UNIVERSITE CATHOLIQUE DE LOUVAIN

FACULTE DES SCIENCES APPLIQUEES

DEPARTEMENT D'INGENIERIE INFORMATIQUE

Développement d'un éditeur graphique de workflow générant automatiquement ses spécifications fonctionnelles

Promoteur : Professeur J. VANDERDONCKT

Mémoire présenté en vue
de l'obtention du grade
de licencié en informatique

par
Christophe LEMAIGRE

Louvain-La-Neuve
Année académique 2006-2007

Remerciements

Je tiens à remercier les personnes suivantes pour leur contribution dans la réalisation de ce mémoire.

Le Professeur Vanderdonckt, pour m'avoir suivi et aidé lors de la rédaction de ce mémoire. Josefina Guerrero Garcia et Juan Manuel Gonzalez Calleros, qui m'ont encadré tout au long de ce travail. Francisco Montero pour son éditeur IdealXML et l'intégration de celui-ci. Ronald Vanderborght, professeur au secondaire.
Mes parents, grands parents et amis pour leur soutien.

Table des matières

Table des figures

Chapitre 1 Introduction

L’organisation du travail est la discipline qui a pour but l’amélioration du rendement par une approche scientifique. Nombreux sont les domaines qui ont un apport à cette tâche, qu’il s’agisse des sciences humaines, économiques, ou encore informatiques. C'est plus particulièrement sur ce dernier que se centre le présent travail.

La réalisation d’un système de gestion de flux de travail (workflow management system) est l'objet d’un domaine de connaissance spécifique dont nous allons aborder les déterminants dans le présent chapitre.

Nous commencerons par introduire les concepts de base (section 1).
La seconde section aura pour objet de déterminer l’objectif poursuivi dans cette thèse. A partir de là nous établirons les hypothèses délimitant le cadre du travail (section 3). Nous poserons ensuite les jalons d'une solution logicielle répondant au mieux aux contraintes existantes (section 4).

Enfin, une carte de lecture résumant les chapitres et donnant un ordre de lecture possible selon différents types de lecteur fera l’objet de la dernière section.

1.1 Contexte de la thèse

1.1.1 Workflow

Les organisations ont des objectifs qu'elles atteignent grâce au travail de leurs membres. Ces objectifs sont divers : financiers, qualitatifs ou encore liés à la productivité. La manière dont le travail est structuré et réparti a une importance capitale sur la qualité de sa performance. Forts de ce constat, nous allons en aborder une version informatisée : les systèmes de gestion de workflow.

De tels systèmes reposent sur l'utilisation des processus business. Un processus étant un ensemble de tâches reliées entre elles pour en définir l'ordre. Lorsque nous ajoutons à cela la notion de ressource (l'entité capable de prendre en charge la réalisation d'une tâche), ainsi que sa gestion, nous obtenons un workflow. La Workflow Management Coalition (WFMC), organisme ayant pour but de standardiser les moyens mis en œuvre pour l'implémentation des workflows en entreprise, définit cette notion de workflow de la manière suivante [Wfmc2] : « support ou automatisation informatisée d'un processus business, en tout ou en partie ».

Afin de clairement délimiter d'une part la définition du processus, des ressources et de leur attribution, et d'autre part l'utilisation dynamique de ces informations, nous limiterons l'utilisation du terme « workflow » à la première partie. Le « système de gestion de workflow », aura quant à lui pour rôle de gérer et exécuter l'utilisation du workflow.

La représentation d'un workflow consiste en une forme de cartographie des tâches. Plusieurs chemins peuvent y être empruntés pour l'objet de travail qui y cheminera. Ces objets peuvent être des dossiers, des personnes, ou n'importe quelle entité nécessitant de réaliser un ensemble de tâches. La figure 1 permet de cerner graphiquement la notion de workflow.

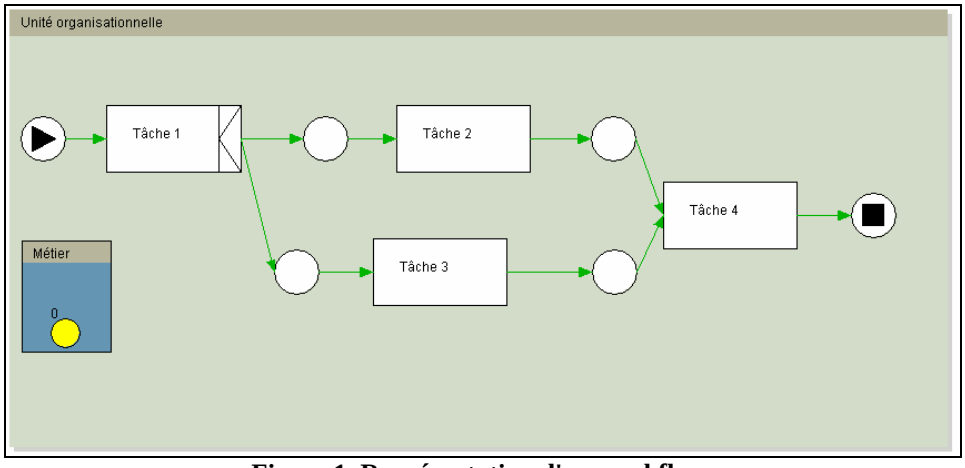


Figure 1. Représentation d'un workflow.

Historiquement, si la place de l'informatique au sein des entreprises a connu un essor important au cours des dernières années, son utilisation dans le cadre spécifique de la gestion du travail aura été plus tardive. Auparavant, la règle d'or était « *En premier lieu l'organisation, en second lieu l'informatisation.* » [Vda1] Désormais l'organisation et le suivi du travail sont des compétences déléguées à l'informatique, réduisant le gouffre entre les organisations et leur système informatique.

Des outils tels que Staffware, Cosa, Windows Workflow Foundation, pour ne citer que les plus connus, sont des exemples de systèmes de gestion de workflow qui sont utilisés par un grand nombre d'organisations.

Les entreprises se trouvant actuellement sur le marché ont bien compris les avantages que l'utilisation des systèmes de gestion de workflow pouvait leur apporter : gains de productivité, allocation plus efficace des ressources, maîtrise de la complexité inhérentes à de grands projets, … Dès lors l'investissement réalisé en conséquence peut être modélisé sur la figure 2 [Cons].

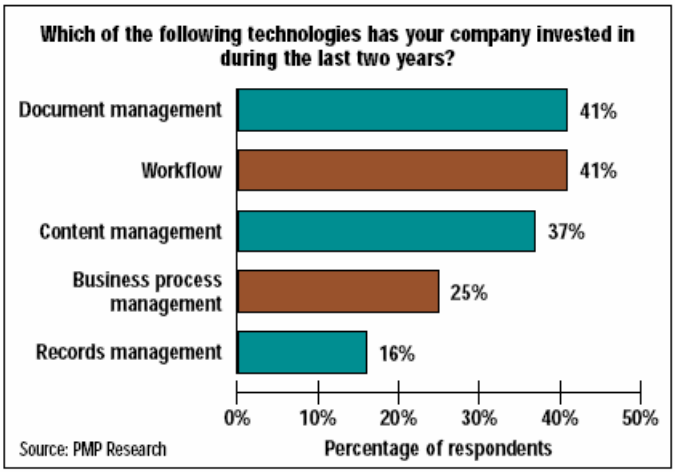


Figure 2. Investissement des entreprises en matière de technologie.

Mais si elle offre des avantages importants, l'utilisation actuelle des moyens informatiques dans le cadre de la gestion de workflow souffre également d'un certain nombre de limitations. Dans une étude réalisée par la Workflow Management Coalition [Cons], il ressort que « (...) *many companies are unhappy with their current mix of software applications and accept that their business processes are subject to constant changes.* » Il est donc important d'identifier les problèmes auxquels sont confrontées les entreprises afin de déterminer quelles seront les améliorations et corrections futures.

1.1.2 Problématique spécifique au niveau logiciel

La réalisation d'un programme (phase de conception comprise) dans le cadre des workflows confronte son auteur à une série de problèmes propres à l'informatisation. Nous allons ici les identifier.

- **Manque de cohérence dans l'approche des workflows**
  Un premier manque qui se fait sentir lorsque l'on est confronté au domaine est l'absence de normes, de lignes directrices claires sur la manière de mener à bien un projet de spécification de workflow. C'est ce que dénonce [Veyb], lorsqu'il suggère de «*Proposer un moyen commun assurant la cohérence en terme d'objectifs d'organisation de processus, de valeurs et des outils est indispensable, car la mise en place de la gestion des connaissances est rarement systématisée.* » Si nous nous référons à l'état de l'art de ce même travail (chapitre 2 section 1), nous pouvons prendre deux cas extrêmes. StaffWare répond au standard architectural du WFMC et est orienté processus, tandis que ActionWorkflow utilise une approche originale se démarquant fortement de ses concurrents, orientée sur la coordination.

- **Indépendance des programmes existants**
  Lorsque nous nous intéressons à un niveau plus concret de l'implication logicielle des workflows, nous avons alors affaire à des problèmes informatiques classiques. Le développement d'outils par des entreprises différentes pose la question de l'utilisation des standards. Il est en effet actuellement souvent impossible de partager des spécifications de workflow, en tout ou en partie, d'un logiciel à l'autre.

  C'est la critique que souligne la Workflow Managagement Coalition [Wfmc1] :
  "The availability of a wide range of products within the market has allowed individual product vendors to focus on particular functional capabilities and users have adopted particular products to meet specific application needs. However, there are, as yet, no standards defined to enable different WFM products to work together, which is resulting in incompatible "islands" of process automation."

  Cette indépendance entre les programmes devrait s'effacer dans le but de pouvoir réutiliser le travail investi dans la modélisation à l'aide d'un logiciel en particulier. Pour cela il est nécessaire d'arriver à des consensus pour l'output des programmes, et notamment le format de sauvegarde.

- **Absence de standard de modélisation**
  Une convergence dans la manière de spécifier un workflow est également nécessaire, afin de ne plus avoir affaire à une série de modélisations isolées se recoupant sur certains points et divergents sur d'autres. Cet aspect est également une cause de l'existence de programmes totalement indépendants, obligeant l'utilisateur à une forme de fidélité logicielle. Le choix d'un programme étant donc finalement une donnée déterminante de la manière dont l'utilisateur perçoit la notion de workflow. Notons que les entreprises sont conscientes du problème, c'est ce qu'évoque [Cons] : « *Companies also recognise the importance of industry standards (…)* ».

- **Notion de relations inter-modèles floues**
  En admettant que l'on atteigne un compromis en matière de choix de modélisation, il reste encore à déterminer la manière dont les modèles se coordonnent pour parvenir à la solution finale. Un cas très concret d'application de ce problème est la difficulté à déterminer la frontière entre un modèle de workflow et un modèle de tâches. Le premier a trait à l'agencement des tâches et le second à la décomposition d'une tâche en particulier. A l'origine ces deux modèles n'ont pas été conçus pour se compléter mais leur utilisation conjointe est une force dont on peut profiter. Pour cela il est important d'établir une frontière entre les deux, afin par exemple de ne pas associer une relation temporelle du modèle des tâches à deux tâches appartenant au modèle des processus. Dans ce cas nous aurions un conflit.

1.1.3 Problématique du logiciel au niveau organisationnel

Tout système de gestion de workflow doit faire face aux problèmes liés à son intégration au sein de l'organisation. Une étude interrogeant les entreprises sur ces questions [Cons] a donné les résultats suivants en termes de flexibilité, capacité de réponse, adaptabilité et intégration (figure 3).

