Moteur de workflow

Mise en oeuvre d'OpenWFE

Version 1.0 - 25 septembre 2006

SOMMAIRE

1 AVANT PROPOS

Dans le cadre d'une de ses missions, Clever Age a eu à faire le choix d'un moteur de worflow pour l'intégrer dans une application de gestion puis a pu ensuite réaliser cette intégration.

Ce document commence par présenter succinctement OpenWFE, un moteur de workflow en Java. Les problématiques d'intégration d'un moteur de workflow en général, et plus particulièrement OpenWFE, sont ensuite présentées en détail.

2 PRÉSENTATION

2.1 Quelques définitions

Workflow :

On retrouve dans wikipedia[1] la définition suivante pour Workflow : « On appelle « workflow » (traduisez littéralement « flux de travail ») la modélisation et la gestion informatique de l'ensemble des tâches à accomplir et des différents acteurs impliqués dans la réalisation d'un processus métier (aussi appelé processus opérationnel ou bien procédure d'entreprise). Le terme de « workflow » pourrait donc être traduit en français par « gestion électronique des processus métier ». De façon plus pratique, le workflow décrit le circuit de validation, les tâches à accomplir entre les différents acteurs d'un processus, les délais, les modes de validation, et fournit à chacun des acteurs les informations nécessaires pour la réalisation de sa tâche. Pour un processus de publication en ligne par exemple, il s'agit de la modélisation des tâches de l'ensemble de la chaîne éditoriale. »

Moteur de workflow :

On retrouve dans wikipedia[2] la définition suivante pour moteur de workflow : « Un moteur de workflow est un dispositif logiciel permettant d'exécuter des instances de workflow (l'enchaînement des activités décrit par la définition de processus de workflow) ».

2.2 Besoins d'intégration d'un moteur de workflow

Pourquoi vouloir intégrer un moteur de workflow dans une application ?

Intégrer un moteur de workflow dans une application n'est pas évident à réaliser et cela pose un certain nombre de questions d'architecture que nous allons traiter plus loin, et une charge de développement systématiquement alourdie par rapport à l'implémentation d'un workflow simple intégré à l'application.

On n'intègre donc pas un outil de workflow dans une application juste pour faire joli mais pour répondre à des besoins. C'est pour cela d'ailleurs que ce type d'architecture est rarement mis en place. Les différents éléments suivants peuvent être des signes sur la pertinence de mettre en place un moteur de workflow, il peut falloir :

- Que l'application comporte un certain nombre de workflows
- Que les workflows soient appelés à changer souvent
- Que des utilisateurs non informaticiens aient à modifier des worflows
- Qu'il faille gérer des versions de workflow. C'est à dire que l'on doit pouvoir terminer les

1 http://fr.wikipedia.org/wiki/Workflow

2 http://fr.wikipedia.org/wiki/Moteur_de_workflow

processus commencé avec un workflow d'une certaine version et commencer des nouveaux processus avec une autre version.

- Que l'application soit concentré sur les processus et non sur les données.

Dans ce cas, les coûts de la mise en place du moteur de workflows seront amortis sur les différents cycles de vie de l'application.

2.3 Présentation de OpenWFE

Nom	OpenWFE
Site internet	http://web.OpenWFE.org/
Version étudiée	1.7.0
Licence	BSD
Plateformes supportées	Toutes
Logiciels requis	Java Runtime Environnement
Tarification	Gratuit

OpenWFE est un moteur de workflow développé en Java. Il permet de modéliser des workflows puis de les instancier et de contrôler leur cycle de vie.

Les workflows sont modélisés dans un langage XML propre à OpenWFE.

OpenWFE est constitué de différents modules :

- L'*engine* : le moteur à proprement parlé
- le *workList* : un module permettant de gérer les instances de workflows durant leur cycle de vie
- l'*APRE* : un module permettant de lancer des actions automatisées (envoie de mail ...)
- Le *workflow definition server* : un serveur fournissant les définitions de workflow, c'est à dire les workflows modélisés dans la syntaxe OpenWFE.
- Le *webclient* : une application web permettant de contrôler de manière basique les fonctionnalités du moteur de workflow.
- *Droflow* : une application permettant de modéliser des workflows via une interface web. Cette application permet de construire les workflows sans avoir à éditer des fichiers XML. Par contre, il n'y a pas d'abstraction du langage XML. Cette application reste assez technique.

