Tutorial Alfresco - jBPM: exemple d'implémentation d'un workflow avancé

par
Landry KOUAM (Centre de Compétence Alfresco/jBPM, Koossery Technology)
Baké Jc. BAKENEGHE (Software Coordinator, Koossery Technology)

Date de publication : 12 mai 2009

Dernière mise à jour :

Dans cet article, nous allons voir comment la plate-forme Alfresco combinée à la plate-forme jBoss BPM permet d'implémenter de façon aisée un exemple d'application qui combine à la fois les points suivants: la gestion électronique de documents, la gestion de contenu et les workflow BPM.

Dans le cadre des projets GED/BPM que ses clients lui confient, **Koossery Technology** s'appuie sur les standards suivants :

- **Alfresco** pour ce qui est de l'implémentation de la GED/ECM
- **jBoss BPM** pour ce qui est de l'implémentation des workflow BPM
- **jBoss BPEL** pour ce qui est de l'orchestration de processus
- et enfin l'ESB **Mule ESB** pour l'intégration de processus hétérogènes.

Cet article traite d'un exemple d'implémentation d'un workflow avancé.
Dans la partie I, nous présentons les frameworks (Alfresco, jBoss BPM, etc..) qui ont été utilisés dans ce tutorial.
Dans la partie II nous présentons un résumé des spécifications du workflow.
Dans la partie III nous parlons de la modélisation jBPM du workflow.
Dans la partie IV nous parlons de l'implémentation dans Alfresco.

I - Frameworks standards utilisés : Alfresco, JBoss BPM, Mule ESB

Les frameworks standards utilisés pour les développements au forfait GED/ECM/BPM chez **Koossery Technology** sont :

Eléments Constitutifs	Framework adopté par Koossery Tech'	Description
Implémentation de la GED et de l'ECM	Alfresco	Alfresco est une plate-forme pour la Gestion de Contenu d'Entreprise (ECM) - Gestion de Documents, Collaboration, Gestion des Archives/Enregistrements légaux, Gestion de Contenu Web et Gestion des Documents Numérisés
Moteur de Workflow	JBoss BPM	Orchestrateur des tâches des processus, gestion des instances des processus, persistance du contexte et de l'état du processus dans une base de données relationnelle.
Outil de pilotage et reporting (BAM)	Console Web de jBPM	Console d'administration et de monitoring des processus. Permet de faire le BAM.
Outil de modélisation des processus	Eclipse jPDL graphical process designer	Modélisation à l'aide d'une interface graphique des processus métier BPM de l'entreprise. Plugin Eclipse
Enterprise Service Bus	Mule ESB	Enterprise service bus permettant notamment de faire communiquer les processus dans le cadre d'une orchestration de processus

Figure 1 : Frameworks utilisés dans le Centre de Compétence Alfresco/jBPM de Koossery Tech'

Dans les paragraphes ci-dessous nous allons brièvement parcourir chacun de ces frameworks ou outils.

I-1 - Implémentation GED/ECM : Alfresco

Alfresco fournit des API Java permettant d'implémenter les fonctionnalités suivantes dans une application GED/ECM:
- Gestion de Contenu d'Entreprise (ECM), référentiel JSR170
- Gestion de Documents
- Collaboration
- Gestion des Archives/Enregistrements légaux
- Gestion du Documents numérisés

La plate-forme Alfresco est bâtie sur une architecture J2EE et s'appuie sur plusieurs standards comme JSF MyFaces, Lucene, Ajax, JCR, Spring, Hibernate etc..

I-2 - Moteur de workflow BPM: jBoss BPM

BPM (Business Process Management) est l'approche consistant à modéliser les processus métiers de l'entreprise. Le but étant d'aboutir à une meilleure vue globale de l'ensemble des processus et de leurs interactions afin d'être en mesure de les optimiser et de les automatiser.

Koossery Technology s'appuie sur la plate-forme jBoss BPM pour :
- la modélisation des processus (utilisation de l'outil jBoss BPM jPDL)
- l'orchestration des tâches à l'intérieur d'un processus
- l'orchestration de processus (jBoss BPEL)

Un des aspects important du BPM est le monitoring des processus (BAM Business Activity Monitoring).
Il s'agit d'analyser les logs d'exécution des process afin d'obtenir des statistiques d'information à propos du processus métier (exemple : combien de temps en moyenne met chaque tâche du process, etc..).
La console web jBPM est un outil standard pour faire du BAM. C'est une console d'administration et de monitoring qui permet d'inspecter et de manipuler le runtime des instances des processus jBPM.

I-3 - Intégration de processus hétérogènes : Mule ESB

Koossery Technology s'appuie sur l'esb Mule ESB pour intégrer les processus hétérogènes, et ce dans le cadre d'une orchestration de processus.

L'orchestration de processus fera l'objet d'une publication ultérieure.

- Mise en oeuvre Alfresco/jBPM : implémentation d'un workflow d'approbation des dossiers de crédit

Dans ce chapitre nous allons voir comment s'appuyer sur Alfresco et jBPM pour implémenter un workflow avancé d'approbation des dossiers de crédit.
Dans les paragraphes qui suivent, nous allons tour à tour présenter :
- un résumé des spécifications du workflow
- la modélisation BPM du processus métier qui en découle
- et enfin les opérations mises en oeuvre pour son implémentation

II - Résumé des spécifications du workflow d'approbation des dossiers de crédit

Un organisme de prêt à la consommation souhaite mettre en place un Workflow d'approbation de crédit qui combine l'étude du dossier de crédit avec son passage par plusieurs étapes, de l'approbation jusqu'à son archivage.
Ci-dessous les différentes étapes de l'approbation:

1. Le cycle commence par la réception de la demande de crédit du client.
Cette demande peut être faite par courrier électronique ou par courrier papier.

2. La demande est transmise au service du crédit qui ensuite quotte le dossier à l'agent chargé du crédit.

3. L'agent chargé du crédit rentre en contact avec le client, lui remet un formulaire (fiche interne de demande de crédit) qui contient au moins une trentaine (30) de questions (création d'un type de crédit, conditions d'attribution, remboursement, garanties etc..).

4. Suite aux réponses du client, l'agent de crédit fait un 1er filtrage : à l'issue de ce filtrage, le dossier de crédit est soit rejeté, soit présenté au COC (Comité d'Octroi de Crédit).
Dans la suite on suppose que le dossier est présenté au Comité d'Octroi de Crédit.

5. Le Comité d'Octroi de Crédit reçoit le dossier et l'étudie.
Un avis est donné au dossier: le dossier est soit approuvé, soit rejeté.
Lorsque le dossier est approuvé un rapport de synthèse est élaboré et transmis simultanément au service juridique et à la cellule en charge de formaliser les décaissements de fonds.
Dans la suite on suppose que le dossier est approuvé et qu'un rapport de synthèse est établi.

6. Le service juridique reçoit le rapport de synthèse et sur la base de ce rapport, il s'occupe de la préparation du contrat stipulant les engagements des deux partis avec des précisions d'autres clauses clés (pénalités, garanties, etc..).
S'il y a nantissement sur les garantes la procédure recommence (on retourne à l'étape 5).
La cellule en charge de formaliser les décaissements de fonds quant à elle commence à formaliser le décaissement des fonds sur la base du rapport reçu.

7. Une fois le contrat élaboré et le traitement des garanties achevé, l'organisme de Crédit contacte le client pour signature dudit contrat.

8. Le dossier est alors transmis à la Direction Administrative et Financière pour le déblocage des fonds. (signature des chèques, des ordres de virements, etc..).

9. Le déblocage du crédit peut être fait soit par chèque à l'ordre du client bénéficiaire du crédit, soit par virement du compte de l'organisme de crédit au compte du client bénéficiaire du crédit.

Remarque :
- A chaque passage du dossier dans une étape, la date et la signature doivent être portées sur le dossier pour mesures de sécurité (qui a fait quoi et quand).

III - Modélisation (workflow) jBPM du processus métier

Il s'agit de réaliser le diagramme du Workflow, modéliser les tâches et leurs assignations.
On ressortira aussi la cartographie fonctionnelle associée au processus.

III-1 - Diagramme du Workflow

La figure ci-dessous est le diagramme (Process design) du Workflow du cas d'étude approbation de crédit.