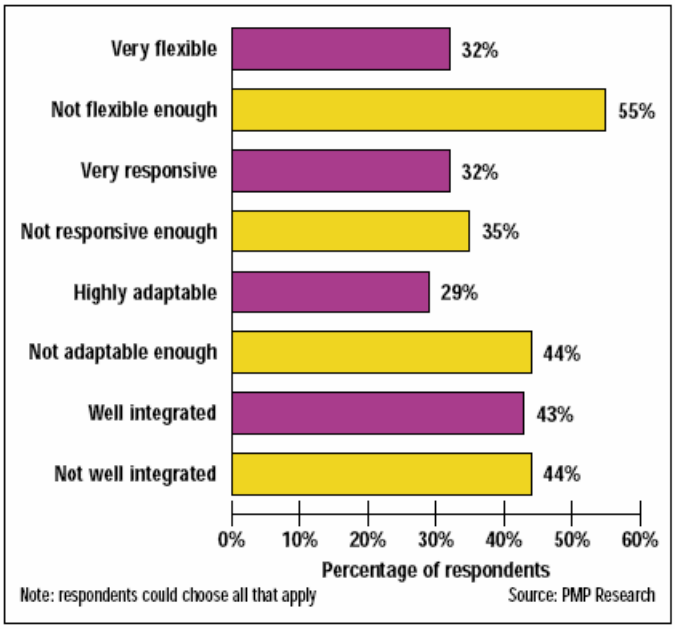


Figure 3. Nature des systèmes informatiques d'entreprise

Il y a donc un taux significatif d'entreprises qui ne sont pas satisfaites de la situation actuelle. Une série de lacunes a été observée sur base d'une étude de la littérature et de l'utilisation de systèmes de workflows actuels. Ces challenges se résument de la manière suivante.

1. **Manque d'intégration de systèmes interactifs individualisés dans le workflow global.**
   En entreprise, il est fréquent d'observer que chaque utilisateur détermine la manière dont il communique les résultats de son travail et l'avancement de celui-ci [Cons]. Le canal de communication n'est donc pas pris en charge par le système de gestion de workflow (pour autant qu'il y en ait un) ou à tout le moins n'est pas géré spécifiquement pour l'ensemble de l'organisation.

2. **Manque d'intégration système à l'échelle locale et globale.**
   Peu d'organisations prennent le pari d'informatiser totalement la répartition du travail à effectuer. Il en résulte les problèmes classiques dus à une gestion humaine de cet aspect organisationnel : les contraintes imposées par un être humain sont plus facilement négociables voire évitables [Guer].

3. **Changements d'activités induits par une intégration technologique inappropriée**
   Idéalement l'adaptation du travail nécessaire à l'implémentation d'un système de gestion de workflow devrait être la plus minime possible. Or cela engendre bien souvent une refonte complète de l'organisation et de ses pratiques [Guer].

4. **Difficulté à obtenir un système de gestion de workflow évoluant avec l'organisation**
   Lorsque l'organisation évolue, il faudrait idéalement que le système de gestion soit capable de prendre en compte ces modifications et s'adapte [Vda1]. Qu'il s'agisse de changements de ressources, de tâches ou dans les processus.

5. **Absence de considération des besoins des groupes**
   Mandvilla & Olfman [Mand] ont déterminé les critères suivants permettant le support du travail en groupe :

- Support de tâches du groupe, du niveau individuel jusqu'au niveau de l'organisation. La compréhension des implications de chacun des sous-groupes formés dans l'organisation est une des clés d'un bon fonctionnement.
- Multiplicité des manières de parvenir à réaliser les tâches. Une redondance dans la manière de réaliser les tâches permet d'être plus souple vis-à-vis des problèmes auxquels peut faire face l'organisation.
- Support de l'évolution du groupe dans le temps. La définition du workflow devrait être modifiée dynamiquement pour tenir compte de l'évolution organisationnelle.

1.1.4 Interfaces hommes-machines

Après avoir abordé les problématiques propres aux logiciels en général et à ceux utilisés au sein d'une organisation en particulier, il nous reste à évoquer celle lié à un dernier aspect : les interfaces homme-machines.

L'interface homme-machine ou interaction humain-machine (IHM) étudie la façon dont les humains interagissent avec les ordinateurs ou entre eux à l'aide d'ordinateurs, ainsi que la façon de concevoir des systèmes informatiques qui soient ergonomiques, c'est-à-dire efficaces, faciles à utiliser ou plus généralement adaptés à leur contexte d'utilisation.

L'utilisation d'un système de gestion de workflow pose obligatoirement la question de savoir quelle information sera donnée et à quelle personne. Bien entendu, il n'est pas souhaitable qu'une intervention humaine soit nécessaire à chaque fois qu'une information doit transiter entre le système et un utilisateur. Au contraire, il faut que le système demande et propage proactivement les informations nécessaires à son bon fonctionnement. Qu'il s'agisse de la spécification du workflow, de l'assignation d'une tâche à un employé ou encore de la confirmation de la terminaison de cette même tâche. A la figure 4, nous pouvons observer un système interagissant avec ses utilisateurs.

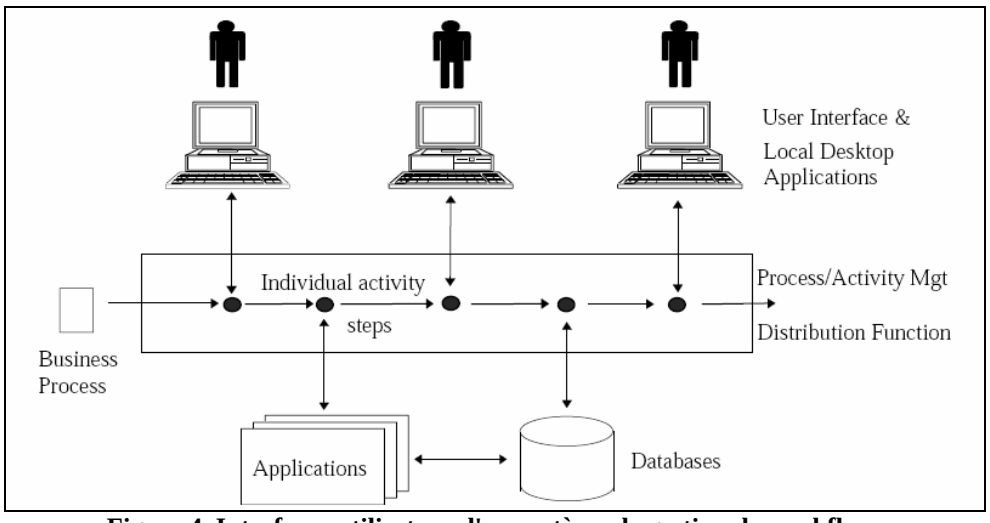


Figure 4. Interfaces utilisateur d'un système de gestion de workflow.

Il est nécessaire de décomposer le programme de telle sorte qu'il respecte harmonieusement les différentes vues que seront la spécification, le suivi du workflow et l'attribution d'une tâche à un utilisateur. Cela engendre des difficultés car il est important d'obtenir un produit qui puisse fonctionner de manière modulaire, chaque partie devant réaliser son travail propre, tout en réalisant en parallèle une coordination de ce travail.

Malgré les difficultés, la motivation derrière cette notion de système d'information interagissant avec les utilisateurs est multiple. La communication et ses formes sont non seulement bien définies (par opposition à un système où l'utilisateur définit lui-même la manière dont il veut communiquer) mais en outre permettent un gain de temps par rapport à une version humainement gérée. De plus l'imposition de contraintes par un système informatisé rend les choses plus transparentes.

1.2 Objectif central du mémoire

Nous énoncerons cet objectif de la manière suivante :

> En vue de mettre en oeuvre un outil de workflow, un éditeur graphique de son comportement devrait permettre la génération aussi automatique que possible de ses spécifications fonctionnelles à partir desquelles une interface homme-machine centrée sur l'utilisateur peut être obtenue.

Etudions plus avant les déterminants de cette définition :

- **L'outil logiciel développé est donc un système de gestion de workflow**
  Nous avons défini en quoi cela consiste et la problématique inhérente aux concepts qui s'y rapportent. Pour donner une ligne directrice, nous voulons donc répartir le travail en groupes de tâches appelés « processus », réaliser la gestion des ressources nécessaires à la réalisation de ce travail et effectuer le suivi des opérations.

- **La forme que prendra cet outil est un éditeur graphique**
  Par opposition à d'autres éditeurs, par exemple sous forme de texte, il s'agit d'un programme dont l'utilisation correspond à la manipulation d'objets graphiques. Cet éditeur doit en premier lieu nous permettre de réaliser la modélisation du workflow. Comme nous le verrons plus loin dans ce document, il s'agira d'une représentation graphique de type Petri Net. Le fonctionnement de l'éditeur sera analogue à celui du logiciel « paint » bien connu des utilisateurs de windows.

- **Génération automatique des spécifications fonctionnelles**
  Il s'agit ici de dériver des informations à partir du workflow construit graphiquement. Les tâches seront définies et ordonnées en utilisant l'éditeur. A partir de cette étape nous obtiendrons une forme de cartographie permettant un fonctionnement dynamique du programme. Il s'agira en quelque sorte d'un trafic représentant l'objet du travail et son cheminement au sein du logiciel.

- **Obtention possible d'une interface homme-machine centrée sur l'utilisateur**
  L'utilisation du format UsiXML rendra possible la génération d'une interface utilisateur. Cette interface permettra de réaliser les tâches définies dans le modèle des tâches.

Cette approche globale de l'implémentation d'un système de gestion de workflow est importante pour la cohésion de l'ensemble. Le système sera évolutif, suffisamment riche pour permettre de travailler au niveau tâche mais également être un véritable système de gestion de workflow réalisant ce que l'on attend d'un tel outil, l'attribution du travail aux ressources notamment.

1.3 Hypothèses

Ce travail repose sur un certain nombre d'hypothèses délimitant plus clairement sa portée. Il s'agit des aspects suivants :

- **Systèmes d'information**

En premier lieu, nous nous sommes focalisés sur les systèmes d'information (IS), par opposition aux logiciels complexes. Ce type de logiciels représente d'ailleurs la majeure partie des logiciels interactifs aujourd'hui.

- **Interface homme machine (IHM)**

Seul l'aspect interface homme machine de ces systèmes d'information est pris en compte. Par exemple, aucun traitement spécifique à l'acheminement des données via un réseau n'a été implémenté. Il ne s'agit pas non plus d'un logiciel réalisant des tâches ou aidant à leur réalisation à proprement parler. Nous nous centrons sur une solution permettant de définir le travail et de le répartir.

- **IHM en version graphique**

Ces IHM ne se déclinent que sous des modalités graphiques, implémentées en java/swing. Il s'agit des GUI (graphical user interfaces). D'autres techniques telles la reconnaissance vocale n'ont pas été explorées.

- **Simulation d'attribution du travail aux ressources**

Le logiciel comprend une gestion de la distribution du travail à des ressources qui permet de simuler un fonctionnement en organisation. Dans cette optique tout a lieu sur un seul ordinateur et une application unique.

1.4 Méthodologie

Pour atteindre l'objectif, une méthodologie est nécessaire. Nous devons autant que possible apporter une réponse aux problèmes évoqués plus haut.

En premier lieu nous nous pencherons sur l'état de l'art (2.1) afin de constater la situation actuelle. Nous dériverons alors une modélisation de workflow ainsi que sa représentation (2.2).

L'approche MDA (model driven architecture) créée par l'Object Management Group, se base sur l'approche MDD (model driven development) consistant à séparer le niveau de l'implémentation de celui de la connaissance. Il y a donc une « séparation nette dans la logique métier de l'entreprise et la logique d'implémentation » [Andr]. En ce sens les différents modèles établis seront donc indépendants d'un langage de programmation, d'une technologie ou d'un logiciel en particulier.

Afin de garantir cette indépendance au niveau logiciel, la possibilité de sauver la spécification du workflow dans le format Usi-XML [Usi] sera offerte. Mais en soi cela ne suffit pas, il faut également se conformer à certains standards. C'est ce que nous ferons en nous pliant à des modélisations répandues, qu'il s'agisse du modèle de tâches, de l'utilisation des Petri nets ou des patterns de ressource [Russ1].

Du point de vue de l'implémentation, le langage java est portable et a l'avantage d'une structure par objets très claire, et donc relativement facile à maintenir et faire évoluer par rapport à d'autres langages. L'API (application programming interface) swing développée pour java permet de réaliser des interfaces graphiques de très bon niveau. L'architecture se conformera au modèle établi par le WFMC que nous étudierons (3.1)

1.5 Carte de lecture

- Chapitre 1 : Introduction
  Le présent chapitre a permis d'introduire les concepts fondamentaux de workflow et de système de gestion de workflow. Nous avons déterminé les enjeux de l'utilisation de ces techniques : les bénéfices apportés mais également la problématique soulevée par l'implémentation d'un tel système. Le cadre du travail a été établi, en posant un certain nombre d'hypothèses de travail.