Un moteur de workflow n'est pas utilisable tel quel, son intérêt est qu'il permet de gérer d'une manière très souple les workflows dans une application. Il faut donc intégrer le moteur de workflow dans une application spécifique. Le webclient d'OpenWFE est une implémentation très sommaire et n'offre que les services du moteur de workflow, on peut tout de même l'utiliser comme outil d'administration pour visualiser l'état d'OpenWFE et des différents workflows instanciés.

3 Architecture d'OpenWFE

OpenWFE comprend un ensemble d'exécutables. Chacun des modules présentés au dessus est exécuté dans une JVM spécifique. La communication entre chaque module se fait par des connexions RMI (remote method invocation).

3.1 Concepts structurants dans OpenWFE

Comme tout logiciel, OpenWFE est basé sur des concepts fonctionnels et techniques qui font la spécificité de l'outil. Il est donc important de maîtriser ces concepts avant de se lancer dans son intégration.

Participants

Dans la définition d'un workflow, on fait systématiquement intervenir des acteurs. Ces acteurs vont effectuer des activités, le rôle du moteur de workflow est de faire avancer le processus d'activité en activité.

« Participant » est le terme dans OpenWFE désignant ces acteurs. Un participant peut typiquement être :

- Un utilisateur : John doit exécuter l'activité de valider les feuilles de temps
- Un rôle : le responsable de l'équipe doit exécuter l'activité de valider les feuilles de temps
- Un robot : l'application externe valide par défaut les feuilles de temps si le responsable ne l'a pas fait dans les 2 jours.

Il est déconseillé de modéliser des workflows avec des personnes réelles mais plutôt de le faire avec des rôles. On a ainsi moins de maintenance à faire le jour ou John change de poste.

Les participants peuvent être définis dans 3 supports de stockage au choix, un fichier XML de configuration, une base de donnée relationnelle, un serveur LDAP.

Workflow definitions

Les « workflows definitions » sont les modèles de workflow que l'on veut mettre en place dans son application. Ces définitions de workflows sont des fichiers XML (un fichier par définition) décrivant les états, les conditions sur les changements d'états et les enchaînements

