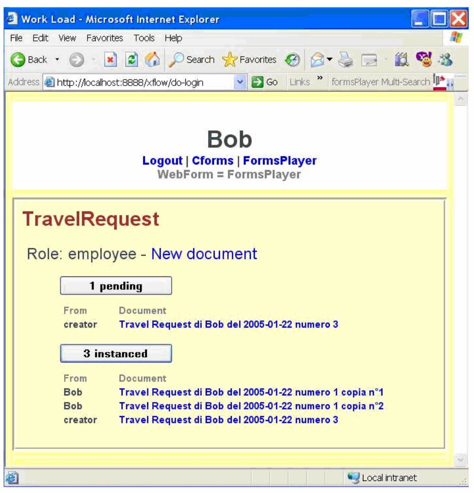
5.1 The Client Side: External Agent Interaction

Our system is currently implemented as a web-based application where the human external agents interact with system through a web browser. All the human external agents attending the different document workflows are the users of the system. Once authenticated through user/psw (Fig. 11A) the user accesses his/her workload area (Fig. 11B) where the system lists all his/her pending documents sorted by flow.
The system shows only the flows to which the user has access.
From the workload area the user can browse his/her documents and select some operations such as:

- select and process a pending document (Fig. 12C)
- create a new document partly filled with his/her data.
- display a graph representing a DW of a previously created document, highlighting the current position of the document (Fig. 12D). This information is rendered as an SVG image.


http://localhost:8888/xflow/login - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8888/xflow/login Go Links formsPlayer Multi-Search
XFlow
[login | logout]
Inserire nome utente e password
User Name:
Password:
Login
Ritorna alla pagina iniziale
Local intranet

a


Work Load - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8888/xflow/do-login Go Links formsPlayer Multi-Search
Bob
Logout | Cforms | FormsPlayer
WebForm = FormsPlayer
TravelRequest
Role: employee - New document
1 pending
3 instanced
Local intranet

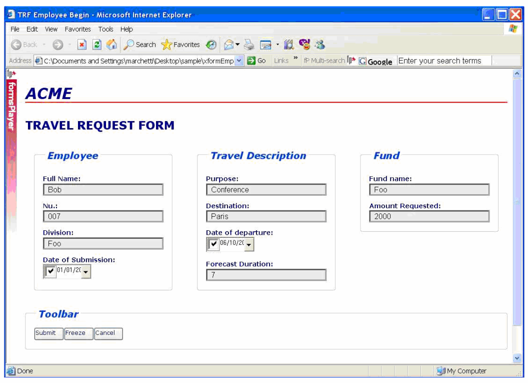
From	Document
creator	Travel Request di Bob del 2005-01-22 numero 3

From	Document
Bob	Travel Request di Bob del 2005-01-22 numero 1 copia n°1
Bob	Travel Request di Bob del 2005-01-22 numero 1 copia n°2
creator	Travel Request di Bob del 2005-01-22 numero 3

b

Fig. 11. Some screenshots of XFlow

The form used to process the documents is rendered with XForms (See Fig. 12C). XForms can communicate with the server by means of XML documents and is capable of displaying the document with a user interface that can be defined for each type of document. XForms is a recommendation of the W3C for the specification of Web forms. In XForms the description of how the form is displayed is separated from the description of what the form must do, so it is easy to use different type of views depending on the platform and on the document. A browser with XForms capabilities will receive an XML document that will be displayed according to the specified template, then it will let the user edit the document and finally it will send the modified document to the server.