- Chapitre 2 : Etude du domaine
  Dans cette partie nous nous attacherons à la compréhension des concepts inhérents à la spécification d'un workflow. En premier lieu nous établirons un état de l'art des solutions logicielles existantes. Cela consiste en l'évaluation et la comparaison des programmes qui se trouvent actuellement sur le marché. Leurs avantages et faiblesses seront pris en considération, ainsi que les techniques qui ont été choisies pour la modélisation et les capacités offertes par chacun des logiciels. Nous évoquerons ensuite les modélisations sélectionnées pour la réalisation du logiciel de ce mémoire et les liens entre ces différents modèles. L'ensemble des concepts intervenant dans notre spécification de workflow seront expliqués.

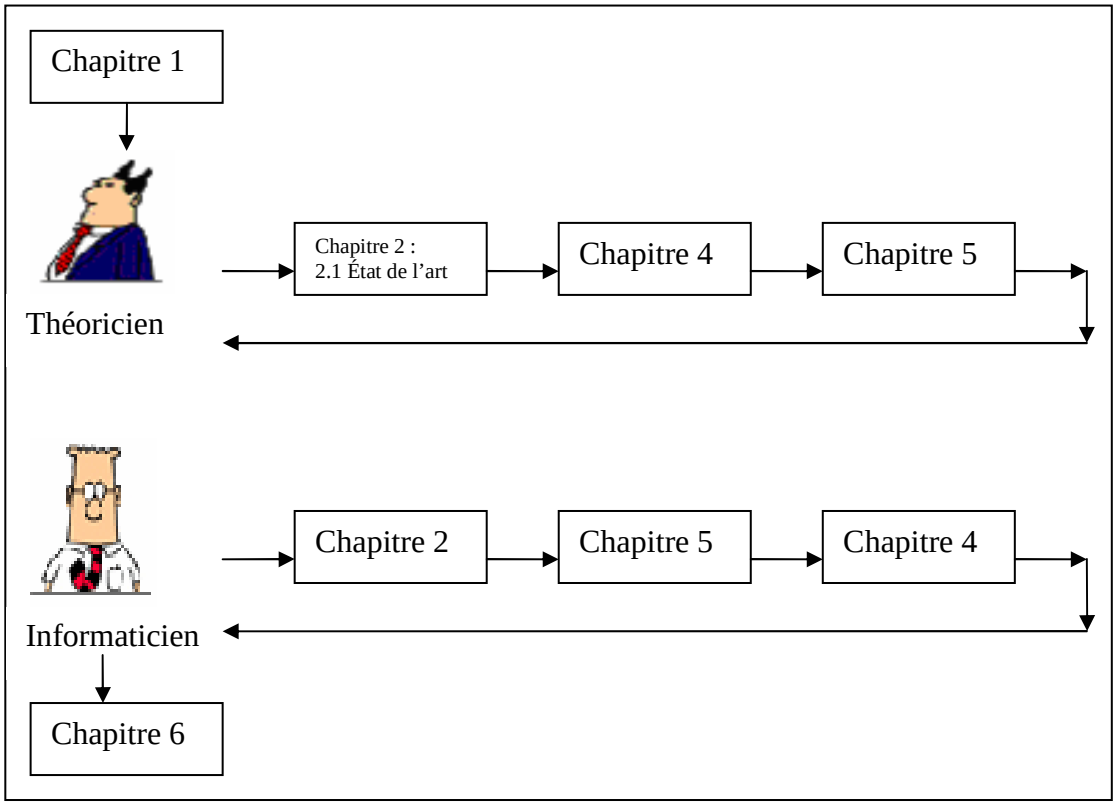
- Chapitre 3 : Implémentation logicielle d'un système de gestion de workflow
  Il sera ici question du programme proprement dit. En premier lieu nous verrons l'architecture proposée par le WFMC et ses déterminants. Ensuite, nous aborderons les caractéristiques (features) offertes à l'utilisateur ainsi que la manière dont il a été découpé en entités distinctes. Nous évoquerons aussi les détails techniques de l'implémentation, par le biais de l'explication du diagramme de classes.

- Chapitre 5 : Etude de cas
  A ce stade nous utiliserons le programme sur un cas concret, illustration de ce que nous obtenons grâce au logiciel. Un workflow sera dessiné, spécifié et les ressources correspondantes seront affectées. Il s'agit de la mise en pratique.

- Chapitre 6 : Conclusions
  Nous évaluerons ici les contributions apportées au domaine. En terme de contributions aux concepts d'abord, et du point de vue de l'implémentation ensuite. Nous évaluerons le produit de ce travail en fonction des objectifs. Enfin, les possibles développements futurs du travail seront évoqués.

Etant donné que chaque personne, selon ses connaissances et ce qu'il attend d'un document, aborde différemment la lecture d'un texte, nous avons établi deux ordres de lecture possibles correspondant à deux stéréotypes de lecteurs :

- Un théoricien, ayant une certaines maîtrise des concepts inhérents au workflow, et dont l'attention se portera plus particulièrement sur la partie logicielle : état de l'art, implémentation et étude de cas.
- Un informaticien, qui va en premier lieu étudier le contenu théorique, puis parcourir le chapitre sur l'étude de cas afin de partir du fonctionnement externe du programme et enfin se plonger dans l'implémentation proprement dite.



Chapitre 2 Etude du domaine

Ce chapitre sera dédié à l'acquisition du support théorique nécessaire à la compréhension d'un système de gestion de workflow. Dans le premier chapitre nous avons défini de quoi il s'agissait, la problématique associée et les objectifs poursuivis dans la réalisation de ce travail. A présent nous abordons les techniques mises en œuvre pour construire notre système.

La première section a pour but de recenser les programmes de système de gestion de workflow existants. Elle mettra en lumière le panel des techniques de modélisations existantes.

La seconde a trait aux techniques que nous utiliserons dans le cadre de la représentation et la modélisation. Nous aborderons donc la définition de la tâche, d'un processus ou encore de ce qu'est une ressource et de la manière dont elle est utilisée dans un workflow.

2.1 Etat de l'art des programmes de spécification de workflow

Un grand nombre de logiciels ayant trait au domaine du workflow se trouvent actuellement sur le marché. Nous nous attacherons donc à présenter un panel représentatif bien que non-exhaustif. Notons également que ces programmes font partie des produits les plus utilisés par les entreprises.

Etant donné que le logiciel développé dans le cadre du mémoire se centre sur l'outil de définition de processus[1], nous ferons de même dans notre approche logicielle. Le premier aspect que nous mettrons en lumière est le type de représentation du workflow utilisée. Nous observerons également la conformité avec le modèle architectural de la Workflow Management Coalition (voir le chapitre consacré à l'architecture WFMC de ce même document).

2.1.1 Staffware

Staffware est un système de gestion de workflow très répandu. Il est produit par la firme du même nom, basée en Angleterre. Son architecture respecte le modèle du WFMC [Vda1]. L'outil de définition de processus de Staffware est le « Graphical Workflow Definer » (GWD, voir figure 4). Il ne possède aucune fonction d'analyse.

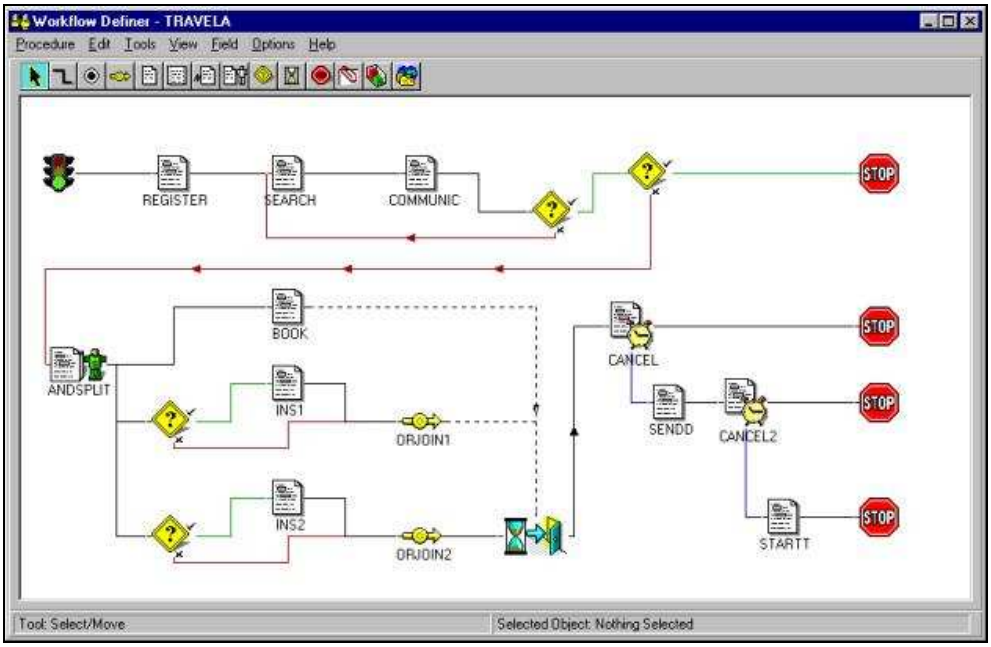


Figure 5. Staffware Graphical Workflow Definer.

Le GWD a l'avantage d'être visuellement très clair. Il repose sur l'utilisation de tâches et d'indicateurs de routage. Les tâches sont de trois types : automatiques (offertes à une application plutôt qu'à un être humain), normales (offertes à une ressource humaine) et évènementielles (déclenchées par un évènement externe). Par défaut le routage consiste en un OR-join/AND-split, ce qui signifie que la réalisation d'une des tâches précédant directement la tâche courante va permettre d'activer cette dernière. Une fois terminée, toutes les tâches destinations de la tâche courante seront mises en route. Les autres patterns de routage que sont AND-join et OR-split nécessitent l'utilisation d'un avatar graphique qui sera inséré dans le schéma. Il n'y a pas de places (ou états) dans la définition d'un processus.

[1] Voir à ce sujet la rubrique sur l'architecture se trouvant dans le chapitre dédié à l'implémentation.

La gestion des ressources fonctionne par groupes d'utilisateurs. Le travail est alors alloué à une des personnes appartenant au groupe déterminé pour l'accomplissement de la tâche. Chaque utilisateur a une liste de travail dans laquelle il choisit le prochain objet de travail qu'il exécutera.

2.1.2 Cosa

Cosa est un système de gestion de workflow qui suit très fidèlement l'architecture proposée par la WFMC. Il a été créé par la société allemande Sofware Ley GmbH. Son outil de définition de processus est le « Cosa Network Editor » (CONE, voir figure 5).

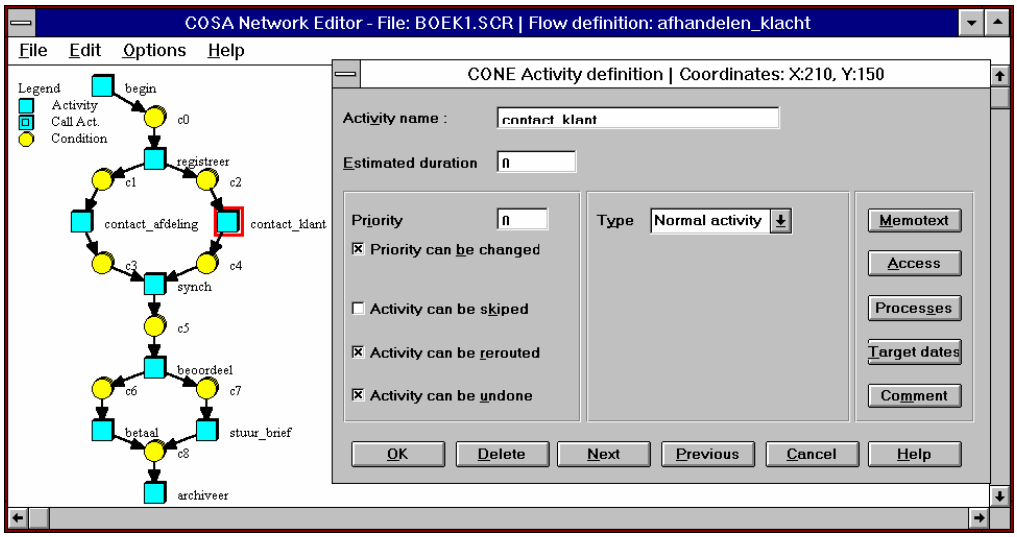


Figure 6. Cosa Network Editor.