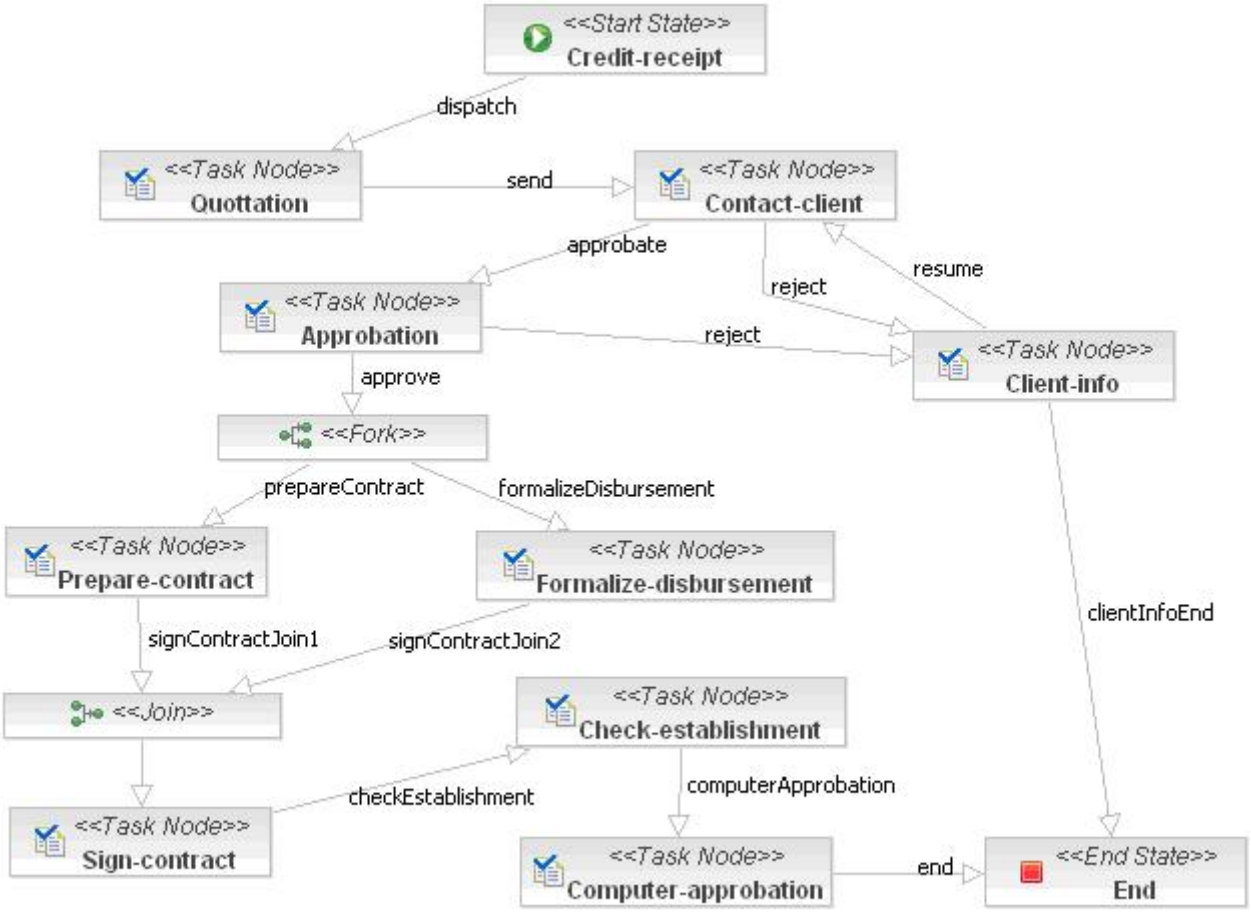


Figure 2 : Process design du Workflow d'approbation du crédit

Dans la figure, 2 ci-dessus, on distingue plusieurs noeuds d'exécution du processus BPM. Nous allons les parcourir :

III-1-1 - Node "Credit-receipt"

Il représente le noeud de réception de la demande du crédit du client.
Ce noeud possède une unique tâche réalisée par un agent du courrier. Cette tâche consiste à transmettre la demande au responsable du service du crédit.
Dans le cas où l'agent du courrier réceptionne la demande du client sous format papier, l'agent la numérise à travers une chaîne de capture. La demande numérisée est directement prise en compte dans le système de GED. L'agent du courrier initialise alors le Workflow sur la demande numérisée et réalise la présente tâche.

Le noeud "Credit-receipt" est un noeud de type "Start State", c'est à dire qu'il représente le début du Workflow.
L'entrée de ce noeud dans le fichier jPDL du Workflow est la suivante :

Figure 3: Node 'Credit-receipt' dans le fichier jPDL

```
<start-state name="Credit-receipt">
 <task name="ac:dispatch-application" swimlane="initiator">
 </task>
 <transition to="Quottation" name="dispatch"></transition>
 <event type="node-leave">
          <action class="org.alfresco.repo.workflow.jbpm.AlfrescoJavaScript">
             <script>
                <variable name="ac_cocPool" access="write"/>
```

Figure 3: Node 'Credit-receipt' dans le fichier jPDL

```
                <expression>people.getGroup('GROUP_coc');</expression>
             </script>
          </action>
          <action class="org.alfresco.repo.workflow.jbpm.AlfrescoJavaScript">
             <script>
                <variable name="ac_codirPool" access="write"/>
                <expression>people.getGroup('GROUP_codir');</expression>
             </script>
          </action>
          <action class="org.alfresco.repo.workflow.jbpm.AlfrescoJavaScript">
             <script>
                <variable name="ac_dafPool" access="write"/>
                <expression>people.getGroup('GROUP_daf');</expression>
             </script>
          </action>
          <action class="org.alfresco.repo.workflow.jbpm.AlfrescoJavaScript">
             <script>
                <variable name="ac_directionPool" access="write"/>
                <expression>people.getGroup('GROUP_direction');</expression>
             </script>
          </action>
          <action class="org.alfresco.repo.workflow.jbpm.AlfrescoJavaScript">
             <script>
                <variable name="ac_financialServicePool" access="write"/>
                <expression>people.getGroup('GROUP_financial-service');</expression>
             </script>
          </action>
          <action class="org.alfresco.repo.workflow.jbpm.AlfrescoJavaScript">
             <script>
                <variable name="ac_judicialServicePool" access="write"/>
                <expression>people.getGroup('GROUP_judicial-service');</expression>
             </script>
          </action>
   </event>

</start-state>
```

Le moteur jBPM nous permet d'exécuter certaines actions suite à des évènements sur les noeuds comme l'entrée ou la sortie d'un noeud.
Dans le cas de la figure 3, une série d'actions seront exécutées dès que le jeton d'exécution quitte le noeud (<event type="node-leave">). La principale action qui doit être exécutée dès la sortie du noeud est l'Action Handler d'Alfresco AlfrescoJavaScript. Cette action permet l'exécution du code JavaScript (côté serveur grâce au moteur Rhino) présent dans le fichier jPDL du Workflow.

Nous configurons aussi certaines variables globales :
- ac_cocPool: représente le pool d'acteurs membres du comité d'octroi de crédit;
- ac_codirPool: représente le pool d'acteurs membres du comité de direction;
- ac_dafPool: représente le pool d'acteurs membres de la direction des affaires financières ;
- ac_directionPool: représente le pool d'acteur membres de la direction;
- ac_financialServicePool: représente le pool d'acteurs du service des finances;
- ac_judicialServicePool: représente le pool d'acteurs membres du service juridique.

Dans ce noeud, l'agent du courrier doit envoyer la demande de crédit au service du crédit : cet envoi est matérialisé par la transition nommée "dispatch".