TRF Employee Begin - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\marchetti\Desktop\sample\xformEmp
Go Links fP Multi-search Google Enter your search terms
formsPlayer
Done
My Computer

ACME

TRAVEL REQUEST FORM

Employee

Full Name: Bob

Nu.: 007

Division: Foo

Date of Submission: ✓ 01/01/20

Travel Description

Purpose: Conference

Destination: Paris

Date of departure: ✓ 06/10/20

Forecast Duration: 7

Fund

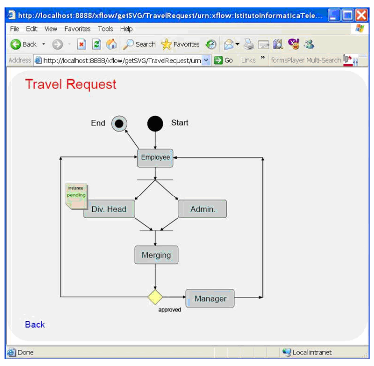
Fund name: Foo

Amount Requested: 2000

Toolbar

Submit Freeze Cancel

c


http://localhost:8888/xflow/getSVG/TravelRequest/urn:xflow:IstitutoInformaticaTele...
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8888/xflow/getSVG/TravelRequest/urn
Go Links formsPlayer Multi-Search
Travel Request
End
Start
Employee
instance pending
Div. Head
Admin.
Merging
approved
Manager
Back
Done
Local intranet

d

Fig. 12. Some screenshots of XFlow

5.2 The server side

The server-side is implemented with Apache Tomcat, Apache Cocoon and MySql. Tomcat is used as the web server, authentication module (when the communication between the server and the client needs to be encrypted) and servlet container. Cocoon is a publishing framework that uses the power of XML.The entire functioning of Cocoon is based on one key concept: component pipelines. A pipeline is composed by a series of steps, which consists of taking a request as input, processing and transforming it, and then giving the desired response. The pipeline components are generators, transformers, and serializers. A Generator is used to create an XML structure from an input source (file, directory, stream ...) A Transformer is used to map an input XML structure into another XML structure (the most used is XSLT transformer). A Serializer is used to render an input XML structure into some other format (not necessarily XML)
MySql is used for storing and retrieving the documents and the status of the documents.
There are some modules that allow the interaction with the user agents.
The *Authenticator* and *WorkloadSender* modules use XHTML to display data. The *Receive* and *Sender Document* modules use XForms to exchange XML document with human agents and the SOAP protocol to exchange documents with software agents. (fig. 13).

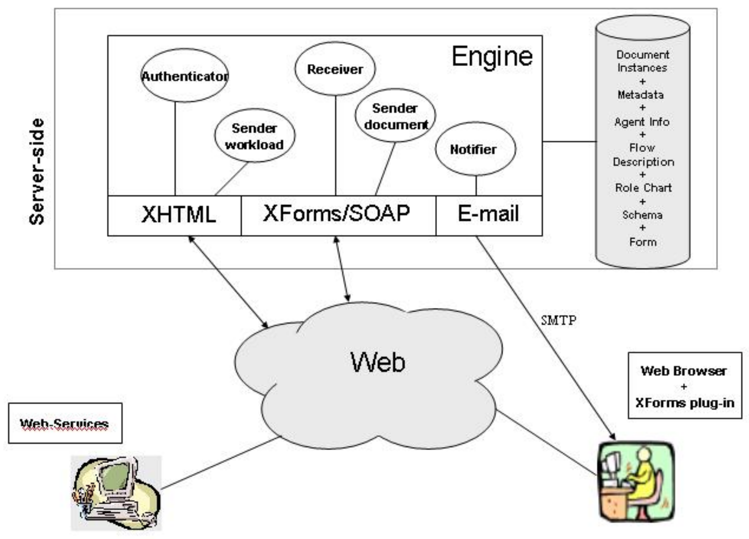

Server-side
Engine
Authenticator
Receiver
Sender workload
Sender document
Notifier
XHTML
XForms/SOAP
E-mail
Document Instances + Metadata + Agent Info + Flow Description + Role Chart + Schema + Form
SMTP
Web
Web-Services
Web Browser + XForms plug-in

Fig. 13. The overall system architecture

Each software agent is implemented as a web-service and the WSDL language is used to define its interface. The *Notifier* is used to alert an agent when some document needs to be processed in a short time (deadline) and optionally when a new document is inserted in his work list.