La représentation des processus dans le Cosa Network Editor se fait sur base des réseaux de Petri. Cet éditeur offre un grand nombre de possibilités, il s'appuie quatre types de tâches (nommées activités) : normale, commencement, terminaison et appel. La modélisation des ressources est reconnue comme étant l'une des meilleures du marché et permet d'utilisant un grand nombre de règles d'attribution [Vda2].

2.1.3 ActionWorkflow

Le système de gestion de workflow ActionWorkflow est développé par Action Technologies. Il ne répond pas à la norme architecturale du WFMC. L'aspect principal sur lequel se base le système est la coordination. Les processus essentiels sont ceux qui font intervenir la communication entre personnes. Le système se base sur les « Business Process Maps », qui sont fait à partir d'un ensemble de workflows. Chaque workflow passant par trois états : la préparation, la négociation, la performance et la terminaison. La figure 6 en est l'illustration.

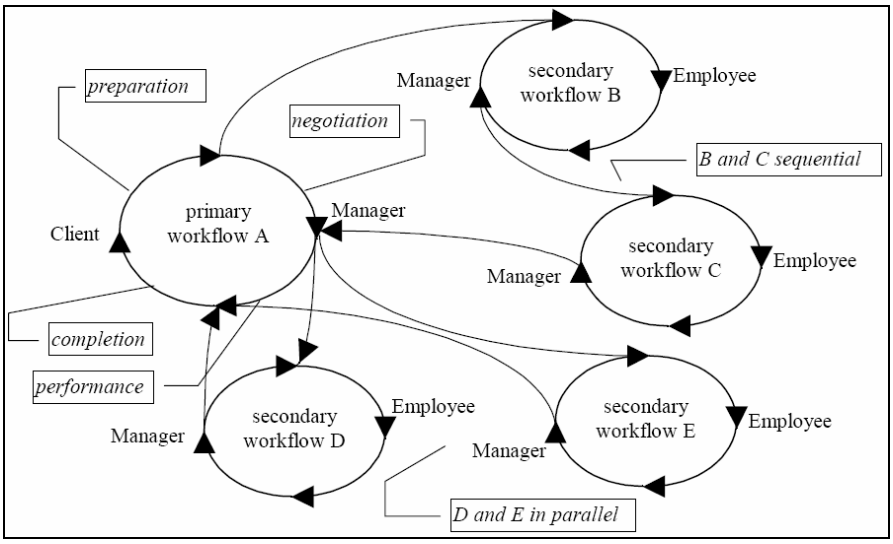


Figure 7. Business Process Map d'ActionWorkflow.

Le routage se fait par l'utilisation des flèches, dans notre cas le worklow C aura lieu après le B, et D et E pourront avoir lieu en parallèle car aucune flèche ne les lie directement. L'outil de définition de processus est ici le ActionWorkflow Process Builder (APB, figure 7).

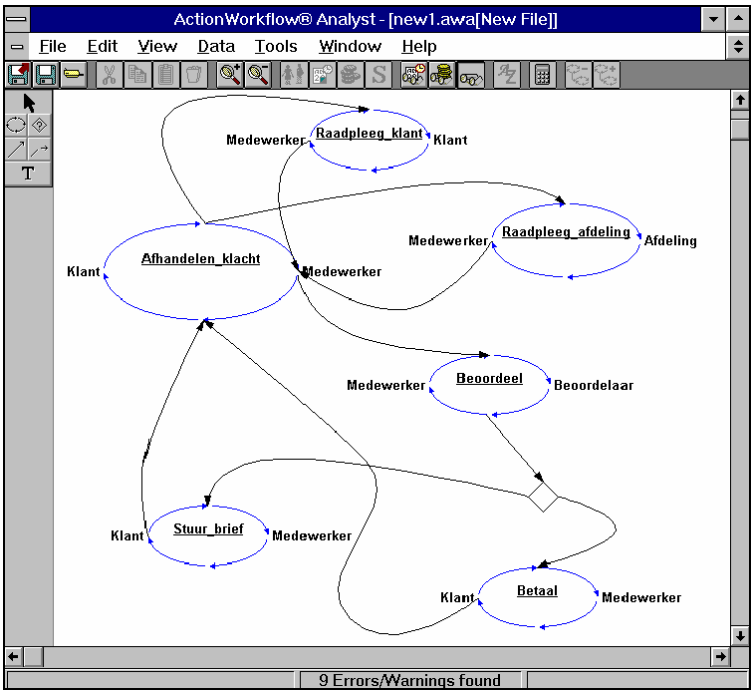


Figure 8. ActionWorkflow Process Builder, Analyst version.

Il se décline en deux parties : une version Analyst pour le designer et une version Developer pour la réalisation effective.

2.1.4 Windows Workflow Foundation

Développé par Microsoft, cet outil permet de créer et spécifier un workflow. Il se base sur une modélisation propre à Microsoft, ce qui ne cadre pas avec la volonté de la WFMC d'établir des standards. Windows Workflow Foundation est un composant de la nouvelle architecture WinFX. Il fournissent une plateforme de développement (un "framework") utilisée pour

concevoir des workflows à destination des applications Windows. Ces applications dépendent du fait qu'elles fassent interagir des personnes ou des applications. Pour cela, Windows Workflow Foundation définit deux types de workflows afin de prendre en compte les spécificités des différents types d'acteurs. En premier lieu les workflows de type séquentiel utilisés pour les workflows faisant intervenir des applications et dont le fonctionnement est prédictible. En second lieu les workflows de type "machine à états" pour les workflows faisant intervenir des personnes et dont le fonctionnement est régi par leur comportement et les actions que ces personnes réalisent.

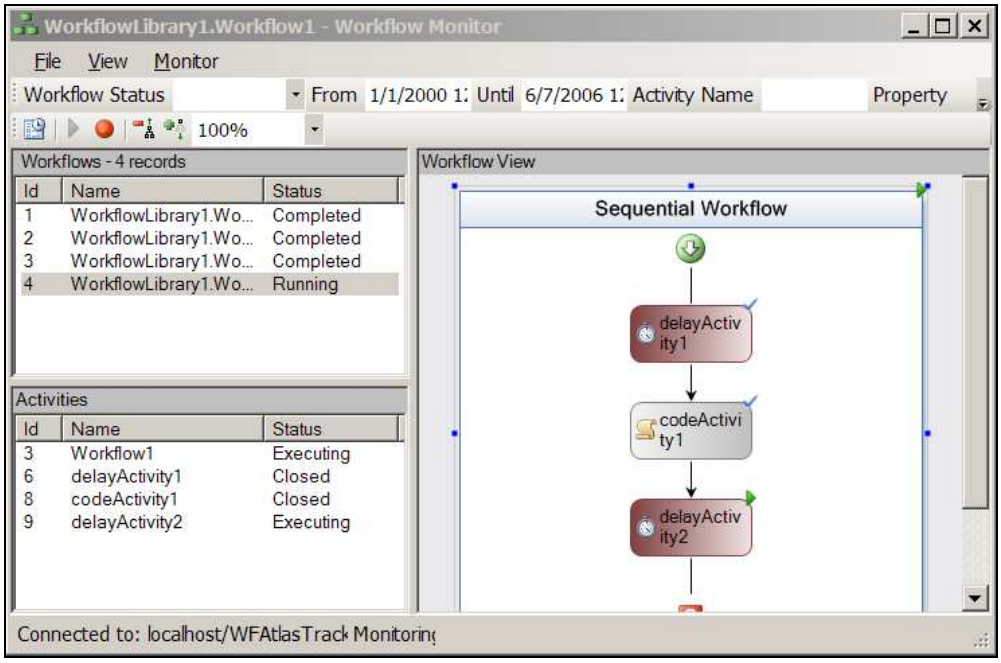


Figure 9. Windows Workflow Foundation, outil de définition de processus.

2.1.5 Flexo

La firme à l'origine de ce programme est la société belge Denali. Il s'agit d'un système de gestion de workflow utilisant les réseaux de Petri pour sa modélisation. Le directeur de Denali explique les motivations à l'origine du logiciel : *«L'idée de base est d'améliorer le travail collaboratif par le biais de l'informatique. (…)Nous avons mis au point une méthodologie de développement d'applications informatiques qui permet d'éviter que des fossés ne se créent entre les utilisateurs de ces applications et les développeurs et les consultants qui travaillent dessus. »* Flexo n'est pas réalisé selon le modèle architectural du WFMC.

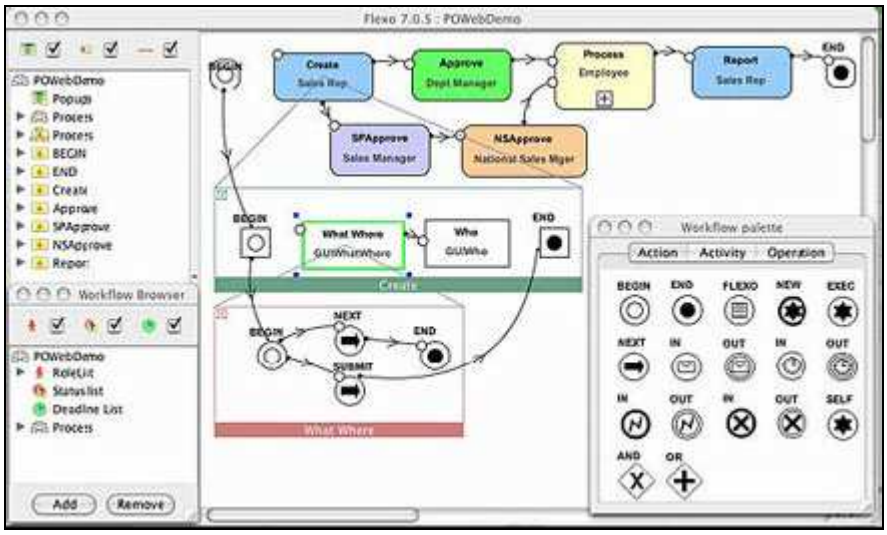


Figure 10. Flexo.

Flexo est orienté prototype et permet par exemple de créer facilement une application web associée à des tâches consistant en l'encodage de données. La représentation utilisée est du type Petri Net.

2.1.6 WebSphere MQ Workflow

Créé par la société IBM, WebSphere MQ Workflow propose des processus d'intégration, prend en charge un grand choix d'interactions humaines et permet de connecter des applications exploitant WebSphere MQ (mise en réseau d'équipements de types différents), les technologies XML et J2EE, les services Web et le logiciel WebSphere Business Integration. Ce logiciel s'appuie sur une architecture standard orientée services. Son moteur de traitement des processus prend en charge les initiatives client e-business : BPM, BAM, EAI, B2B, services Web et intégration de portails.

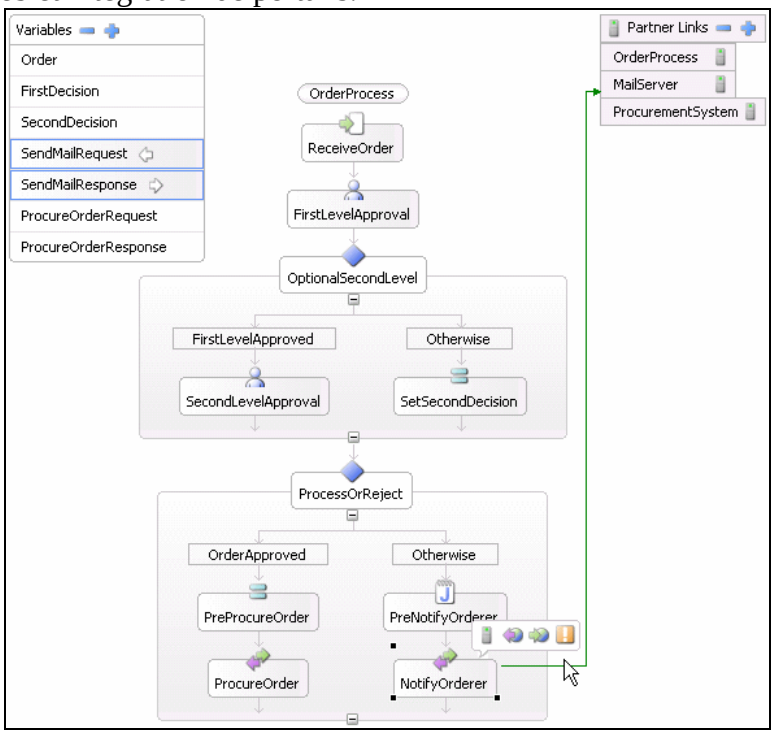


Figure 11. WebSphere MQ Workflow.