III-1-2 - Node "Quottation"

Noeud de cotation du dossier du crédit à un agent du crédit ou au gestionnaire du client.
Ce noeud comporte une tâche réalisée par le responsable du crédit. La tâche consiste à choisir l' agent du crédit qui sera en charge du dossier et de le lui envoyer.
La description de ce noeud dans le fichier jPDL du Workflow est la suivante :

Figure 4: Node 'Quottation' dans le fichier jPDL du workflow

```
<task-node name="Quottation">
 <task name="ac:quottation" swimlane="loan-service">
  <event type="task-create">
   <!-- Set the due date and the priority of the task -->
   <script>
      if (bpm_dueDate != void) taskInstance.dueDate = bpm_dueDate;
      if (bpm_priority != void) taskInstance.priority = bpm_priority;
   </script>
  </event>
  <event type="task-assign">
     <!-- Before the task is assign, send an email to the initiator -->
     <action class="org.alfresco.repo.workflow.jbpm.AlfrescoJavaScript">
   <script>
     if(ac_notifyMe)
     {
       var mail = actions.create("mail");
       mail.parameters.to = initiator.properties["cm:email"];
       mail.parameters.subject = "Loan approbation process ";
       mail.parameters.from = bpm_assignee.properties["cm:email"];
       mail.parameters.text = "Loan application send successfully.";
       mail.execute(bpm_package);
    }
   </script>
     </action>
  </event>
 </task>
 <transition to="Contact-client" name="send"></transition>
</task-node>
```

Dans le noeud de la figure 4, on positionne une date d'échéance (bpm_dueDate) et une priorité par défaut (bpm_priority) dès l'instanciation de la tâche (<event type="task-create">).

Dès que le Responsable de Crédit assigne un agent de crédit pour l'étude de dossier (<event type="task-assign">), une action est exécutée : grâce à JavaScript, un mail de notification va être envoyé au Responsable du crédit pour le confirmer l'assignation de la tâche à l'agent du crédit à qui il a quotté le dossier.
La transition nommée "send" permet de déplacer l'exécution du Workflow au noeud "Contact-client".

III-1-3 - Noeud Contact-client

Noeud de contact avec le client. Ce noeud possède une tâche réalisée par l'agent en charge crédit. Celui-ci est invité à travers cette tâche à contacter le client. L'agent pose un certain nombre de questions au client. Sur la base des réponses du client, il crée (toujours dans cette tâche) une fiche interne de la demande de crédit du client. Il peut soit rejeter la demande si son filtrage est négatif ou envoyer la demande pour approbation au comité d'octroi. Ce choix est matérialisé par les deux transitions nommées "approbate" et "reject".
La description de ce noeud dans le fichier jPDL du Workflow est la suivante :

Figure 5: Noeud 'Contact-client' dans le fichier jPDL du workflow

```
<task-node name="Contact-client">
  <description>
   The agent in charge of credit contacts the customer, gives him a form
   (form of internal customer application) which contains at least thirty (30) questions.
   Following the customer's responses, the agent may make a 1st screening
   before presenting the customer' case to the COC.
   At each stage of transition, date and signature must be entered on the
    record for security measures (who did what and when).
  </description>
  <task name="ac:contact-client" swimlane="loan-agent">
   <event type="task-end">
       <!-- Remember agent actor -->
     <action class="org.alfresco.repo.workflow.jbpm.AlfrescoJavaScript">
    <script>
                  <variable name="ac_agent-actor" access="write" />
```

Figure 5: Noeud 'Contact-client' dans le fichier jPDL du workflow

```
                    <expression>
                           if (taskInstance.actorId != null)
                              taskInstance.actorId;
                           else
                              person;
                        </expression>
    </script>
       </action>
   </event>
  </task>
  <transition to="Client-info" name="reject"></transition>
  <transition to="Approbation" name="approbate"></transition>
 </task-node>
```

III-1-4 - Node "Approbation"

Noeud d'approbation de la demande par le comité d'octroi de crédit. Ce noeud est composé d'une tâche réalisée par un pool d'acteurs (les membres du groupe représenté par la variable ac_cocPool).
Tous les membres de ce groupe reçoivent la même tâche. L'un d'eux doit s'approprier la tâche afin de modifier le système en enregistrant la décision du comité.
La décision du comité sur la demande peut être soit le rejet de la demande, soit l'approbation. Ce choix est matérialisé par les deux transitions nommées "reject" et "approve".
La description de ce noeud dans le fichier jPDL du Workflow est la suivante:

Figure 6: Noeud 'Approbation' dans le fichier jPDL du workflow

```
<task-node name="Approbation">
     <description>
  Each member of the COC receives an approval tack. One of the members
  must take the task to inform the state of the application decided by the committee.
  After reviewing, a judgment is given: application approved or
  rejected. (reason of rejection).
 </description>

        <task name="ac:approbation-coc" swimlane="coc">
   <!-- Remember the member who own the task -->
           <event type="task-end">
              <action class="org.alfresco.repo.workflow.jbpm.AlfrescoJavaScript">
                 <script>
                    <variable name="ac_coc-member" access="write"/>
                    <expression>
                       if (taskInstance.actorId != null)
                          people.getPerson(taskInstance.actorId);
                       else
                          person;
                    </expression>
                 </script>
              </action>
           </event>

       </task>
 <transition to="Contract-preparation" name="approve"></transition>
 <transition to="Client-info" name="reject"></transition>
</task-node>
```

III-1-5 - Node "Contract-preparation"

Dans un Workflow jPDL, un Fork est utilisé lorsqu'on veut réaliser des tâches en simultané. La continuation de l'exécution du Workflow à un autre noeud ne se fera que si toutes les tâches en simultané sont achevées.
La description dans le fichier jPDL est la suivante :

Figure 7: Noeud 'Contract-preparation' dans le fichier jPDL du workflow

Figure 7: Noeud 'Contract-preparation' dans le fichier jPDL du workflow

```
<fork name="Contract-preparation">
   <description>
  The report is forwarded to the judicial department that handles
  the preparation of the contract stipulating the commitments
  of both parties with details on other key clauses in case the appropriation is
  approved. (Penalties, guarantees, etc ...).
   </description>
   <transition to="Prepare-contract" name="prepareContract"></transition>
   <transition to="Formalize-disbursement"name="formalizeDisbursement"></transition>
</fork>
```

Dans la figure 7, à travers les deux transitions nommées "prepareContract" et "formalizeDisbursement ", le moteur du Workflow va lancer simultanément les tâches des noeuds "Prepare-contract" et "Formalize-disbursement" qui permettront respectivement au service juridique de préparer le contrat et à la cellule en charge de formaliser le décaissement de formaliser ledit décaissement.

III-1-6 - Node "Prepare-contract"

Noeud de préparation du contrat. Comporte une tâche réalisée par le service juridique. Le contrat est crée et ajouté au ressources du Workflow.
La description de ce noeud dans le fichier jPDL du Workflow est la suivante :

Figure 8: Noeud 'Prepare-contract' dans le fichier jPDL du workflow

```
<task-node name="Prepare-contract">
 <task name="ac:prepare-contract" swimlane="judicial-service"/>
 <transition to="Sign-contract-join" name="signContractJoin1"></transition>
</task-node>
```

III-1-7 - Node "Formalize-disbursement"

Noeud de formalisation du décaissement. Comporte une tâche réalisée par le service en charge de formaliser les décaissements.
La description de ce noeud dans le fichier jPDL du Workflow est la suivante :

Figure 9: Noeud 'Formalize-disbursement' dans le fichier jPDL du workflow

```
<task-node name="Formalize-disbursement">
 <task name="ac:formalize-disbursement" swimlane="financial-service"/>
 <transition to="Sign-contract-join" name="signContractJoin2"></transition>
</task-node>
```

Une fois que ces deux tâches sont réalisées, l'exécution du Workflow peut continuer au noeud Join "Sign-contract-join".

III-1-8 - Node "Sign-contract-join"

Dans un Workflow jPDL, un noeud Join permet de marquer un point d'attente de l'exécution de plusieurs tâches en parallèle. Le moteur attendra que tous les noeuds de même parent arrivant à ce noeud soient terminés. Ce noeud comporte une unique transition par laquelle l'exécution va continuer.
La description dans le fichier jPDL est la suivante :

Figure 10: Noeud 'Sign-contract-join' dans le fichier jPDL du workflow

```
<join name="Sign-contract-join">
 <transition to="Sign-contract"></transition>
</join>
```

Le noeud de signature du contrat comporte une tâche réalisée par le comité de direction. Un membre du comité de direction invite le client à signer le contrat. L'exécution est ensuite déplacée sur le noeud d'établissement des chèques.

III-1-9 - Node "Check-etablishment"

Noeud d'établissement des chèques. Ce noeud comporte une tâche réalisée par le directeur des affaires financières. Les différents chèques établis sont scannés et importés comme ressources dans l'instance du Workflow (Le paragraphe IV-4-1 traite de la chaîne de dématérialisation de notre tutorial).
L'exécution est ensuite déplacée vers le noeud d'approbation informatique.