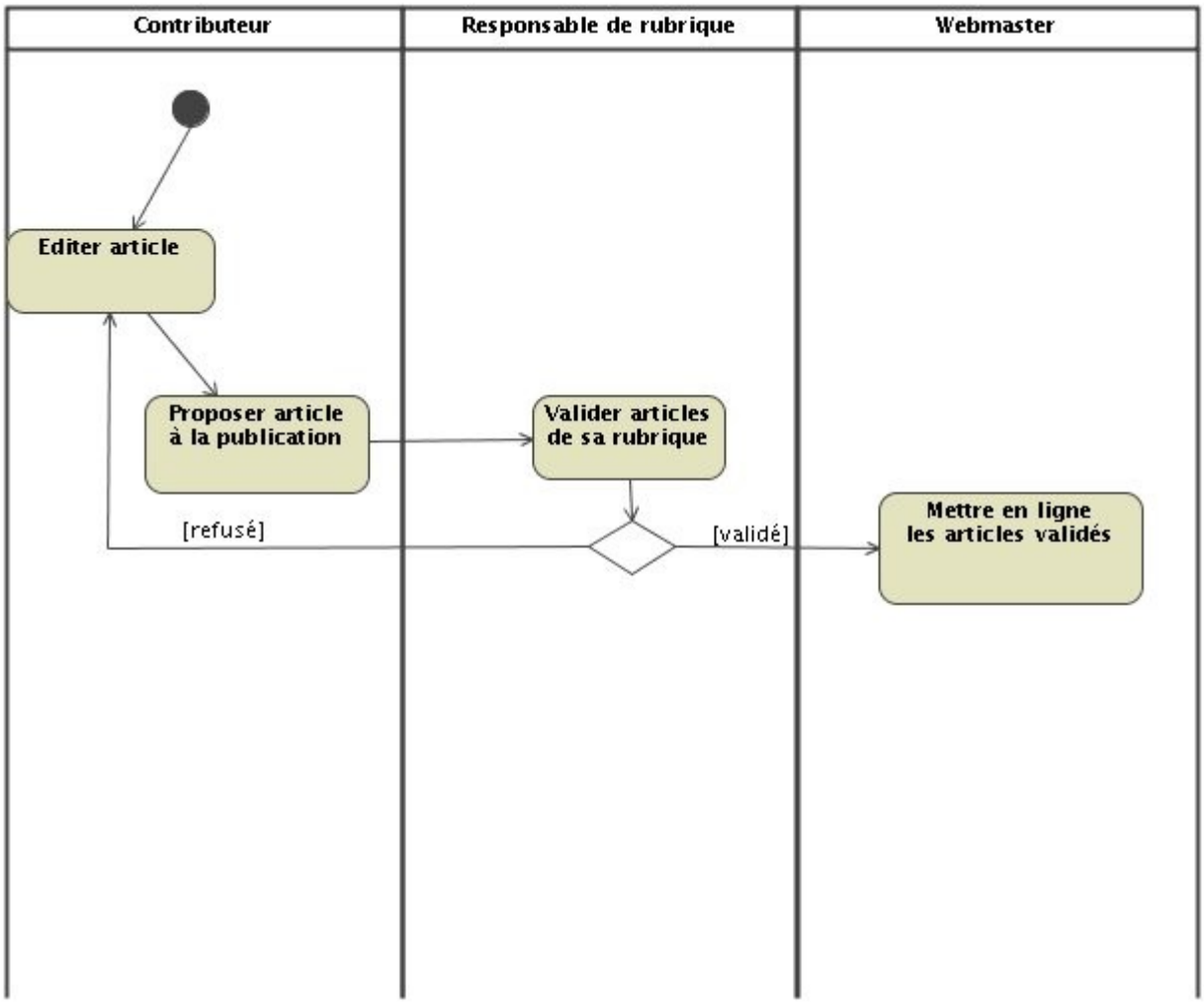
d'interaction avec les utilisateurs.

Ces fichiers peuvent être édités de deux manières :

- soit à la main, une documentation sur le site Internet présente la syntaxe de manière assez complète.
- soit en utilisant l'application web droflow.

La syntaxe utilisée par OpenWFE est une syntaxe propriétaire proche de LISP. Elle ne s'appuie pas sur les standards émergeant comme BPML. On construit les processus avec des enchaînements d'activités insérés dans des boucles ou des séquences concurrentes.

Voici un exemple de workflow de processus rédactionnel simple.



Il se traduit de la manière suivante dans la syntaxe OpenWFE :

```
<?xml version="1.0" encoding="UTF-8"?>
<process-definition xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:noNamespaceSchemaLocation="http://www.openwfe.org/flowdef_r1.6.2.xsd"
                    name="publication" revision="0.1">
    <description language="default">
    Process de publication sur un site internet
    </description>
....
    <set field="statut" value=""/>
    <sequence>
        <loop>
            <sequence>
                <set field="__subject__" value="Editer article"/>
                <participant ref="contributeur"/>
        ....
                <set field="__subject__" value="Proposer article à la publication"/>
                <participant ref="contributeur"/>

                <set field="__subject__" value="Valider article"/>
                <!-- Le responsable de rubrique doit modifier
                     le statut pour dire si l'article est validé ou non -->
                <participant ref="responsable-rubrique"/>
            </sequence>

            <until>
                <equals field="statut" other-value="validé"/>
            </until>
        </loop>
        <set field="__subject__" value="Mettre en ligne"/>
        <participant ref="webmaster"/>
    </sequence>

</process-definition>
```

A la question de savoir si l'outil permettra de modéliser tous les cas d'utilisations que mon application pourrait rencontrer dans sa vie, la documentation d'OpenWFE présente les implémentations de patrons habituels.

Le site workflowpatterns[3] a pour but de présenter tous les patrons nécessaires à l'expression de workflows comme par exemple l'enchaînement d'activités ou l'exécution d'activités en parallèle. OpenWFE présente dans sa documentation les implémentations possible pour chacun de ces patrons dans la syntaxe d'OpenWFE.