WebSphere respecte les standards architecturaux du WFMC.

2.1.7 Arena

Arena est un logiciel permettant de faire de la spécification de workflow et des opérations de simulation dans le but d'améliorer le workflow créé. La modélisation utilisée est celle des diagrammes de flux. Arena permet de créer des animations afin de visualiser l'utilisation dynamique des processus. Les ressources sont également définies dans le logiciel.

Les processus sont créés via un éditeur graphique basé sur l'utilisation de modules. Ces modules représentent des processus et des patterns. Des formes de modules différentes sont utilisées pour représenter les différents patterns de routage.

- Create – Le début du flux de processus. C'est là que commencent les entités de simulation.
- Dispose – Fin du flux de processus. Les entités atteignant ce point lors de la simulation sont retirées du workflow.
- Process – une tâche, généralement réalisée par une ressource unique et nécessitant un certain temps.
- Decide – Un choix de routage est effectué, l'une des branches est prise.
- Batch – Un certain nombre d'entités doivent être collectées avant de continuer
- Separate – Dupliquer les entités pour un routage parallèle.
- Assign – Changer la valeur de certains paramètres pendant la simulation, telles que le type de l'entité.
- Record – Collecter des statistiques telles que le nombre d'entités ou le temps de cycle.

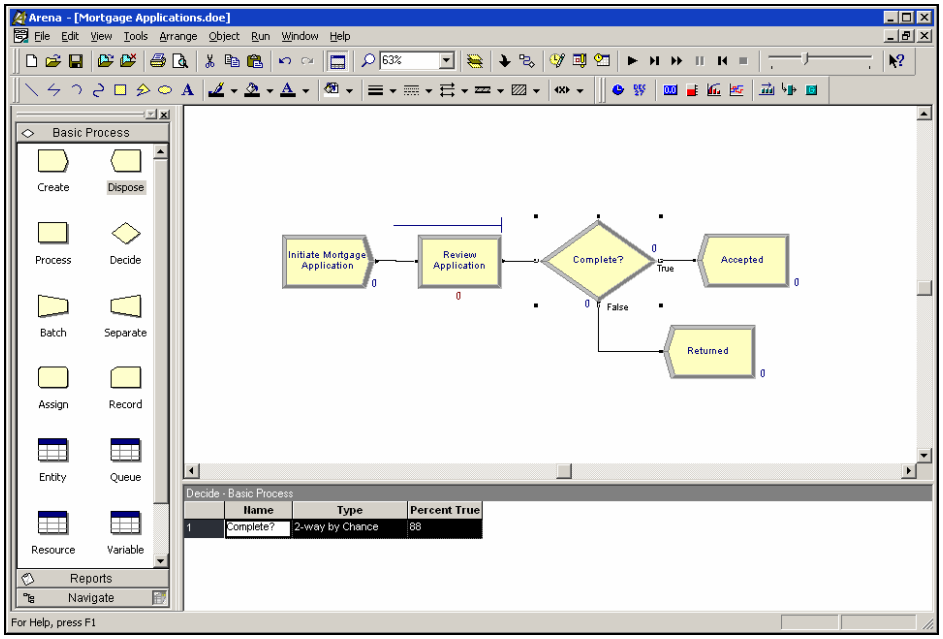


Figure 12. Arena.

Arena ne suit pas l'architecture WFMC.

2.1.8 Flower

Le système de gestion de workflow FLOWer est un outil développé par Pallas Athena. Il est data-driven, relativement flexible et permet la gestion des exceptions. FLOWer permet l'utilisation d'un grand nombre de patterns propres aux workflows, pour le routage,

l'attribution des ressources et les mécanismes d'exception. En tant qu'outil respectant les normes WFMC la séparation entre la définition des processus et l'exécution du workflow permet des changements en temps réel dans la distribution des ressources. FLOWer possède un outil de définition de processus sous la forme d'un éditeur graphique.

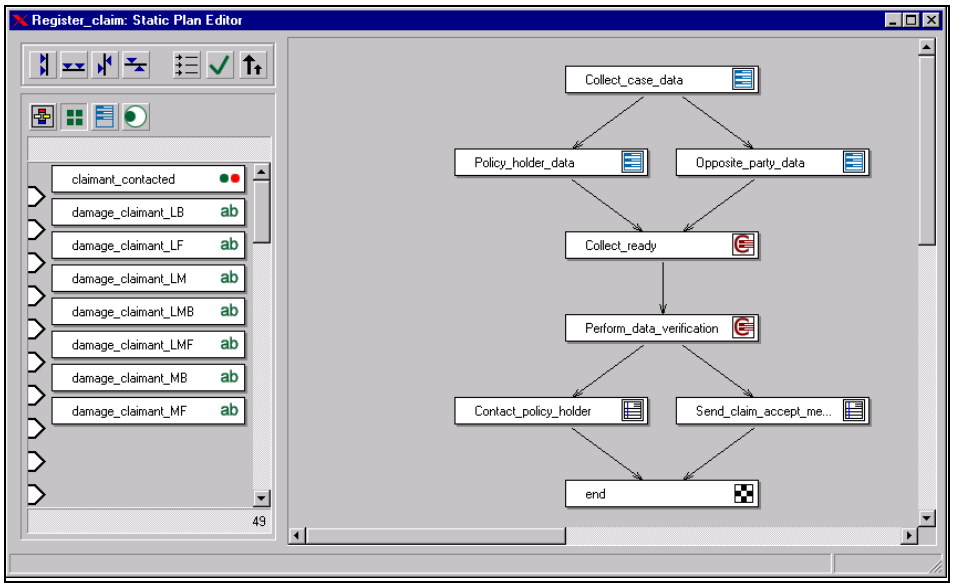


Figure 13. Flower.

Cet éditeur se base sur une structuration en blocs et offre un excellent support aux instances multiples.

2.1.9 IPlanet Integration Platform

IPlanet Integration Platform est une plateforme orientée vers les web services. Elle a été créée par la société Iplanet, fruit d'une alliance entre Sun et Netscape. Le produit se veut être une alternative à la plate-forme de services web proposée par Microsoft .NET. Il se décompose en trois volets, qui s'appuient sur les standards émergeants, tels que XML, Java, et SOAP. Le but est d'avoir une solution prête à l'emploi, mais aussi intégrable dans d'autres solutions qui s'appuient sur les standards du marché.

Le premier volet, iPlanet Integration Server EAI Edition (Enterprise Application Integration), est destiné aux projets d'EAI de grandes entreprises. Il permet la séparation entre la logique métier et la couche applicative. Il est alors possible de définir le processus métier ou de le modifier, sans avoir à toucher aux composants de base des systèmes d'entreprise. L'outil de définition de processus est le Process Builder.

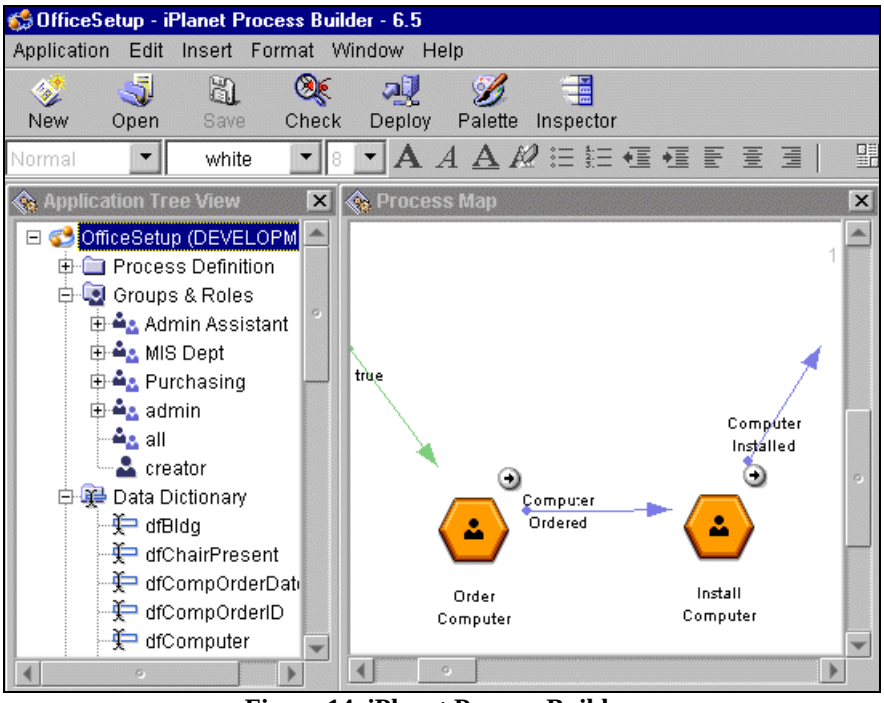


Figure 14. iPlanet Process Builder.

La notion de base d'un schéma processus est l'étape (step). C'est le nom donné à l'élément graphique, qui peut être de l'un des types suivants :

- Point d'entrée. Il s'agit de l'endroit où seront créés les cas.
- Activité utilisateur. Une étape du processus nécessitant la performance d'une tâche.
- Activité automatique, ne requérant pas l'implication d'un utilisateur.
- Sous-processus. Cela permet d'inclure un processus entier à l'intérieur d'une étape.
- Manager d'exception. Une étape permettant à l'administrateur d'intervenir manuellement en cas d'erreur au temps d'exécution.
- Point de décision, il s'agit d'un routage de type conditionnel.
- Split-join. Cette étape permet de modéliser le routage parallèle.
- Point de sortie, indiquant la fin du processus.
- Activité personnalisée. Une étape est connectée à des composants ou services externes.
- Notification. Indique qu'un e-mail sera envoyé dès que l'exécution de la tâche a lieu.

2.1.10 SAP Business Workflow

SAP est un programme de planning des ressources d'entreprise (ERP). Il répond à la norme architecturale établie par la WFMC. SAP Business Workflow est utilisé pour définir des processus business qui ne sont pas encore intégrés dans le système actuel. Ces processus peuvent être de niveau simple tels que des procédures d'approbation ou plus conséquents tels des processus de gestion des ressources et des départements associés. Le logiciel peut également être utilisé pour modéliser des processus qui prendront le relais lors de la gestion des erreurs. Il s'intègre complètement à l'application SAP et aux normes qui lui sont propres. Il se base sur l'outil de définition de processus Workflow Builder.

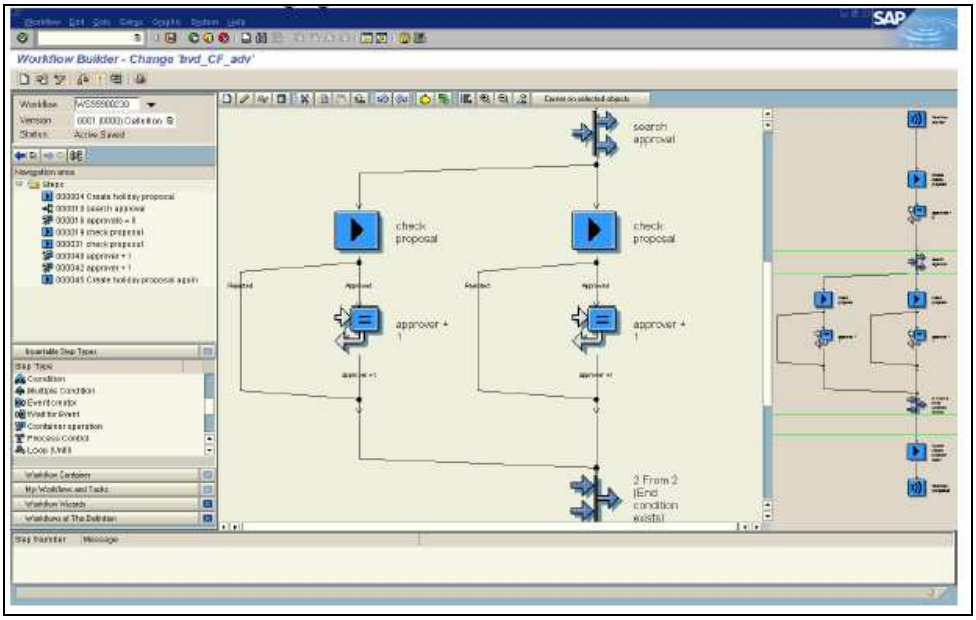


Figure 15. SAP Workflow Builder.