6 Conclusion

In this paper we have described a framework to define a workflow for collaborative document procedures. It is based upon a complete independence between workflow definition and engine, and supports the simultaneous processing of collaborative documents.
The benefits arising from the usage of this system include:

- interoperability/portability deriving from using XML technologies;
- concurrent documents workflows support;
- reduction of the cost of documents processes, through the e-documents processing and distribution;

XML is a suitable technology for representing not only data/documents but also to describe the document workflow logic. XFlowML is the XML application defined to describe a document workflow.
We have defined a model to describe DW based on three XML documents (schema, rolechart and xflow) that allows an easy description of many DW.
We have implemented a DW engine interpreting XFlowML documents, by using Cocoon, a very powerful middleware, to develop XML prototypes.
The DW engine implemented is heavily based on XML technologies (XSLT, XPath, XForms, SVG) and open-source tools (Cocoon, Tomcat, mySQL).

We have used Xforms technology to create dynamic user interfaces.
Our future work will focus on extension of the DW engine with new internal agents and development of a distributed framework based on mobile agents.

Thanks to Tatiana Bacci who made significant contributions.

7 References

1. A. Marchetti, S. Minutoli, P.Lazzareschi, and M.Martinelli. *A System for Managing Documents in a Step by Step Process*. In Proc. XML World Euro Edition, 26-28 March 2001, Amsterdam-Holland.
2. M. Tesconi, A. Marchetti, S. Minutoli, F. Ronzano, Web Forms T.R. IIT TR-09/2005 April 2005.
3. R. Krishnan, L. Munaga, and K. Karlapalem. *XDoC-WFMS: A Framework for Document Centric Workflow Management System.* In Proc. 20th International Conference on Conceptual Modeling – ER 2001, 27-30 Springer Verlag, November 2001, pp. 348-362, ISBN:3-540-44122-0.
4. L. Aversano, G. Canfora, A. De Lucia, P. Gallucci. *Integrating document and workflow management tools using XML and web technologies: a case study*. In Proc. Sixth European Conference on Software Maintenance and Reengineering, 11-13 March 2002, Budapest – Hungary, pp. 24-33.
5. P. Ciancarini, R. Tollksdorf, F. Zambonelli. *Coordination Middleware for XML-Centric Applications.* In Proc. 16th ACM Symposium on Applied Computing – SAC2000, 10-14 March 2002, Madrid, Spain.
6. G. Kappel, S. Rausch-Schott, S. Reich, W. Retschitzegger. *Hypermedia document and workflow management based on active object-oriented databases*. In Proc. of the Thirtieth Hawaii International Conference on System Sciences, Vol. 4, 7-10 Jan. 1997, pp. 377-386.
7. L. Baresi, F. Casati, S. Castano, M.G. Fugini, I. Mirbel, B. Pernici. *WIDE Workflow Development Methodology*. In Proc. of the International Joint Conference on Work activities Coordination and Collaboration, 1999, San Francisco, California, United States, pp. 19 – 28, ISBN: 1-58113-070-8.
8. F. Casati, M. G. Fugini, I. Mirbel, B. Pernici. WIRES: *A Methodology for Developing Workflow Applications*. In Requirements Engineering Journal, Springer Verlag Press., Vol. 7, Num. 2, 2002, pp. 73-106.
9. D. Georgakopoulos, H. Hornick and A. Sheth. *An Overview of Workflow Management: from Process Modelling to Workflow Automation Infrastructure*. In Distributed and Parallel Database Journal, Kluwer Academic Publishers Press., Vol. 3, Num. 2, April 1995, pp. 119-152.
10. A. Tripathi, T. Ahmed, V. Kakani, S. Jaman. *Implementing Distributed Workflow Systems from XML Specifications.* Department of Computer Science, University of Minnesota, May 2000. Available at http://www.cs.umn.edu/Ajanta/publications.html
11. R. Tolksdorf, Marc Stauch *Using XSL to Coordinate Workflows* Kommunikation in Verteilten Systemen 2001 127-138
12. R. Tolksdorf *Workspaces: A Web-Based Workflow Management System*IEEE Internet Computing September 2002 v.6 n.5 p.18-26