Instance de workflow : workitem

On appelle une instance de workflow le processus lancé à partir d'une définition de workflow. On va pouvoir avancer d'activité en activité sur cette instance. Il est également possible d'attacher des données à l'instance. Cela peut être décrit dans la définition du workflow. Par exemple, si l'activité « valider le document » est exécutée alors je mets automatiquement le champ « validé » à vrai. Les données peuvent être attachées à l'instance de workflow également de manière dynamique. L'application externe peut ajouter/mettre à jour dans champs dans l'instance. Ces valeurs peuvent être utilisées pour évaluer les enchaînements d'activités ou alors pour être utilisées par l'application externe.

3 http://www.workflowpatterns.com

Utilisateurs et stores

On peut définir différents utilisateurs dans OpenWFE. On rattache à chaque utilisateur un *store*[4]. Un store est un endroit dans lequel sont stockés les instances de workflows. Chaque participant est lié à un store. Un store peut être utilisé par plusieurs participants. C'est à dire que l'on retrouve dans le store les différentes instances de workflow ayant des activités affectées à des participants de ce store.

Store : Store.query

workflow instance	subject	dispatch time	last modified	participant

Store : Store.bravo

workflow instance	subject	dispatch time	last modified	participant
campaign-definition 0.2 : 1147350754405.0	Check publishing constraints	2006-06-15 14:24:28+0200	2006-06-15 14:24:28+0200	role-bravo
campaign-definition 0.2 : 1147338627087.0	Check publishing constraints	2006-06-15 14:24:29+0200	2006-06-15 14:24:29+0200	role-bravo
campaign-definition 0.2 : 1147334892399.0	Check publishing constraints	2006-06-15 14:24:30+0200	2006-06-15 14:24:30+0200	role-bravo
campaign-definition 0.2 : 1150882365057.1	Define creatives	2006-06-21 11:33:51+0200	2006-06-21 11:33:51+0200	role-bravo
campaign-definition 0.2 : 1150882365057.1	Define placements	2006-06-21 11:33:51+0200	2006-06-21 11:33:51+0200	role-bravo

Store : Store.default

workflow instance	subject	dispatch time	last modified	participant
insertion_order 0.2 : 1149604118480.1	Restart insertion order	2006-06-15 14:25:00+0200	2006-06-15 14:25:00+0200	media-sales
insertion_order 0.1 : 1150453811968	Create insertion order	2006-06-16 12:30:13+0200	2006-06-16 12:30:13+0200	media-sales
test-soap 0.4 : 1150469631941	Test participant soap	2006-06-16 16:54:04+0200	2006-06-16 16:54:04+0200	publisher
insertion_order 0.3 : 1150469685285.0.0	Approve insertion order	2006-06-16 16:55:26+0200	2006-06-16 16:55:26+0200	advertiser
insertion_order 0.3 : 1150470152172.0.0	Approve insertion order	2006-06-16 17:02:43+0200	2006-06-16 17:02:43+0200	advertiser
insertion_order 0.3 : 1150470372378.1	Restart insertion order	2006-06-16 17:09:18+0200	2006-06-16 17:09:18+0200	media-sales
insertion_order 0.3 : 1150470546137.2	Restart insertion order	2006-06-21 10:32:02+0200	2006-06-21 10:58:36+0200	media-sales

Store : Store.alpha

workflow instance	subject	dispatch time	last modified	participant

Quand on demande à l'API d'OpenWFE une liste d'instances de workflows, on va pouvoir uniquement filtrer par rapport à l'utilisateur faisant la demande et le store sur lequel faire la recherche. L'organisation des stores est donc une partie importante de la mise en place d'OpenWFE.

APRE : l'exécution automatique d'activités

APRE est un acronyme pour « Automatic Participant Runtime Environment ». Comme énoncé précédemment, il est possible de définir des participants automatiques qui peuvent exécuter des actions. Ce genre de participant est utile notamment dans le cas de timeout. C'est un moyen de prévenir l'application tierce que le temps pour exécuter un processus est écoulé. APRE est implémenté en Java et en Python, ce qui permet d'avoir le choix entre ces deux langages pour écrire ses scripts.