Workflow Builder est un logiciel ne nécessitant pas de connaissances en matière de scripts BPEL (business process execution language), ce qui facilite la prise en main pour les gens ne maîtrisant pas ces aspects sous SAP. L'éditeur permet de définir des rôles, d'assigner des tâches et action utilisateur via une interface basée sur l'utilisation de formulaires.

2.1.11 Enseignements

Nous pouvons désormais résumer les deux aspects observés sur l'ensemble des programmes, la conformité à la norme architecturale WFMC et la représentation utilisée dans l'éditeur de processus.

Programme	StaffWare	Cosa	ActionWorkflow	Windows Workflow F.
Architecture WFMC	Oui	Oui	Non	Non
Représentation de l'éditeur de processus	Propre	Petri Net	Business process map	Séquence / Machine à état

Flexo	WebSphere	Arena	ActionWorkflow	iPlanet
Non	Oui	Non	Oui	Non
Petri Net	Diagramme d'activité	Modules	Blocs	Etape (step)

Sap Business Workflow
Oui
Petri Net

Cet état de l'art a mis en lumière le manque de standardisation en matière de spécification, mais également d'architecture de système de gestion de worklow. La problématique évoquée dans le premier chapitre saute ici aux yeux : manque de cohérence dans la manière de concevoir, spécifier et implémenter un workflow (et le système qui le supporte), ainsi que forte indépendance des outils. Il est donc important de tout mettre en œuvre pour solutionner cet état de fait. Cette volonté est à l'origine de la section suivante, dédiée à la représentation du workflow que nous retiendrons dans le cadre de notre implémentation logicielle.

2.2 Représentation du workflow retenue

2.2.1 Niveau tâches

2.2.1.1 Définition de la tâche

La tâche est une unité logique de travail, qui sera prise en charge en tant que telle par la ressource chargée de son exécution. Ce niveau permet donc de se centrer sur l'individu en charge du travail.

Pour aller plus loin nous reprendrons ici la définition de [Vdd], la tâche étant définie de la manière suivante :

Tâche : traitement possédant une unité spatio-temporelle d'exécution, dans une cellule d'activités disposant des mêmes ressources.

Expliquons plus avant les termes de la définition et leurs implications. Lorsque l'on souhaite répartir le travail, notamment au niveau des ressources, il nous faut réaliser une décomposition de manière à obtenir des entités distinctes. Ces entités sont les tâches.

Cette fragmentation permet en réalité de maîtriser la complexité. Imaginons que nous ayons affaire au processus de construction d'un véhicule dans une usine. L'organisation du travail, autrement dit "qui fait quoi, et à quel moment", devra être d'une granularité telle que l'on puisse assigner une ressource ou un ensemble de ressources à chaque tâche. Et cela sans ambiguïté, car énoncer l'ensemble des moyens humains, mécaniques et informatiques de l'usine précitée en se contentant d'expliquer que chaque tâche sera effectuée par une "partie des ressources" serait une formulation beaucoup trop vague.

La notion de tâche est également apparentée à une unité temporelle. Autrement dit, la manière dont la tâche sera effectuée dans le temps a un impact sur la répartition du travail qui sera effectuée. Une interruption, un point d'attente permettant la prise de décision, la continuité temporelle et la périodicité sont autant de critères pouvant mener à identifier des tâches [Vdd].

La notion de lieu a aussi son importance. Il peut s'agir d'un bureau ou d'une agence par exemple. Un changement de lieu est un critère permettant d'identifier une tâche. Dans les modèles qui ont été retenus afin de réaliser l'application de ce travail, cette notion de lieu est représentée par l'unité organisationnelle. Notons qu'un lieu se voit doté d'un certain nombre de ressources, comme cela est expliqué dans la définition d'unité organisationnelle présente dans ce même travail.

Tout changement des 4 unités (même temps, même lieu, mêmes ressources, même unité organisationnelle) induit un changement de tâche.

2.2.1.2 Types de tâches

Il existe quatre types de tâches, indiquant les moyens mis en oeuvre pour la réalisation du travail.

1. Manuelle : cette tâche sera exécutée par une personne humaine.
2. Interactive : un utilisateur réalisera la tâche à l'aide d'un système informatique.
3. Automatique : le système informatique se chargera de la tâche.
4. Mécanique : une machine (non-informatique) se verra assigner le travail.

Type / Resource	Humain	Ordinateur	Machine
Manuelle	x		
Interactive	x	x	
Automatique		x	
Mécanique			x

2.2.1.3 Cycle de vie

La notion de cycle de vie d'une tâche est l'ensemble des états par lesquels cette tâche pourra transiter, depuis sa création jusqu'à sa terminaison. Les différents états sont :

- Créée (created) : une instance de la tâche a été créée.
- Offerte (offered) : une ou plusieurs ressources ont reçu la proposition de travail.
- Allouée (allocated) : une resource est désignée pour réaliser le travail.
- Commencée (started) : la réalisation de la tâche commence.
- En cours (in course): la tâche est en cours de réalisation.
- Terminée (completed): la performance de la tâche est terminée.
- Annulée/Echouée (canceled/failed) : la réalisation de la tâche a été arrêtée avant son terme.

Le diagramme suivant est la représentation du cycle de vie étendu de la tâche. Il s'agit d'une machine à états finis. La tâche se trouve lors de sa création dans l'état « created ». Ensuite selon les conditions indiquées par les arcs elle pourra évoluer et se trouver dans l'un des états atteignables par l'état en cours.

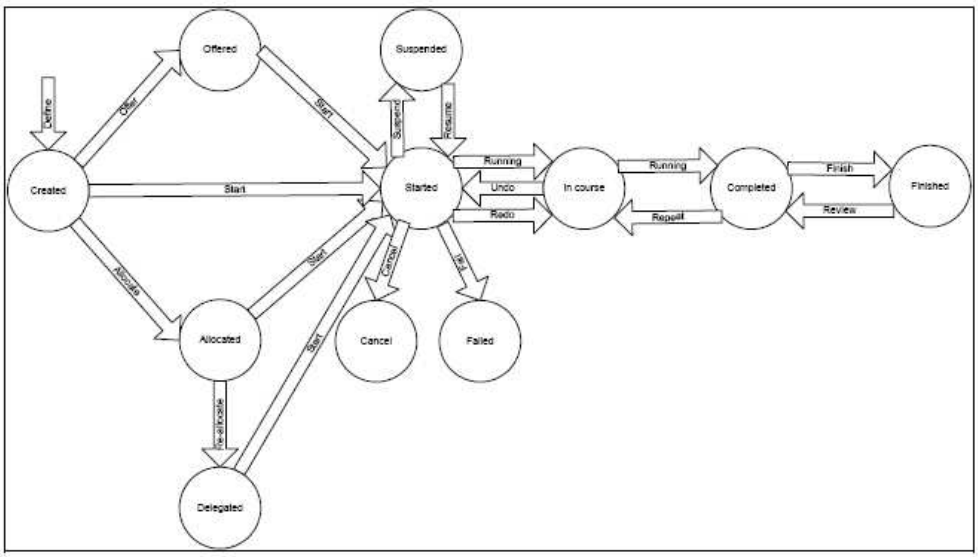


Figure 16. Cycle de vie d'une tâche.

La simplification du schéma permet de se concentrer sur les éléments fondamentaux que sont la création, l'affectation du travail à une ressource et le déclenchement de la tâche. Nous obtenons alors la représentation suivante :

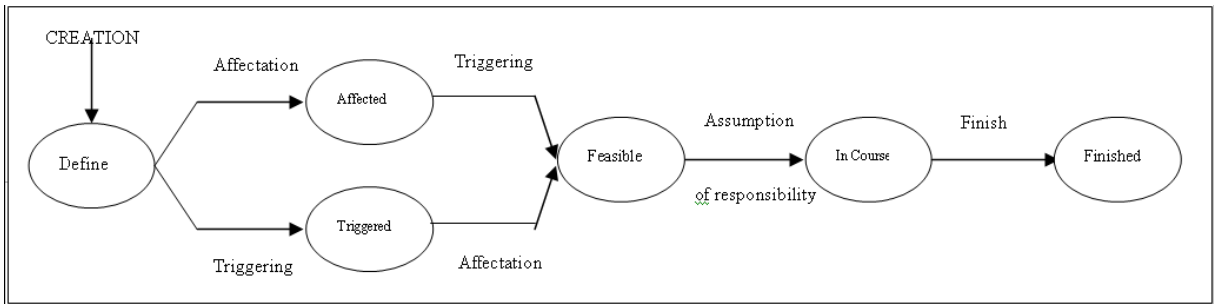


Figure 17. Cycle de vie simplifié de la tâche.

2.2.1.4 Arbre de décomposition

Pour obtenir un niveau de granularité plus fin nous voudrions pouvoir décomposer une tâche en sous-tâches. Pour cela nous avons recours à l'utilisation d'un arbre, dont les noeuds sont des tâches et les fils les sous-tâches nécessaires à la réalisation de la tâche parente.

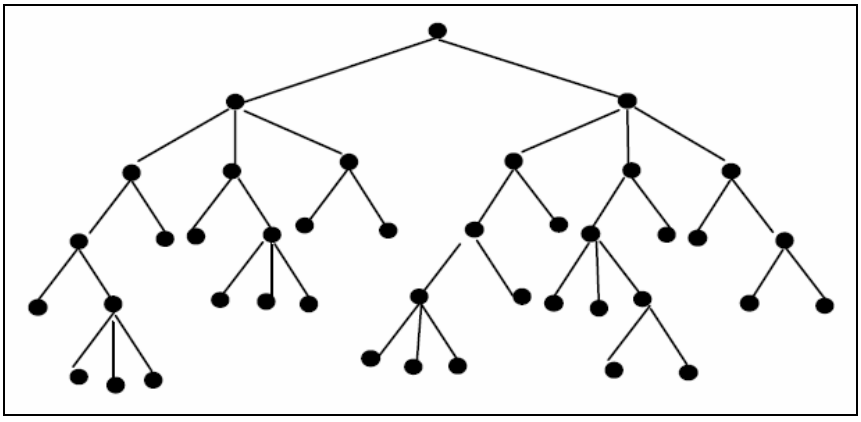


Figure 18. Arbre de décomposition de la tâche.

2.2.1.5 Opérateurs temporels

La réalisation sous-tâches permet de connaître en détails les opérations nécessaires pour la réalisation du travail. Il reste alors à déterminer l'ordre dans lequel les tâches seront réalisées. Les opérateurs temporels LOTOS sont utilisés à cette fin. La notation consiste à lier deux tâches par un trait qui sera orné de l'opérateur temporel associé. Nous obtenons alors une représentation du type de la figure suivante.

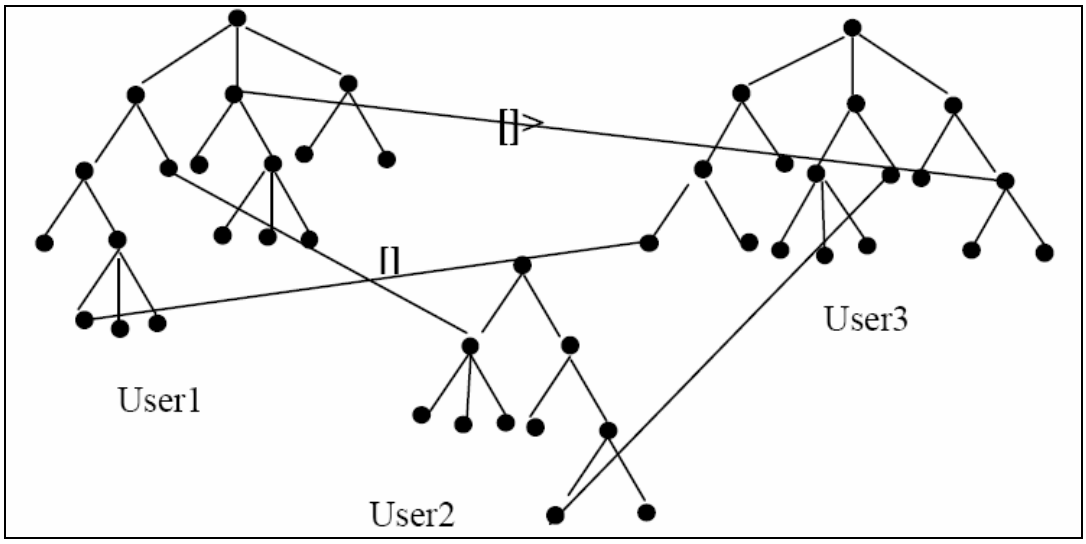


Figure 19. Relations temporelles entre tâches.