Figure 11: Noeud 'Check-establishment' dans le fichier jPDL du workflow

```
<task-node name="Check-establishment">
 <description>
  Documents are forwarded to the Administrative and Finance directions
  for the release of funds (signing checks, Orders of transfers, etc ...).
  The Chief of Financial Department validates the amount to be granted,
  then he sends one or more checks.
  The Direction receives all documents and validates them.
 </description>
 <task name="ac:check-establishment-signature" swimlane="daf"></task>
 <transition to="Computer-approbation" name="computerApprobation"></transition>
</task-node>
```

III-1-10 - Node "Computer-approbation"

Noeud d'approbation informatique. Comporte une tâche d'approbation informatique réalisée par un membre du comité d'octroi du crédit. Celui-ci doit faire les dernières opérations afin de clore le dossier du client.
La description de ce noeud dans le fichier jPDL du Workflow est la suivante :

Figure 12: Noeud 'Computer-approbation' dans le fichier jPDL du workflow

```
<task-node name="Computer-approbation">
 <task name="ac:computer-approbation" swimlane="coc"></task>
 <transition to="End" name="end"></transition>
</task-node>
```

III-1-11 - Node "Client-info"

Noeud d'information du client. Comporte une tâche réalisée par l'agent en charge du crédit. Cette tâche consiste à informer le client sur l'état de sa demande. L'information peut être par téléphone ou email.
La description de ce noeud dans le fichier jPDL du Workflow est la suivante :

Figure 13: Noeud 'Client-info' dans le fichier jPDL du workflow

```
<task-node name="Client-info">
 <description>
  Applications not approved are returned to  the agent who inform the
  customer. In case of loan rejected, a letter of regret is prepared.
  The agent makes a consultation and informs the customer about the reason
  for the rejection of his request.
 </description>
 <task name="ac:inform-client" swimlane="loan-agent"></task>
 <transition to="Contact-client" name="resume"></transition>
 <transition to="End" name="clientInfoEnd"></transition>
</task-node>
```

III-2 - Assignations des tâches

Assigner une tâche c'est designer l'acteur ou le pool d'acteurs qui réaliseront cette tâche. Toutes les tâches sont assignées. Nous allons voir comment les tâches du cas d'étude approbation du crédit sont assignées.

Dans le langage jPDL, l'assignation des tâches se fait à partir des rôles déclarés à travers des éléments swimlane:

Figure 14 : déclaration du rôle 'initiator'

```
<swimlane name="initiator"/>
```

La déclaration ci-dessus (figure 14) permet de créer un rôle "initiator" qui par défaut sera joué par l'initiateur du Workflow c'est à dire l'agent du courrier. Ainsi une tâche assignée à "initiator" sera envoyée à l'initiateur du Workflow. Les rôles suivants ont été créés:

- Rôle 'initiator'
Déclaration dans le fichier jPDL :

Figure 14 : déclaration du rôle 'initiator'

```
<swimlane name="initiator"/>
```

- Rôle 'loan-service'
Le responsable du service crédit est emmené à jouer ce rôle.
Déclaration dans le fichier jPDL :

Figure 15 : déclaration du rôle 'loan-service'

```
<swimlane name="loan-service">
 <assignment class="org.alfresco.repo.workflow.jbpm.AlfrescoAssignment">
  <actor>#{bpm_assignee.properties['cm:userName']}</actor>
 </assignment>
</swimlane>
```

Une expression EL est utilisée pour injecter l'acteur jouant ce rôle. La variable globale "bpm_assignee" contient une valeur qui représente l'acteur à qui la tâche est assignée.
On pourrait se demander où est le responsable du service crédit dans "bpm_assignee" dans le schéma (figure 15). En fait, lors du passage du noeud "Credit-receipt" au noeud "Quottation", la variable de scope processus "bpm_assignee" existe déjà et à pour valeur l'utilisateur responsable du service du crédit sélectionné par l'agent du courrier au noeud "Credit-receipt".
Cette variable est créée à travers le content model du task "ac:dispatch-application" du noeud "Credit-receipt" (cf. paragraphe IV-2-1):

```
<mandatory-aspects>
 <aspect>bpm:assignee</aspect>
</mandatory-aspects>
```

Cette aspect permet d'ajouter une propriété "bpm_assignee" au model de la tâche et de ce fait le Client Web génèrera un composant UI dans la page de la tâche pour la sélection du user à attribuer à la propriété. Le comportement ajouté par l'aspect créée la variable de scope processus nommé "bpm_assignee" et la rend accessible à tout le contexte du processus. Ceci permet ainsi d'utiliser la variable dans le fichier jPDL pour l'assignation du loan-service.

- Rôle 'loan-agent'
L'agent de crédit qui va étudier le dossier est emmené à jouer ce rôle.
Déclaration dans le fichier jPDL :

Figure 16 : déclaration du rôle 'loan-agent'

```
<swimlane name="loan-agent">
 <assignment class="org.alfresco.repo.workflow.jbpm.AlfrescoAssignment">
  <actor>#{bpm_assignee.properties['cm:userName']}</actor>
 </assignment>
</swimlane>
```

La valeur de la variable globale "bpm_assignee" a changé et contient cette fois l'acteur agent du crédit.

- Rôle 'coc'
Déclaration dans le fichier jPDL :

Figure 17 : déclaration du pool d'acteur 'COC'

```
<swimlane name="coc">
        <assignment class="org.alfresco.repo.workflow.jbpm.AlfrescoAssignment">
            <pooledactors>#{ac_cocPool}</pooledactors>
        </assignment>
</swimlane>
```

Cette fois (figure 17) ça n'est plus un acteur unique qui doit jouer un rôle mais un pool d'acteurs, c'est à dire un ensemble d'acteurs qui recevront la même tâche et devront travailler en collaboration pour la réaliser.
La variable "ac_cocPool" est une variable globale qui a été initialisée dans le noeud de début du Workflow. L'injection de l'ensemble des acteurs se fait avec une expression EL.

- Rôle 'codir'
Déclaration dans le fichier jPDL :

Figure 18 : déclaration du pool d'acteur 'CODIR'

```
<swimlane name="codir">
        <assignment class="org.alfresco.repo.workflow.jbpm.AlfrescoAssignment">
            <pooledactors>#{ac_codirPool}</pooledactors>
        </assignment>
</swimlane>
```

- Rôle 'daf'
Directeur des affaires financières.
Déclaration dans le fichier jPDL :

Figure 19 : déclaration du pool d'acteur 'DAF'

```
<swimlane name="daf">
        <assignment class="org.alfresco.repo.workflow.jbpm.AlfrescoAssignment">
            <pooledactors>#{ac_dafPool}</pooledactors>
        </assignment>
</swimlane>
```

- Rôle 'direction'
Déclaration dans le fichier jPDL :

Figure 20 : déclaration du pool d'acteur 'direction

```
<swimlane name="direction">
        <assignment class="org.alfresco.repo.workflow.jbpm.AlfrescoAssignment">
            <pooledactors>#{ac_directionPool}</pooledactors>
        </assignment>
</swimlane>
```

- Rôle 'financial-service'

Service financier.
Déclaration dans le fichier jPDL :

Figure 21: déclaration du pool d'acteur 'financial-service'

```
<swimlane name="financial-service">
       <assignment class="org.alfresco.repo.workflow.jbpm.AlfrescoAssignment">
           <pooledactors>#{ac_financialServicePool}</pooledactors>
       </assignment>
</swimlane>
```

- Rôle 'judicial-service'
Service juridique.
Déclaration dans le fichier jPDL :

Figure 22 : déclaration du pool d'acteur 'judicial-service'

```
<swimlane name="judicial-service">
       <assignment class="org.alfresco.repo.workflow.jbpm.AlfrescoAssignment">
           <pooledactors>#{ac_judicialServicePool}</pooledactors>
       </assignment>
</swimlane>
```

IV - Implémentation de la solution dans Alfresco

La solution à été implémentée avec Alfresco. Alfresco fournit un mécanisme conceptuel et des APIs pour la réalisation du BPM sous forme d'une application web J2EE.
Le diagramme d'activités préconisé par le **Centre de Compétence Alfresco/jBPM** de **Koossery Technology** pour l'implémentation d'un workflow se résume au tableau ci-dessous:

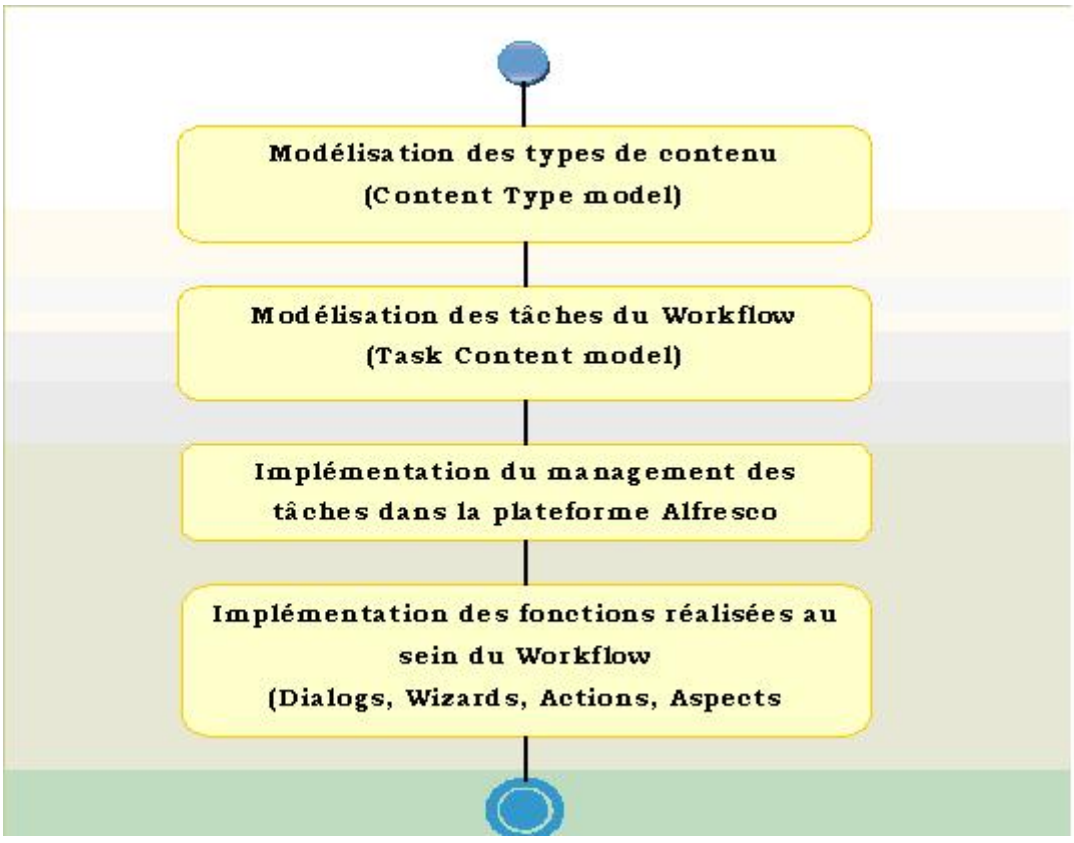
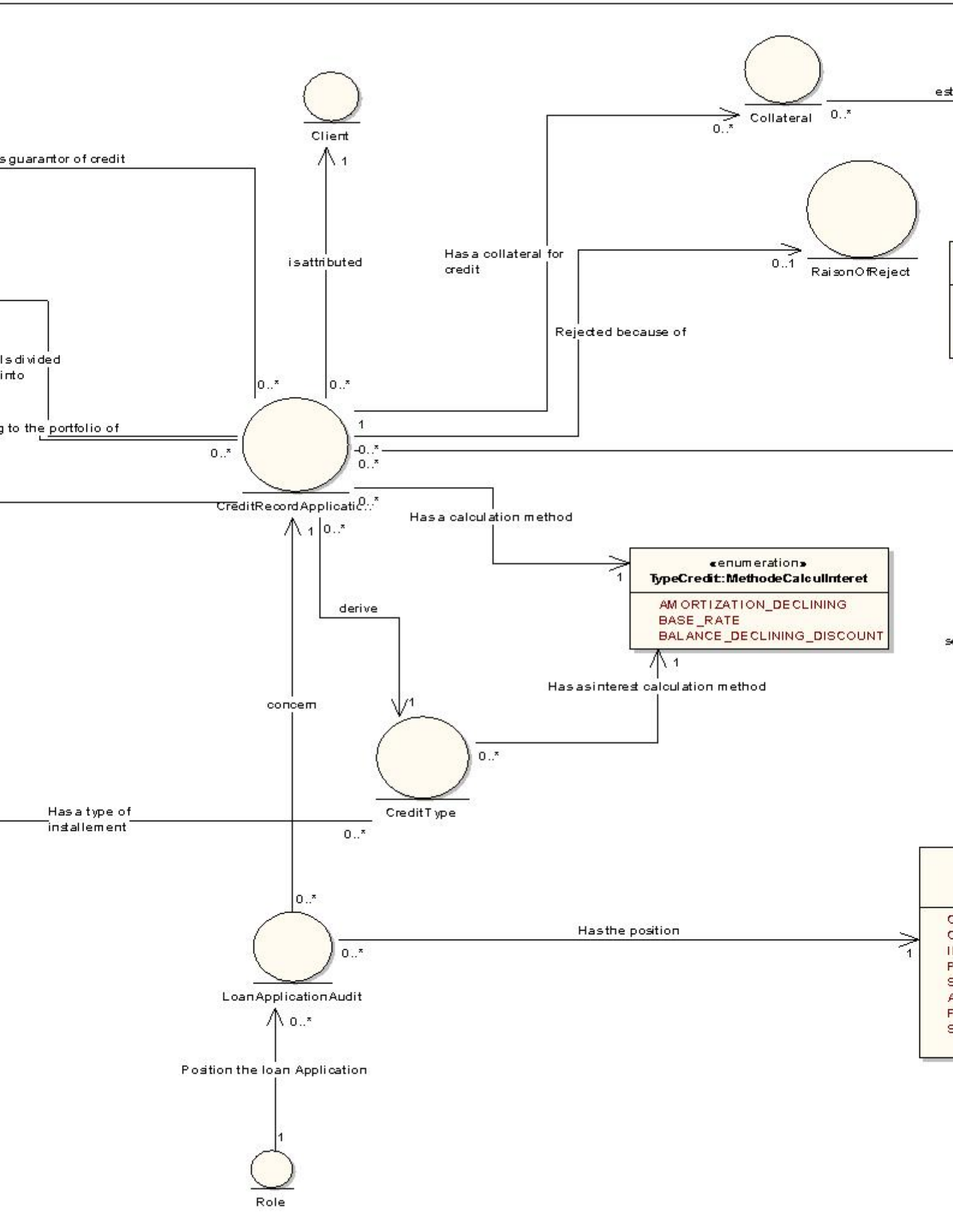


Figure 23 : Diagramme d'activités de l'implémentation d'un Workflow préconisé par le Centre de Compétence Alfresco/jBPM de Koossery Technology

Nous allons appliquer ce diagramme d'activités sur le cas d'étude du Workflow d'approbation du crédit.

IV-1 - Modélisation des types de contenu ('Content Type model')

Ci-dessous le schéma du modèle d'entité de l'application :


Client
1
is attributed
0..*
0..*
s guarantor of credit
Collateral
0..*
0..*
est
Has a collateral for credit
0..1
RaisonOfReject
Rejected because of
ls divided into
g to the portfolio of
0..*
1
0..*
0..*
0..*
CreditRecordApplicati...
1
0..*
Has a calculation method
1
«enumeration»
TypeCredit::MethodeCalculInteret
AMORTIZATION_DECLINING
BASE_RATE
BALANCE_DECLINING_DISCOUNT
1
Has asinterest calculation method
derive
concern
1
0..*
CreditType
Has a type of installement
0..*
0..*
Hasthe position
0..*
1
LoanApplicationAudit
0..*
Position the loan Application
1
Role

Le content type model de chaque entité du schéma ci-dessus (figure 24) doit être écrit à l'aide du dictionnaire de données Alfresco. Le paragraphe IV-1-1 montre le content model de l'entité "creditRecordApplication" (Fiche de demande de crédit).