3.2 Problématiques de paramétrage d'OpenWFE

Choix des stores

Les stores sont une notion technique importante dans OpenWFE. Les stores sont configurés dans un fichier XML mais la prise en compte de modifications entraîne le redémarrage du serveur et donc une interruption de service. Le choix des stores a un impact au niveau de :

4 que l'on pourrait traduire par magasin mais nous préférons garder le terme anglais plus riche

- La sécurité : les droits utilisateurs sont liés aux stores.
- Les fonctionnalités de listing demandées par l'application externe. Les instances de workflows peuvent être listées par stores.

Choix des users

Utiliser les mêmes utilisateurs pour l'application externe et le moteur de workflow peut être une solution simple à mettre en oeuvre. Les droits utilisateurs sont définis ainsi dans OpenWFE et l'on pourra s'appuyer sur cette gestion de la sécurité pour présenter aux utilisateurs les instances de workflows sur lesquels il peuvent intervenir.

Il est également possible d'utiliser un utilisateur générique qui aura les droits sur tous les stores. Dans ce cas, la sécurité et les droits d'accès doivent être gérés par l'application tierce. Ce choix est souvent fait quand on regarde les forums d'OpenWFE, cela vient du fait que OpenWFE ne peut gérer que la sécurité des workflows et que dans la plupart des cas, les droits utilisateurs doivent être gérés au niveau des workflows et des données. Les données étant gérées par l'application externe, on laisse la gestion des droits à l'application externe plutôt que de la séparer sur les deux outils.

Choix des modes de persistance

Par défaut OpenWFE enregistre toutes les données sur le système de fichiers dans des fichiers XML. Il existe plusieurs modes de persistance pour les éléments suivants :

- Utilisateurs : XML ou base de données
- Workitems : XML ou base de données. Il existe deux implémentations pour la base de données. La première est un enregistrement de fragments XML dans des champs CLOB, la seconde est basé sur un modèle relationnel plus poussé. Néanmoins, La première est préconisée par rapport à la seconde car elle est mieux maintenue.
- Les participants : XML ou LDAP Il est possible d'utiliser un serveur LDAP pour définir les participants potentiels pour les workflows. Par défaut, les participants sont définis dans un fichier XML.

3.3 Possibilités d'interfaçage avec OpenWFE

OpenWFE fournit une API java qui permet de contrôler complètement toutes les fonctionnalités qu'il met à disposition. Cette API peut être utilisé de plusieurs manières :

- Directement en java (si l'on embarque OpenWFE dans son application développée en java)
- Via des appels Java RMI. C'est l'implémentation utilisée par l'application webclient. L'utilisation de l'API se fait depuis java vers
- Via des appels Rest. Cette dernière solution permet de contrôler OpenWFE via une interface Rest. Des requêtes HTTP contenant du XML permettent de lancer des actions dans OpenWFE. Les réponses d'OpenWFE sont faites en XML.

Pour l'interface Rest, OpenWFE distribue des librairies clientes dans différents langages. L'API Java est retranscrite dans une API du langage spécifique. Il existe ainsi une bibliotèque

pour .Net, Perl, Pnuts, Ruby, PHP, Python. Ces bibliotèques sont de qualités inégales mais l'interface Rest n'étant pas très complexe, leur mise à niveau n'est pas très coûteuse.

4 Architecture d'intégration

4.1 Présentation

Nous avons intégré OpenWFE dans une application de gestion (que nous appellerons CANAP) développée spécifiquement en PHP. Cette application de gestion permet de gérer un ensemble de cas d'utilisations. Nous prendrons comme exemple le processus éditorial présenté plus haut.

