

Problème du système de WORKFLOW

Cahier des charges proposé par P. Mathieu et JC Routier

Equipe SMAC/LIFL

<http://www.lifl.fr/SMAC>

email : mathieu@lifl.fr routier@lifl.fr

17 mars 2003

1 Introduction

Un système de WORKFLOW est un système informatique de suivi automatique d'informations entre les différents utilisateurs du système à travers le réseau. L'un des plus utilisés dans l'industrie est le logiciel bien connu **Notes** fourni par **I.B.M.** qui possède de nombreux modules dédiés à ce problème. Le système de WORKFLOW permet notamment de spécifier un circuit de destinataires à suivre par un message envoyé puis de garantir que chaque destinataire du document l'a bien reçu. Il permet aussi à l'émetteur de voir l'avancement de la consultation de son document par les destinataires. Circuit et message sont deux choses bien distinctes car un même circuit peut être utilisé par plusieurs messages, de même qu'un message peut éventuellement suivre plusieurs circuits. Ce type de système est aujourd'hui très en vogue en recherche car il pose de nombreux problèmes de passage à l'échelle pour les WORKFLOW à circuits complexes mettant en oeuvre de nombreux utilisateurs.

Ce problème est bien évidemment clairement distribué et favorise donc une conception par agents. Il y a évidemment au moins un agent "interface" par personne physique participant au WORKFLOW, mais aussi éventuellement d'autres agents "invisibles" permettant la communication, la gestion des circuits ou simplement la gestion des messages.

Le premier intérêt de ce problème est donc d'abord de voir comment chacun va effectuer le contrôle de son SMA, quels sont les agents dont il aura besoin et quelle va être la dynamique des envois de messages. Dans les propositions de chacun, on prendra garde à bien insister sur ces points.

2 Scénarii proposés

Afin de pouvoir comparer les différentes solutions mises en oeuvre pour résoudre ce problème, nous proposons un scénario typique. Trois variantes sont proposées, elles permettent d'introduire une gradation dans la complexité du système de WORKFLOW. Pour chacune des variantes proposées ci-dessous, nous fournissons un exemple permettant de clarifier un peu le problème.

2.1 Variante 1

Tous les agents sont présents, le circuit est séquentiel

Chaque intervenant du système est symbolisé par un agent qu'il connecte au serveur de WORKFLOW et qui reste ensuite actif sur la machine de l'utilisateur. Pour simplifier, les documents seront de simples textes (on pourrait aussi imaginer un document XML ou même une url). Les circuits sont aussi constitués d'un simple texte contenant la suite ordonnée des agents qui doivent recevoir le message. Le message doit passer automatiquement à l'agent suivant dans la liste dès que le précédent a validé la lecture. Chaque utilisateur doit pouvoir obtenir le suivi des messages qu'il a envoyé à la demande. Il doit notamment pouvoir savoir pour chaque message qu'il a envoyé, qui l'a lu et où le message en est. Dans cette première variante on considère que tous les agents sont présents dès le premier message envoyé et qu'ils ne quittent jamais le système. Il n'y a donc pas de dynamique.

Pour simplifier on peut considérer que chaque interface utilisateur est constituée d'un onglet d'envoi de message dans lequel on tape le texte à envoyer et la liste des destinataires formant le circuit, ainsi que d'un onglet de visualisation des messages qui arrivent avec un simple bouton de validation.

Exemple. *On possède 4 utilisateurs Pierre, Paul, Jean, Jacques. Ces quatre utilisateurs sont connectés. Pierre envoie le message "Hello world" associé au circuit "Paul, Jean, Jacques". Paul est alors le premier à recevoir le message. Tant qu'il ne l'a pas validé, rien ne se passe, mais dès qu'il le valide, le message part aussitôt au suivant, c.a.d Jean. Dès que cette validation est effectuée, Pierre est averti qu'une personne de plus a validé son message. Une fois que Jacques l'a validé, le circuit est terminé et n'a plus lieu d'être.*

2.2 Variante 2

Gestion dynamique des agents

Dans cette seconde variante, les agents ne sont pas forcément présents dans le système quand un message leur est adressé. Chaque utilisateur peut se connecter et se déconnecter à sa guise. Si un message lui est destiné et qu'il n'est pas présent, le système stocke le message jusqu'à son retour.

Exemple. *On possède 4 utilisateurs Pierre, Paul, Jean, Jacques. Seuls Pierre et Paul sont connectés. Pierre envoie le message "Hello world" associé au circuit "Paul, Jean, Jacques". Paul est alors le premier à recevoir le message. Dès qu'il le valide, le message part aussitôt au suivant, c.a.d Jean. Mais Jean n'est pas connecté. Le message reste donc en attente. Plus tard, Jean se connecte, il reçoit alors automatiquement ce message qu'il peut alors valider.*

2.3 Variante 3

Circuits complexes

Les circuits sont maintenant plus complexes. Les destinataires ne sont pas forcément traités en série mais aussi en parallèle. L'idéal est alors de décrire le circuit en format XML avec par exemple un tag **sequence** regroupant les agents qui reçoivent le message en séquence et un tag **par** pour les agents qui le recevront en parallèle. Dans ce dernier cas il faut la validation de tous les destinataires en parallèle avant de passer à la suite.

```
<message>
<from>mathieu@lifl.fr</from>
<to>
  <agent>beaufils</agent>
  <par>
    <sequence>
      <agent>routier</agent>
      <agent>secq</agent>
    </sequence>
    <agent>verrons</agent>
  </par>
</to>
<body>
  voici le message que je souhaite faire transiter
</body>
</message>
```

Exemple. *On possède 4 utilisateurs Pierre, Paul, Jean, Jacques. Seuls Pierre et Paul sont connectés. Pierre envoie le message "Hello world" associé au circuit "< seq >< par >Paul, Jean< /par >, Jacques< /seq >". Paul et Jean reçoivent le message simultanément. Tant que les deux ne l'ont pas validé, Jacques ne recevra rien. Paul valide, rien ne se passe, plus tard Jean valide et cette fois le message passe à Jacques.*