Soient deux tâches T1 et T2, les opérateurs temporels binaires sont les suivants :

- Activation (T1 >> T2)
  La première tâche enclenchera la seconde, une fois que cette première tâche est arrivée à son terme.
- Activation avec passage d'information (T1 []>> T2)
  L'activation de la seconde tâche est assortie d'une information que la première tâche possède.
- Suspension/continuation (T1 |> T2)
  La seconde tâche a la possibilité d'interrompre T1 afin de s'exécuter. Lorsque T2 est terminée elle envoie un signal de continuation à la première tâche, qui reprendra le travail là où elle s'était arrêtée.
- Choix (T1 [] T2)
  T1 ou T2 sera réalisée en fonction du choix de l'utilisateur. La réalisation d'une tâche empêche la réalisation de la seconde.
- Désactivation (T1 [> T2)
  Lorsque la seconde tâche se met en route elle envoie un signal à la première, lui signifiant qu'elle est désactivée. T1 est donc arrêtée par T2.
- Indépendance de l'ordre (T1 |=| T2)
  T1 ou T2 peut être commencée en premier lieu. Par contre lorsque l'une des deux tâches est commencée l'autre doit attendre sa terminaison avant de commencer. En définitive nous avons soit T1 suivie de T2 soit T2 suivie de T1.
- Concurrence (T1 ||| T2)
  Les deux tâches peuvent être réalisées sans aucune contrainte. Soit être réalisées en même temps (ne fut-ce que partiellement), soit strictement l'une après l'autre.

- Concurrence avec passage d'information (T1 |[x]| T2)
  La concurrence nécessite ici certains échanges d'information lorsque le travail des deux tâches sera en cours.

Lorsqu'une seule tâche est concernée par l'opérateur temporel nous avons les relations réflexives suivantes.

- Itération (T*)
  Lorsque la tâche a été réalisée, nous avons alors une autre occurrence de la même tâche qui débute.
- Itération finie (T{n})
  Dans ce cas de figure l'utilisateur connaît à l'avance le nombre d'itérations qui seront nécessaires, ce nombre étant n.
- Optionnelle ([T])
  Cet opérateur permet de déclarer que la réalisation d'une tâche est optionnelle et pourrait donc ne pas avoir lieu.
- Récursion (T)

  Ce procédé consiste à nécessiter la réalisation d'une tâche dans une autre occurrence de la même tâche. Nous pouvons dire que « la tâche s'appelle elle-même ».

2.2.1.6 Choix du modèle

Ordonnancement et décomposition sont les motivations au coeur de la modélisation des tâches. Un état de l'art des modèles de tâches a recensé les modélisations suivantes : ConcurTaskTree (CTT), User Action Notation (UAN), Goals, Operators, Methods, Selection rules (GOMS), Hierarchical Task Analysis (HTA), Groupware Task Analysis (GTA) et Diane+.

Le modèle choisi afin de déterminer l'agencement des tâches est l'arbre des tâches concurrentes (ConcurTaskTree). Il a la particularité de présenter à la fois une décomposition de la tâche mais également les relations temporelles inter-tâches grâce aux opérateurs LOTOS.

En outre, les tâches sont réparties selon les quatre catégories (user, abstract, interactive et application) définies précédemment. Ce qui mène à la représentation suivante (figure 20).

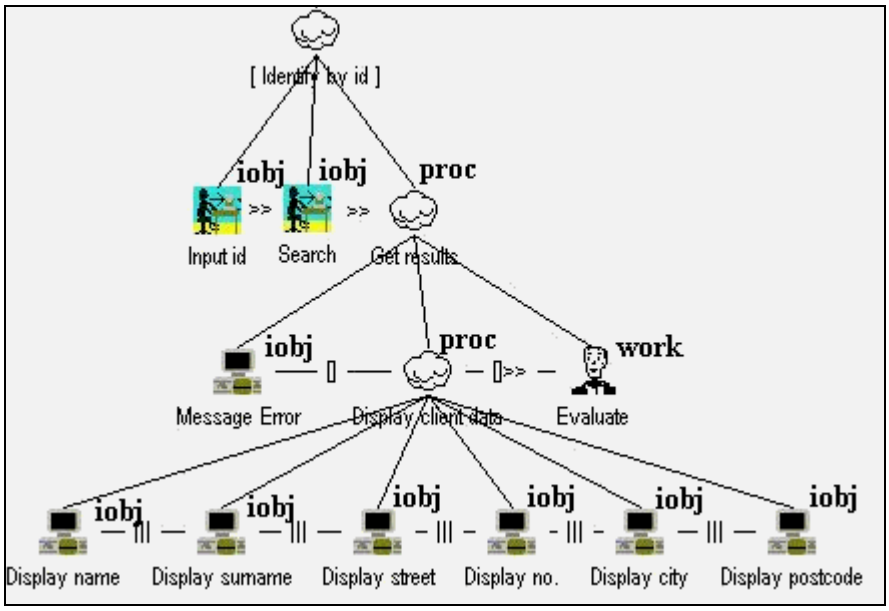


Figure 20. Arbre de tâche concurrent.

2.2.1.7 Modèle des tâches

Ce modèle permet de sauver l'information inhérente aux tâches. Le diagramme suivant [Guer1] contient l'information structurée sous forme de classes.

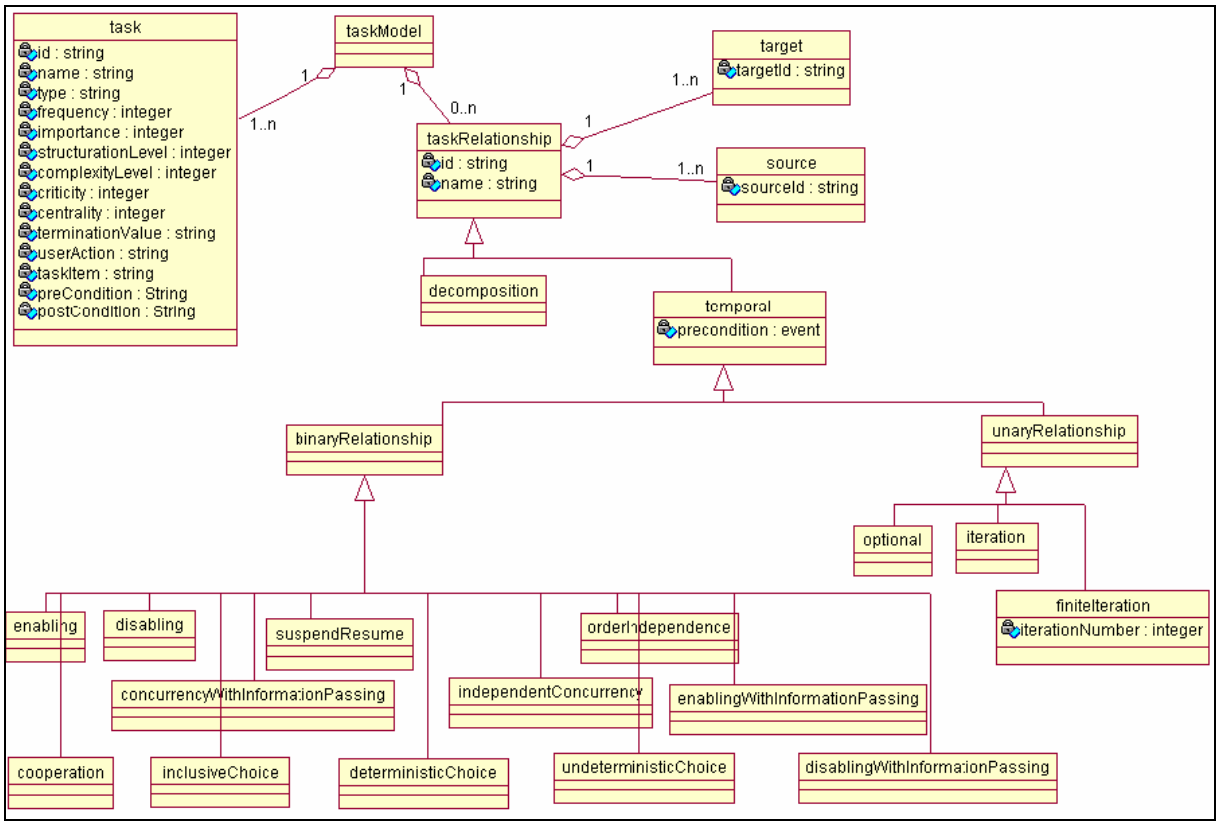


Figure 21. Modèle conceptuel des tâches.

Le modèle des tâches (taskModel) est fait de tâches et de relations entre les tâches. Une relation entre tâches (taskRelationship) est définie comme ayant une ou plusieurs sources ainsi que une ou plusieurs cibles. Deux types de relations existent : décomposition ou

temporelle. Dans ce dernier cas les relations binaires et unaires sont développées.

2.2.2 Niveau processus

Un niveau au dessus de la tâche se trouve le niveau processus (process). Il permet d'organiser les tâches en donnant la structure générale, il s'agit donc du niveau de la coordination. A ce stade nous déterminons l'ordre dans lequel nous réaliserons les tâches. Cet ordonnancement donne tout son sens à la notion de flux de travail (work flow).

La WFMC définit la notion de processus (également appelé « business process ») de la manière suivante [Wfmc2].

> Processus : ensemble de tâches ou procédures qui prises collectivement concourent à la réalisation des objectifs business, dans le contexte d'une structure organisationnelle définissant les rôles et relations fonctionnelles.

Nous allons maintenant aborder la catégorisation des processus et leur représentation.

2.2.2.1 Trois catégories de processus

- Processus primaires (production) :

Ce sont les des processus produisant les produits ou services d'une compagnie. Il s'agit de l'organisation du travail permettant de créer l'output qui sera vendu au client. Cet output peut être des produits finis tels des véhicules, des services, mais également des éléments moins tangibles tels qu'une expertise ou l'élaboration du design d'un nouveau produit.
Input : ordres et moyens de production
Output : produits et services

- Processus secondaires (support) :

Il s'agit du maintien des moyens de production, qui consiste en l'achat et la maintenance des machines et véhicules utilisés dans le cadre de l'activité de l'entreprise. C'est à ce niveau également que se trouve la gestion du personnel, qu'il s'agisse de l'embauche, la sélection, la formation, l'évaluation du travail et le versement des salaires. Enfin, on citera la gestion financière et comptable ainsi que le travail du département marketing.
Input : financement pour l'achat des moyens de production
Output : outil de production fonctionnel

- Processus tertiaire (management) :

Sous cette dénomination se retrouve l'ensemble des processus ayant trait au management d'une société. Autrement dit, la coordination des processus primaires et secondaires, qu'il s'agisse de déterminer leurs objectifs ou les moyens qui seront mis en œuvre pour y arriver. Les processus tertiaires comprennent également la dimension financière de l'entreprise, au travers du contact avec les départements concernés et les actionnaires.
Input : objectifs
Output : profit financier

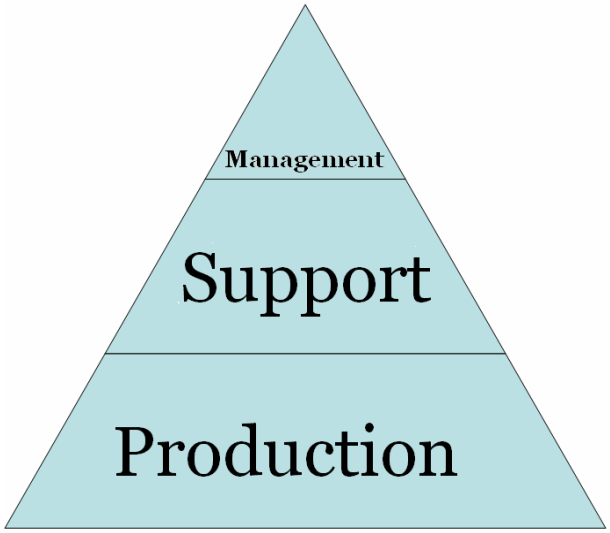


Figure 22. Trois catégories de processus.