IV-1-1 - Exemple de modélisation (Content Type model) de la Fiche de demande de crédit

Figure 25 : Content Model de la fiche de demande de crédit

```
<!-- Credit Application Record -->
  <type name="creditApprobation:creditRecordApplication">
   <title>Credit application record</title>
   <parent>cm:folder</parent>
   <properties>
    <property name="creditApprobation:creditNumber">
     <title>Credit number</title>
     <type>d:text</type>
     <mandatory>true</mandatory>
    </property>
    <property name="creditApprobation:creditProduct">
     <title>Credit product</title>
     <type>d:text</type>
     <mandatory>true</mandatory>
    </property>
    <property name="creditApprobation:applicationDate">
     <title>Application date</title>
     <type>d:date</type>
     <mandatory>true</mandatory>
    </property>
    <property name="creditApprobation:amount">
     <title>Amount</title>
     <type>d:float</type>
     <mandatory>true</mandatory>
    </property>
    <property name="creditApprobation:annualInterestRate">
     <title>Annual interest rate</title>
     <type>d:float</type>
     <mandatory>true</mandatory>
    </property>
    <property name="creditApprobation:nbOfInstallmentPayment">
     <title>Number of Installment Payment</title>
     <type>d:long</type>
     <mandatory>true</mandatory>
    </property>
    <property name="creditApprobation:installmentType">
     <title>Installment type</title>
     <type>d:text</type>
     <mandatory>true</mandatory>
    </property>
    <property name="creditApprobation:interestCalculationMethod">
     <title>Interest Calculation Method</title>
     <type>d:text</type>
     <mandatory>true</mandatory>
    </property>
    <property name="creditApprobation:lastMaturityCapital">
     <title>Last maturity capital</title>
     <type>d:float</type>
     <mandatory>true</mandatory>
    </property>
    <property name="creditApprobation:creditLine">
     <title>Credit line</title>
     <type>d:text</type>
     <mandatory>true</mandatory>
    </property>
    <property name="creditApprobation:agent">
     <title>Agent</title>
     <type>d:text</type>
     <mandatory>true</mandatory>
    </property>
    <property name="creditApprobation:interestCalculationByDay">
     <title>interest calculation by day</title>
```

Figure 25 : Content Model de la fiche de demande de crédit

```
      <type>d:boolean</type>
      <mandatory>false</mandatory>
     </property>
     <property name="creditApprobation:agreedPrepaymentOfInterest">
      <title>Agreed sum for prepayment of interest</title>
      <type>d:boolean</type>
      <mandatory>false</mandatory>
     </property>
     <property name="creditApprobation:deductInterestAtOutlay">
      <title>Deduct interest at outlay</title>
      <type>d:boolean</type>
      <mandatory>false</mandatory>
     </property>
     <property name="creditApprobation:interestDuringTheGracePeriod">
      <title>Interest during the grace period</title>
      <type>d:boolean</type>
      <mandatory>false</mandatory>
     </property>
     <property name="creditApprobation:stagger">
      <title>To stagger</title>
      <type>d:boolean</type>
      <mandatory>false</mandatory>
     </property>
    </properties>
    <associations>
     <child-association name="creditApprobation:installments">
      <target>
       <class>creditApprobation:installment</class>
       <mandatory>false</mandatory>
       <many>true</many>
      </target>
     </child-association>
     <child-association name="creditApprobation:guarantors">
      <target>
       <class>creditApprobation:guarantor</class>
       <mandatory>false</mandatory>
       <many>true</many>
      </target>
     </child-association>
     <child-association name="creditApprobation:collaterals">
      <target>
       <class>creditApprobation:collateral</class>
       <mandatory>false</mandatory>
       <many>true</many>
      </target>
     </child-association>
     <child-association name="creditApprobation:creditAppClient">
      <target>
       <class>creditApprobation:client</class>
       <mandatory>false</mandatory>
       <many>false</many>
      </target>
     </child-association>

    </associations>
   </type>
```

IV-2 - Modélisation des tâches du Workflow (Task Content Model)

Au paragraphe III-1 nous avons vu le design jBPM du processus.

Chaque tâche ou noeud de ce BPM design sera modélisé avec le dictionnaire de données Alfresco.

Le paragraphe IV-2-1 montre la modélisation de la tâche "dispatch application" réalisée par l'agent du courrier et consistant à envoyer la demande de crédit numérisée du client au responsable du crédit. Ce modèle est utilisé par le Client Web de la plateforme Alfresco pour la construction de la vue de présentation de la tâche.

IV-2-1 - Exemple de modélisation (Task Content Model) de la tâche "dispatch-application"

Les tâches sont définies dans le modèle comme type de contenu: en effet les entités tâches sont manipulées dans le JCR exactement comme tout autre type de contenu.

Figure 26 : Content model de la tâche 'dispatch-application'

```
<type name="ac:dispatch-application">
 <parent>bpm:startTask</parent>

 <properties>
  <property name="ac:notifyMe">
   <type>d:boolean</type>
   <default>false</default>
  </property>
 </properties>

 <mandatory-aspects>
  <aspect>bpm:assignee</aspect>
 </mandatory-aspects>

 <!-- Custom aspects may be added to collect any arbitrary
  number of people / groups. -->
</type>
```

Comme on peut le voir (figure 26), ce modèle hérite du modèle "bpm:startTask" qui est le modèle de base des tâches d'initiation des worklflows BPM.
Cette tâche est la première tâche qui doit être réalisée lors que le Workflow est initialisé.
Le modèle défini une propriété ("notifyMe") de type booléen. Cette propriété sera affichée à l'interface utilisateur sous forme de case à cocher et permettra de notifier par email l'agent du courrier de l'envoi de la demande de crédit au responsable de crédit.

IV-3 - Implémentation du management des tâches dans la plateforme Alfresco

Une fois le modèle des tâches réalisé, il faut effectuer un certain nombre d'opérations afin que les tâches puissent être accessibles dans le tableau de bord des utilisateurs. Ces opérations sont dans l'ordre :
- configuration du client web pour l'affichage des tâches
- réalisation des dashlet relatives au tableau de bord. Un dashlet est un composant UI Alfresco qui permet d'afficher des informations dans une zone du tableau de bord des utilisateurs

IV-3-1 - Configuration du Client Web pour l'affichage des tâches

La configuration du client web (l'application Web de la plateforme Alfresco) permet de préciser comment vont être affichés les composants UI pour la saisie des données. La configuration du client web se fait dans le fichier "web-client-config-custom.xml". La configuration du client web pour l'affichage de la vue de la tache de type "dispatch-application" est la suivante :

Figure 27 : Configuration du Client Web de la tâche ac:dispatch-application

```
<config evaluator="node-type" condition="ac:dispatch-application" replace="true">
 <property-sheet>
  <separator name="sep1" display-label-id="general"
      component-generator="HeaderSeparatorGenerator" />
  <show-property name="bpm:description" component-generator="TextAreaGenerator" />
  <show-property name="bpm:priority" />
  <show-property name="bpm:dueDate" />
  <show-property name="ac:notifyMe" />
  <separator name="sep2" display-label-id="responsable_service_credit"
      component-generator="HeaderSeparatorGenerator" />
  <show-association name="bpm:assignee"/>
 </property-sheet>
```

Figure 27 : Configuration du Client Web de la tâche ac:dispatch-application

```
</config>
```

IV-3-2 - Internationalisation I18N

L'internationalisation permet d'afficher la vue de la tâche dans la langue du l'utilisateur. La configuration de l'internationalisation relative à la tâche consiste à définir le bundle I18N des propriétés de la tâche définies dans le content model. En prenant l'exemple de la tâche d'initialisation du Workflow (la tâche "ac:dispatch-application"), une entrée doit être définie pour la propriété "notifyMe" et une autre pour l'aspect "bpm:assignee".
La figure 28 montre les entrées de ces deux éléments dans le bundle I18N.

Figure 28 : I18N de la tâche ac:dispatch-application

```
ac_workflowmodel.type.ac_dispatch-application.title=Acheminer une demande de crédit vers le service du
 crédit.
ac_workflowmodel.type.ac_dispatch-application.description=Acheminer une demande de crédit vers le
 service du crédit.

...
# Associations

ac_workflowmodel.association.ac_assignee.title=Agent en charge du crédit
ac_workflowmodel.association.ac_assignee.description=Agent en charge du crédit
```

Dans notre cas d'étude, nous avons défini la ressource bundle d'internationalisation du Workflow dans le fichier de propriétés : "approbationcreditworkflow-model.properties".