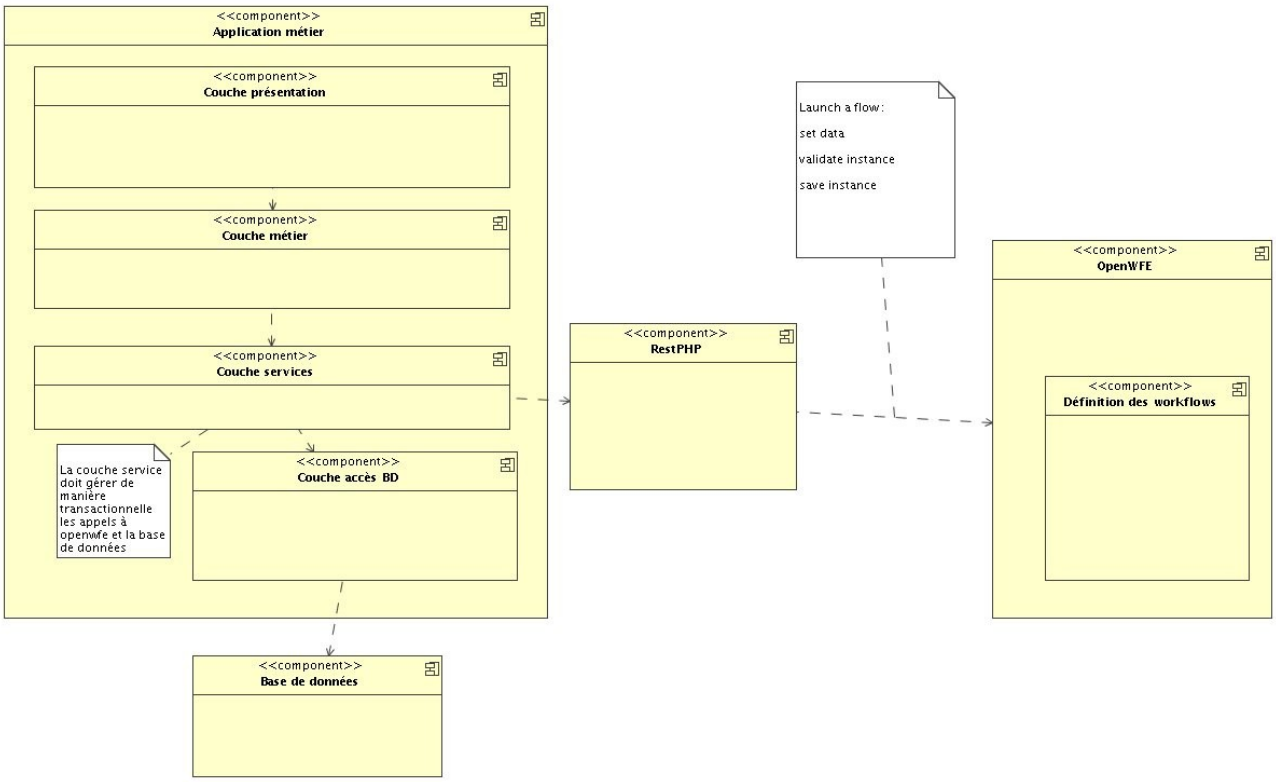
De ce processus métier les fonctionnalités suivantes vont être demandées à l'application :

- Chaque responsable de rubrique voudra lister les articles de sa rubrique en attente de validation.
- Chaque contributeur voudra voir l'avancement des articles qu'il a écris.
- Le contributeur pourra créer/modifier un article dans un état brouillon
- Le responsable pourra valider les articles de sa rubrique
- Le webmaster pourra lister tous les articles validés
- Le webmaster pourra mettre en ligne chaque article

CANAP utilise une base de donnée relationnelle dans laquelle sont stockées les données.

Elle s'appuie sur OpenWFE pour gérer les processus concernant les utilisateurs et les données.

CANAP a une architecture interne découpée en couches suivant le diagramme suivant. Les couches affectés par l'intégration d'OpenWFE sont les couches techniques d'accès aux données.



4.2 Problématique d'intégration d'un outil de workflow dans une application

Intégrer un outil dans une application pose toujours des problématiques. Le moteur de workflow ne déroge pas à la règle. Le fait qu'il soit garant de l'état des données dans le temps en fait un élément très sensible de l'architecture qu'il faut donc soigner.

Rôles des modules

L'application externe et le moteur de workflow ont des rôles bien différents qu'il faut définir rapidement. OpenWFE sait bien gérer les workflows, mais il n'offre pas de fonctionnalités pour gérer les données de manière poussée.