IV-3-3 - Réalisation d'un dashlet pour l'affichage des tâches relatives aux crédits dans le tableau de bord des utilisateurs

Un dashlet est un composant UI Alfresco qui permet d'afficher des informations dans une zone du tableau de bord des utilisateurs.
La réalisation du dashlet est classique et nécessite des configurations XML et du code java. Nous avons réalisé un dashlet Alfresco pour afficher les tâches en attente de traitement par les utilisateurs dans leur tableau de bord.
La figure 29 montre l'affichage des tâches d'un utilisateur dans la dashlet des tâches de demande de crédit.

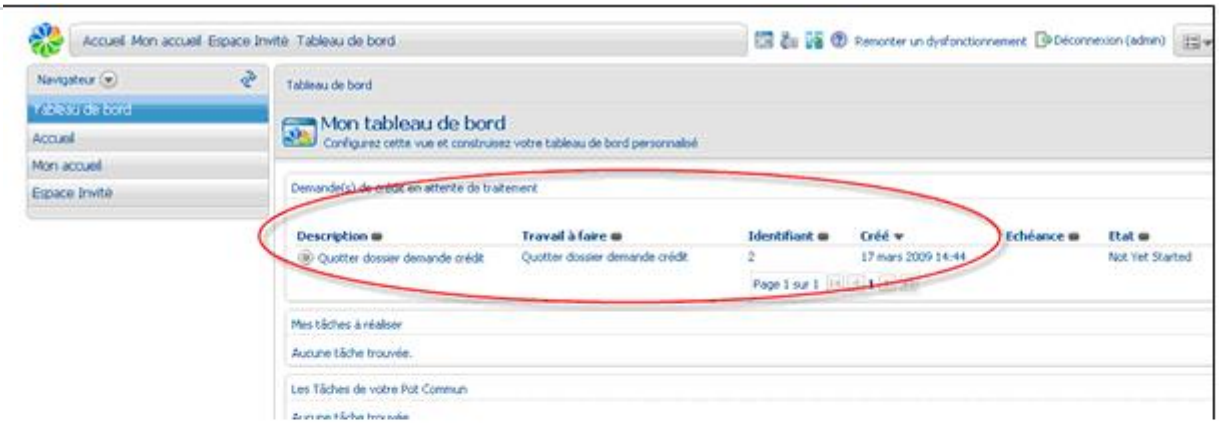


Figure 29 : Dashlet des tâches de demande de crédit

Cette figure montre un travail à faire "Quotter dossier demande crédit" dans le dashlet "Demande(s) de crédit en attente de traitement".
Créer un dashlet, c'est :

- le configurer dans le fichier "web-client-config-custom.xml"
- définir la page jsf du dashlet
- définir le backed-bean de la page jsf ci-dessus
- définir la dashlet dans la liste des dashlet à afficher par défaut

- 1 . Le configurer dans le fichier "web-client-config-custom.xml":

Figure 30 : Configuration du dashlet des tâches de demande de crédit

```
<dashlet  id="loan-application-to-threat"
   label-id="my_loan_to_treat_title"
   description-id="my_tasks_todo_desc"
     jsp="/jsp/extension/workflow/custom-tasks-todo-dashlet.jsp"
   allow-narrow="false" />
```

Dans cette configuration (figure 30), "my_loan_to_treat_title" représente l'entrée I18N du label de dashlet qui est affiché à l'interface utilisateur (dans notre cas de figure, il s'agit du label "Demande(s) de crédit en attente de traitement").
La JSP du dashlet est la JSP custom-tasks-todo-dashlet.jsp.

- 2 . Définir la page jsf du dashlet:
La JSP de notre cas d'étude est la JSP "custom-tasks-todo-dashlet.jsp". Cette JSP permet de lister les tâches assignées à l'utilisateur connecté.

- 3 . Définir le Backed-Bean :
Le Backed Bean est le managed Bean de la dashlet. C'est ce Bean qui est référencé dans la JSP du dashlet à travers les expressions EL.

- 4 . Définir la dashlet dans la liste des dashlets à afficher par défaut lorsque les utilisateurs se connectent:
La liste par default des dashlets constitue les dashlets qui vont être affichées dans l'ordre défini lorsque l'utilisateur se connecte. Dans notre cas d'étude, la liste est la suivante :

Figure 31 : La liste des dashlets à afficher par défaut

```
<default-dashlets>
            <dashlet id="loan-application-to-threat" />
            <dashlet id="tasks-todo" />
            <dashlet id="pooled-tasks" />
            <dashlet id="myspaces-webscript" />
</default-dashlets>
```

Cette liste place la dashlet des demandes de crédit en tête, puis la dashlet de base du noyau Alfresco (qui affichera les tâches autres que celle relatives au crédit), puis les tâches du Pot Commun de l'utilisateur et en fin le composant web Script AJAX qui affiche l'arborescence des dossiers du repository.

IV-4 - Implémentation des fonctionnalités réalisées au sein du Workflow

Les principales fonctions sont :
- Numérisation et archivage de la demande manuscrite du crédit
- Création de la fiche interne de demande de crédit
- Création d'un garant
- Création d'une garantie
- Création d'un document d'identification
- Création du contrat

Le Workflow doit s'interfacer avec le système d'information de l'entreprise notamment pour invoquer certains services existants ou à créer.

Cet interfaçage peut être réalisé via divers moyens : les web services, ou carrément un ESB (Mule ESB) en particulier dans le cadre de l'orchestration de processus hétérogènes.
L'interfaçage avec le SI est en dehors de l'objet de ce document.
Dans les paragraphes qui suivent, nous verrons brièvement comment chaque fonctionnalité a été implémentée.

IV-4-1 - Implémentation de la fonctionnalité de Numérisation et de l'Archivage

Une chaîne de numérisation est une infrastructure logicielle et matérielle mise en oeuvre pour automatiser la chaîne "Scanner - OCR - Indexage - Archivage".
Le scanner produit une image du document. La reconnaissance de caractère est effectuée sur l'image afin de générer un document Texte ou PDF. Pour faciliter les recherches, l'indexage est également fait. De nos jours, presque tous les scanners réalisent l'OCR et produisent des documents textes (Word, RTF) et des documents au format PDF. Néanmoins, il reste impossible de réaliser des traitements tels que le dimensionnement d'image, les corrections, la lecture automatique de document (LAD), etc..

Pour adresser les traitements complexes sur le document numérisé, **Koossery Technology** utilise deux technologies propriétaires: Kofax et eCopy.

Pour des solutions des chaînes simples, **Koossery Technology** met en oeuvre la chaîne ci-dessous:

Figure 32 : chaîne de numérisation simple

```
<Scanner OCR - Bibliothèque Tiger OCR (Optionnelle) - CIFS - Extraction de métadonnées -
       - Indexage - GED />
```

La bibliothèque Tiger OCR est une librairie JAR qui permet de réaliser de l'OCR dans une application JAVA sur une zone donnée de l'image. Nous utilisons cette librairie pour réaliser la reconnaissance de caractère sur des Chèques Bancaire, des factures etc..
Pour notre cas d'étude, nous avons monté la chaîne de numérisation simple permettant de numériser les demandes manuscrites des clients et de les intégrer automatiquement et directement dans un espace du service du courrier de la GED.
La chaîne prend en compte l'extraction de métadonnées et l'indexage automatique.

IV-4-2 - Implémentation de la fonctionnalité de création des fiches et des rapports

La fiche de demande de crédit est un gros formulaire de saisie de données sectionné en étapes. Les informations d'une fiche de demande de crédit sont saisies à travers un Wizards Alfresco.

Un Wizard est un composant UI qui permet de saisir des informations en étapes. Nous avons réalisé la saisie des fiches de demande de crédit en ce servant du mécanisme conceptuel des Wizards d'Alfresco.

Créer un Wizard Alfresco c'est faire des configurations XML et écrire du code JAVA. La configuration XML concerne la configuration du Client Web (l'application Web) pour l'affichage des étapes du Wizard et l'écriture du code consiste à coder le Backed-Bean (Managed Bean) du Wizard.