Le rôle d'OpenWFE est de gérer l'enchaînement des activités suivant l'état des données et les informations qu'on lui aura fournies. Il est garant de la gestion des processus. Il sera responsable de faire avancer le processus de mise en ligne d'un article quand le responsable de rubrique validera cet article.

Le rôle de l'application externe est de gérer l'appartenance des données aux bons utilisateurs. Elle est garante de la gestion des données. Elle sera responsable de n'afficher au responsable de rubrique que les articles liés à sa rubrique.

Dénormalisation des données

Les données sont partagées entre les deux systèmes moteur de workflow / application de gestion. A un moment t, ces données peuvent être nécessaires dans les deux systèmes.

Le minimum de données qu'il faut partager entre les systèmes est l'identifiant des données dans les deux référentiels. Il faut que CANAP puisse communiquer à OpenWFE l'id du workflow qu'il veut faire avancer. De la même manière, si je demande à OpenWFE de lister les articles en attente de publication, il faut pouvoir retrouver les contenus des artices pour pouvoir créer l'interface graphique. Sur l'affichage de la liste des articles, deux choix sont possibles :

- OpenWFE remonte une liste d'instances de workflow dans laquelle on a les id d'articles. Pour chaque ligne renvoyée, on fait une requête en base de données pour obtenir le titre de l'article pour constituer l'interface graphique. Si la volumétrie commence à être élevée, les performances peuvent rapidement se dégrader (c'était un des comportements des EJB 1.0 qui a été très longtemps décrié).
- On insère dans les instances de workflows dans OpenWFE certaines données comme le titre des articles pour ne pas requêter la base de données sur l'affichage de listes d'instances de workflows. La donnée titre se trouve alors à deux endroits à la fois, dans la base de données relationnelle et dans OpenWFE, on a en contre partie les problématiques de synchronisation classiques des données.

Conflit de rôles

Les rôles ont été clairement séparés entre la partie workflow et la partie gestion des données. On peut avoir dans certains cas des conflits dans la gestion des informations. Des fonctionnalités triviales dans une application basée entièrement sur une base de données relationnelle peut devenir complexe avec l'utilisation d'un module externe.

Dans les fonctionnalités attendues, il y a l'affichage au responsable de rubrique des articles de sa rubrique devant être validés. OpenWFE est garant de l'état des articles, et CANAP est garante de l'appartenance de l'article à la rubrique du responsable. Il y a donc besoin de faire participer les deux modules pour récupérer la liste.

Les solutions natives sont :

- Lister les articles de la rubrique depuis CANAP et regarder pour chacun d'eux son état dans le workflow
- Lister les articles en attente de validation depuis OpenWFE et regarder pour chacun d'eux la rubrique à laquelle il appartient.
- Dans des cas de volumétrie importante, aucune de ces solutions n'est viable.

Pour intégrer un cas équivalent nous avons dû étendre OpenWFE pour ajouter une fonctionnalité de filtre sur le listing d'instances de workflow basé sur un champ défini dans les instances de workflow et sur un paramètre passé à l'interface REST. Ainsi, il est possible de filtrer les instances sur des données qui n'ont pas de sens dans OpenWFE.

Dans les prochaines versions d'OpenWFE, on peut s'attendre à voir apparaître une telle fonctionnalité dans le coeur du produit, la problématique ayant été remontée plusieurs fois et une solution viable ayant été trouvée.

Gestion transactionnelle des cas d'utilisations

Dans toutes les applications de gestion, on tient à s'assurer de la cohérence de ses données. Pour cela on s'appuie dans 95% des cas sur la propriété transactionnelle des bases de données relationnelles et le fameux acronyme ACID :

- Atomique : Toutes les opérations sont exécutées ou aucune ne l'est.
- Cohérente : Mon système passe d'un état cohérent à un autre état cohérent.
- Isolée : Les modifications que j'apporte peuvent être isolées et non visibles des autres processus tant que je n'ai pas validé ces modifications.
- Durable : Les transactions validées sont non répudiables, c'est à dire qu'elles sont conservées dans le temps même si le système venait à s'arrêter brutalement.

Dans le cas de l'intégration d'un moteur de workflow externe dans l'application, il est difficile de s'assurer de l'atomicité des modifications faites.

Dans le cas de la création d'un nouvel article, au moment où l'utilisateur enregistre son article, il faut instancier un workflow pour cet article et l'enregistrer en base de données. OpenWFE ne permet pas de définir de cadre transactionnel à un ensemble d'actions.

Au niveau de la base de données, il est possible pour certaines bases de données de définir des transactions avec validation en deux temps (*two phases commit*) qui sont utilisées dans les transactions distribuées.

Ainsi, si l'on veut avoir une atomicité des modifications, il faut lancer les actions suivant le séquencement suivant :

- Début de la transaction base de données
- Exécution des modifications en base de données
- Preparation du commit (la base de données n'applique pas les changement mais s'engage à les accepter si la demande en est faite)
- Modifications sur le workflow
- Commit sur la base de données (application réelle des modifications sans qu'une erreur puisse survenir)

En théorie ce modèle fonctionne, mais il a le désavantage d'être très gourmand en ressources. Par exemple, les transactions en deux temps demandent aux bases de données MySQL de poser des verrous sur les tables complètes. Cela veut dire un ralentissement très fort des applications en cas d'utilisation à plusieurs utilisateurs simultanés. Cette solution n'est donc pas mise en pratique.

Pour s'assurer de la conformité de l'état dans le workflow et des données dans la base de données, il faut se reporter aux rôles de chaque brique de l'application. Ainsi :

- La couche applicative est garante des données : c'est elle qui doit avoir le pas sur les données dans le moteur de workflow.
- Le moteur de workflow est garant de l'état des processus. C'est lui qui doit avoir le dessus sur l'état dans la base de données.

Pour pouvoir s'assurer de la reprise correcte sur erreur, il faut exécuter les actions dans l'ordre suivant :

- Enregistrer les données correspondantes à l'état courant
- Notification du moteur de workflow
- Application des règles de gestion faisant suite à l'état retourné par le workflow

Intégration des référentiels utilisateurs dans un annuaire LDAP

On peut dans 95% des cas assimiler les rôles de l'application à des participants (des acteurs) dans les workflows. Plutôt que de gérer deux référentiels dans chacune des applications, il est intéressant de tout gérer dans un annuaire commun, typiquement un annuaire LDAP.

OpenWFE a une première implémentation de récupération des participants depuis un annuaire LDAP. Cette implémentation n'est pas satisfaisante mais offre une première base pour faire sa propre implémentation, spécifique à sa propre organisation.

Dans l'annuaire LDAP, l'attribution des rôles se fait soit par l'arborescence des utilisateurs dans le LDAP (représentant la hiérarchie dans l'organisation), soit par la définition de groupes spécifiques pointant sur un ensemble d'utilisateurs (quand la hiérarchie ne correspond pas aux droits des utilisateurs dans l'application).

Dans le cas des groupes spécifiques, il faut modéliser la gestion des rôles de la manière suivante :

- Un groupe LDAP correspond à un participant dans OpenWFE et un rôle dans l'application
- Un utilisateur dans l'annuaire LDAP peut être intégré dans un ou plusieurs groupes.

L'administration des droits et utilisateurs peut ainsi se faire directement dans l'annuaire LDAP en utilisant les outils d'administration externes à OpenWFE et notre application.

5 Synthèse

OpenWFE est un moteur de workflow puissant permettant de modéliser les principaux cas d'utilisation. Il propose une grande souplesse de configuration ce qui offre différentes options dans son intégration. Les connecteurs qu'il propose permettent de pouvoir l'utiliser avec des technologies autres que Java.

Son intégration dans une application de gestion pose un ensemble de problématiques qui ne sont pas forcément simples à résoudre. Ces problématiques sont :

- La définition des rôles et la résolution de conflits en résultant
- La dénormalisation des données
- La gestion transactionnelle des actions
- L'intégration des référentiels communs

La grande valeur ajoutée de l'utilisation d'un moteur de workflow dans une application a un coût qu'il faut bien évaluer pour faire les bons choix technologiques avec de se lancer dans une implémentation.