La réalisation des wizard Alfresco fera l'objet d'une publication ultérieure.
Nous allons néanmoins brièvement voir comment développer un Wizard Alfresco à travers l'exemple de la fiche de demande de crédit.
Voici les étapes suivies pour créer le Wizard de la fiche de demande de crédit :

- configuration du wizard dans le client web
- écrire les pages jsf des différentes étapes d'attribution du crédit
- mise en place des bundles d'internationalisation i18n
- définition du backed-bean du wizard

- 1 : Configuration du Wizard dans le Client web (web-client-config-custom.xml)

Figure 33 : Configuration du Wizard de la fiche de demande de crédit

```
<config>
      <wizards>
         <wizard name="createCreditRecordApplication"
          managed-bean="CreateCreditRecordAppWizard"
          title-id="create_credit_record_app_content_title"
          description-id="create_credit_record_app_content_desc"
          icon="/images/icons/new_content_large.gif">
          <step name="partie1" title-id="credit_write_part1"
               description-id="create_credit_record_app_step1_desc">
           <page path="/jsp/extension/content/create-credit-record-app-wizard/part1.jsp"
            title-id="create_credit_record_app_content_step1_title"
            description-id="create_credit_record_app_content_step1_desc"
            instruction-id="default_instruction" />
          </step>
          <step name="partie2" title-id="credit_write_part2"
            description-id="create_credit_record_app_step2_desc">
           <page path="/jsp/extension/content/create-credit-record-app-wizard/part2.jsp"
            title-id="create_credit_record_app_content_step2_title"
            description-id="create_credit_record_app_content_step2_desc"
            instruction-id="default_instruction" />
          </step>
          <step name="partie3" title-id="credit_write_part3" <br/>
            description-id="create_credit_record_app_step3_desc">
           <page path="/jsp/extension/content/create-credit-record-app-wizard/part3.jsp"
            title-id="create_credit_record_app_content_step3_title"
            description-id="create_credit_record_app_content_step3_desc"
            instruction-id="default_instruction" />
          </step>
```

Figure 33 : Configuration du Wizard de la fiche de demande de crédit

```
            </wizard>
        </wizards>
</config>
```

Nous pouvons remarquer que ce Wizard est composé de 3 étapes (partie1, partie2 et partie3). Chaque étape est une vue JSF (les éléments page) qui peut contenir d'autres Wizards ou des dialogs Alfresco. Nous verrons ce que c'est qu'un dialog Alfresco et brièvement comment le réaliser.
Le nom du Bean Managé du Wizard est "CreateCreditRecordAppWizard" et il est configuré dans le ficher "faces-config-custom.xml". Ce même Bean est utilisé comme Bean Managed des vues des étapes (step) du Wizard.

- 2 : Définition des pages jsf des étapes
L'enchaînement des pages du Wizard est réalisé par le Wizard Framework du noyau de la plateforme Alfresco. Celui-ci utilise la configuration de la figure30 pour savoir quelle vue d'étape il faut afficher. Dans notre cas, nous avons 3 JSP pour les trois étapes : part1.jsp, part2.jsp et part3.jsp.

- 3 : Définition de l'I18N du Wizard
Il s'agit de définir les entrées "title-id" et "description-id" du Wizard dans le bundle.

- 4 : Création du Bean Managé du Wizard
C'est la dernière opération. Il s'agit d'écrire le code java du Bean Managé du Wizard. Le Bean managé d'un Wizard est classique. Toutes les méthodes correspondants aux actions "suivant" , "précédent", "modifier les labels des boutons", "générer de nouveaux boutons", "enregistrer les données saisies", etc.. doivent être implémentées.

IV-4-3 - Implémentation de la création d'un garant

Un garant est une personne physique qui se porte garant pour un crédit. L'entité garant est un type de contenu défini dans le modèle. Les garants sont choisis ou créés lors de la création de la fiche de demande de crédit. Nous avons réalisé la création d'un garant grâce au Framework Conceptuel Dialog d'Alfresco. La réalisation des Dialogs Alfresco fera l'objet d'une publication ultérieure. Néanmoins, nous allons présenter brièvement un petit exemple.

Créer un Dialog Alfresco c'est faire des configurations XML et écrire du code JAVA. La configuration XML concerne la configuration du Client Web pour l'affichage du dialog et l'écriture du code JAVA concerne l'écriture du Bean managé du dialog. Voici les étapes suivies pour créer le Dialog de création des garants :

- configuration du Dialog dans le client web
- écrire les pages jsf des différentes étapes d'attribution du crédit
- mise en place des bundles d'internationalisation i18n
- définition du backed-bean du wizard

- 1 : Configuration du Dialog dans le Client web (web-client-config-custom.xml)

Figure 34 : Configuration du Client Web du Dialog de création des garants

```
<dialog name="create_guarantor_dialog"
   page="/jsp/extension/content/create-guarantor-dialog.jsp"
   managed-bean="CreateGuarantorDialog"
   icon="/images/extension/icons/new_person.gif"
   title-id="create_guarantor_dialog_title"
   description-id="create_guarantor_description_dialog">
</dialog>
```

Les éléments constitutifs du Dialog Alfresco sont : la JSP du dialog (dans notre cas il s'agit de la JSP create-guarantor-dialog.jsp), le Bean managé (dans notre cas il s'agit du Bean CreateGuarantorDialog), l'icône (dans notre cas il s'agit de l'image new_person.gif), du titre et de la description.

- 2 : Définition de la JSP du Dialog

Il s'agit d'écrire la page JSP create-guarantor-dialog.jsp. Les balises (tags librairies) JSF sont utilisées pour la construction de la JSP. Les Expressions EL sont utilisées pour le lien avec le Bean Managé.

- 3 : Définition de l'I18N du Dialog
L'internationalisation du dialog se réduit aux entrées "title-id" et "description-id" de la configuration du dialog.

- 4 :Création du Bean Managé du Dialog
C'est la dernière opération. Il s'agit d'écrire le code java du Bean Managé du Dialog. Le Bean managé d'un Dialog est classique. Certaines méthodes doivent être implémentées pour les actions "annuler" et "Ok", "modifier les labels des boutons", "générer de nouveaux boutons", "enregistrer les données saisies", etc..

IV-4-4 - Implémentation de la création d'une garantie

Une garantie est aussi un type de contenu configuré dans le content model. La création des garanties se fait également à partir du Framework Dialog d'Alfresco. Bien vouloir ce reporter à la section (paragraphe IV-4-3) d'implémentation de la création d'un garant pour les détails techniques.

IV-4-5 - Implémentation de la création d'un document d'identification

Un document d'identification est aussi un type de contenu configuré dans le content modèle. La création des documents d'identification se fait également à partir du Framework Dialog d'Alfresco. Bien vouloir ce reporter à la section (paragraphe IV-4-3) d'implémentation de la création d'un garant pour les détails techniques.

IV-4-6 - Implémentation de la création d'un contrat

Un contrat est défini dans notre application comme un type de contenu. La différence de ce modèle de création avec les autres est que, le contrat n'est pas saisi mais plutôt créé par interfaçage avec le Système d'Information de l'établissement de crédit.

L'interfaçage de l'application ECM/BPM d'approbation de crédit avec le SI de l'établissement de crédit est réalisé via l'esb Mule ESB.

L'aspect interfaçage avec le Système d'Information de l'établissement de crédit est en dehors de l'objet de ce document.

V - Conclusion

Nous avons vu à travers cet article comment l'utilisation combinée de la plate-forme Alfresco (pour la gestion des documents et des contenus) et de jBoss BPM (pour les workflow) permet de bénéficier d'APIs puissants pour la réalisation des applications combinant à la fois la gestion documentaire et gestion de contenu avec les Workflows. Ces frameworks constituent le socle technique du **Centre de Compétence Alfresco/jBPM** de **Koossery Technology**

VI - Remerciements

Tous nos remerciements au comité de relecture **developpez.com**

Nous remercions également tous les membres du **Centre de Compétence Alfresco/jBPM** de **Koossery Technology** ainsi que les autres pool techniques (notamment le **Centre de Compétence ADempiere ERP**) pour l'esprit de curiosité et de partage des connaissances. Leurs multiples questions et critiques nous ont été grandement bénéfiques.