2.2.2.2 Réseaux de Petri (Petri nets)

Pour modéliser les processus nous aurons recours à la modélisation appelée « réseaux de Petri ». Cette modélisation possède deux atouts : la visualisation et le formalisme.

Etant donné que la modélisation du travail est une discipline ayant pour but de faire travailler des individus dans un même but, il est important de pouvoir se reposer sur des techniques permettant de partager la vision du travail. En effet, recevoir l'aval des groupes de personnes impliquées dans l'organisation implique que l'on partage un langage commun et des représentations faciles à décoder par tout un chacun. En ce sens, la modélisation offerte par Carl Adam Petri est à la fois extrêmement simple pour permettre de saisir facilement la structure mais également très expressive.

La seconde force des réseaux de Petri est son formalisme mathématique. En effet, les propriétés découlant de leur utilisation sont vérifiables mathématiquement. C'est un aspect d'autant plus important que nous avons recours à des moyens informatiques pour modéliser les processus. En effet, des vérifications sur la correction du schéma sont alors directement implémentables par un programme informatique. C'est notamment le cas du test d'accessibilité (reachability), qui fait partie des caractéristiques du logiciel de ce mémoire. Sa description se trouve plus bas dans ce même document.

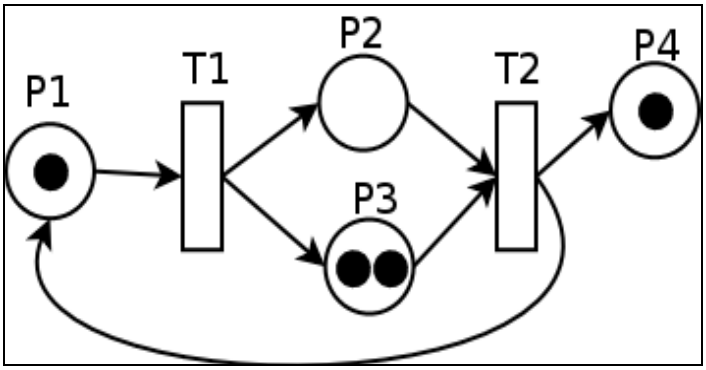


Figure 23. Réseau de Petri.

2.2.2.2.1 Petri nets classiques

La modélisation sous forme de réseaux de Petri repose sur quatre éléments de base : les jetons, transitions, places et arcs.

- Le **jeton** (token) est l'objet réel qui chemine au sein du Petri net. Ce jeton pourrait représenter un patient dans un hôpital ou encore un dossier à l'administration. Selon les conditions qu'il remplira il suivra la route correspondante et recevra les traitements associés à chaque lieu traversé. Il existe en général une multitude de chemins possible entre le point de départ et l'arrivée d'un réseau de Petri.
- Les **transitions**. Il s'agit d'une action qui à un certain moment pourra être « en cours de réalisation», comme c'est le cas de la tâche. Dans ce cas nous avons un processus atomique, autrement dit indivisible. Nous pouvons alors parler sans distinction de transition ou de tâche dans le cadre de l'utilisation du Petri Net.
- Par opposition, les **places** sont un lieu dans lequel aucun traitement ne sera effectué. Il s'agit d'un état stable dans lequel un jeton peut se trouver. A ce stade nous savons que le jeton a parcouru l'un des chemins menant à la place et que sous certaines conditions il pourra atteindre l'une des transitions cibles le cas échéant.
  Il existe trois types de places : commencement (start), terminaison (end) et normale. La place de commencement permet de déterminer le point d'entrée du workflow. C'est là que seront placés les cases qui viennent d'être créés. Quand à la place de terminaison elle permet d'identifier la fin du processus.
- Les **arcs** sont les flèches reliant les transitions et les places. Il s'agit d'arcs dirigés, indiquant que l'on passe d'un élément vers le second. De plus un arc ne relie jamais deux éléments de même type, qu'il s'agisse de deux places ou de deux transitions.

La représentation de ces éléments que sont les jetons, places, transitions et arcs se trouve sur la figure suivante :

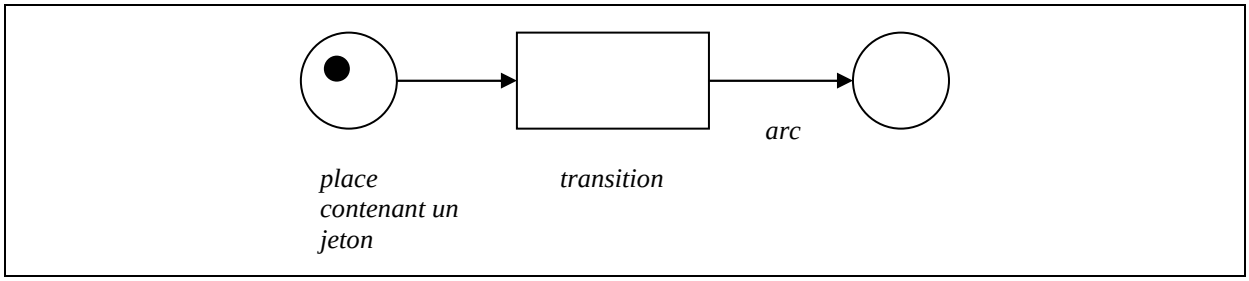


Figure : éléments constitutifs d'un Petri net.

Si nous fixons notre attention sur une transition, on distingue deux types de places qui lui sont liées par des arcs : les places en entrée (input) et les places en sortie (output). Dans notre schéma, l'input se trouve à gauche de la transition. Une flèche part de cet input et atteint la transition. Les places rejointes par une flèche provenant de la transition sont donc les places d'output.

La **mise à feu** (firing) d'une transition est l'action par laquelle les jetons se déplacent dans le schéma. Le jeton est alors retiré de sa place d'input et placé dans la place d'output. Lorsqu'il y a plusieurs places en entrée ou en sortie pour une même transition, le principe est de dire que la mise à feu requiert un jeton par place d'input et générera un jeton par place d'output.

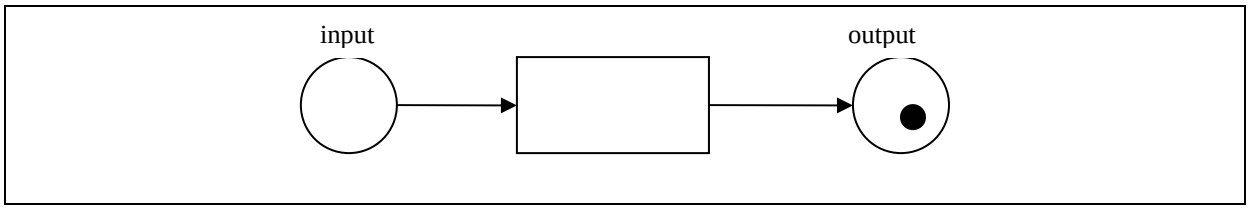


Figure : exemple de mise à feu.

2.2.2.2.2 Petri nets de haut niveau

Il existe trois extensions principales au modèle d'origine. Elles ont pour but de lever certaines limitations des réseaux de Petri classiques.

- Extension de la couleur

La première version des réseaux de Petri ne différenciait pas les jetons. La « colorisation » permet de résoudre ce problème. Cela permet de poser des conditions dépendant des caractéristiques du token : par exemple, la nationalité d'une personne dans un processus de gestion de l'octroi de passeports. En effet, la procédure pourrait être différente selon qu'il s'agit d'une personne venant de l'Union Européenne ou non.

- Extension temporelle

L'ajout d'un cachet temporel (timestamp) permet de procéder à la mise à feu d'une transition en fonction d'un moment à partir duquel il sera possible d'utiliser un jeton. Il s'agit en réalité d'un délai, très utile dans un cadre d'utilisation tel que la modélisation d'un feu de circulation. En effet, passer du rouge au vert ne doit se faire qu'après un nombre déterminé de secondes.

- Extension hiérarchique

Cette extension donne la possibilité de rendre le diagramme plus lisible grâce à l'utilisation des transitions comme conteneur d'un processus, appelé « sous-processus » (subprocess).

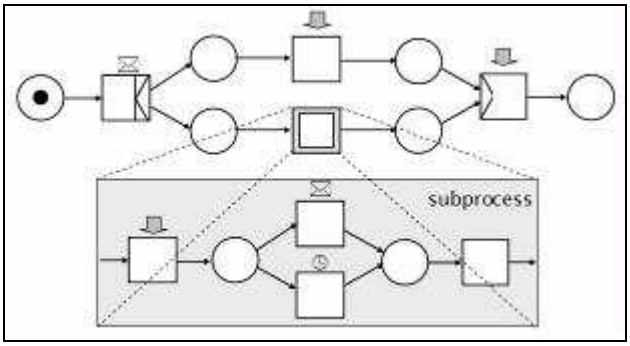


Figure 24. Exemple de sous-processus.

2.2.2.3 Routage (routing)

La notion de routage permet d'obtenir l'ordre des tâches et la sélection de celles qui seront effectuées. On identifie quatre patterns de routage différents.

- Routage séquentiel

Les tâches sont effectuées les unes à la suite des autres. Il s'agit d'une relation d'indépendance entre les tâches.

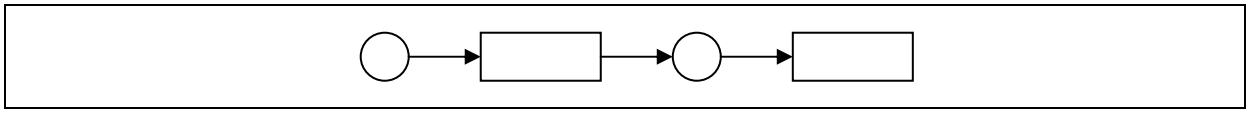


Figure : routage séquentiel.

- Routage parallèle

Il consiste à dire que deux tâches (ou plus) peuvent être réalisées en même temps ou dans n'importe quel ordre. Dans l'exemple ci-dessous il s'agit des tâches 1 et 2.
And-split signifie qu'il y aura autant de jetons générés qu'il y a de places d'output. En l'occurrence, après la mise à feu de t1 permise par le jeton en c1, nous obtiendrons deux jetons identiques. L'un en c2 l'autre en c3.
And-join réalise la jointure de deux jetons identiques se trouvant en c4 et c5, pour finalement obtenir un unique jeton en c6.

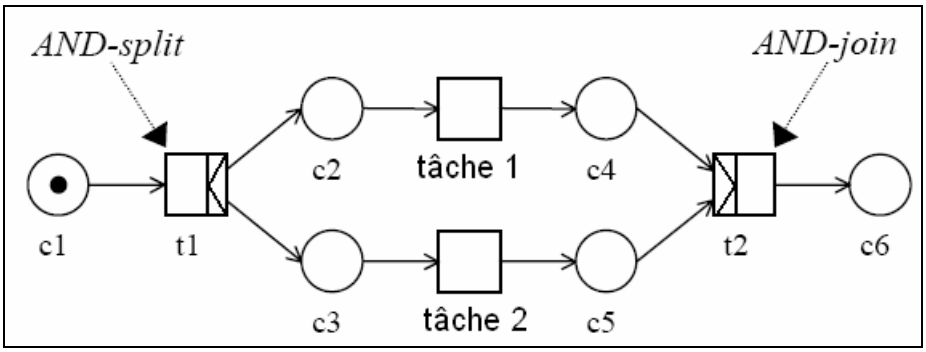


Figure 25. Routage parallèle.

- Routage sélectif

Dans ce cas il s'agit d'une alternative. Un seul jeton sera émis pour les places d'output de la transition t1. Dans le diagramme suivant il s'agira soit de c2 soit de c3. Ce qui aura pour conséquence qu'une seule des tâches sera effectuée parmi la tâche n°1 et la tâche n°2.
Or-join réalise la jointure sélective, un seul jeton se trouvant dans une place d'input suffit à mettre à feu la transition ciblée.
Or-split signifie qu'il y aura un seul jeton généré, dans l'une des places d'output. Sur la figure suivante, après la mise à feu de t1 permise par le jeton en c1, nous obtiendrons soit un jeton en c2 soit un jeton en c3.

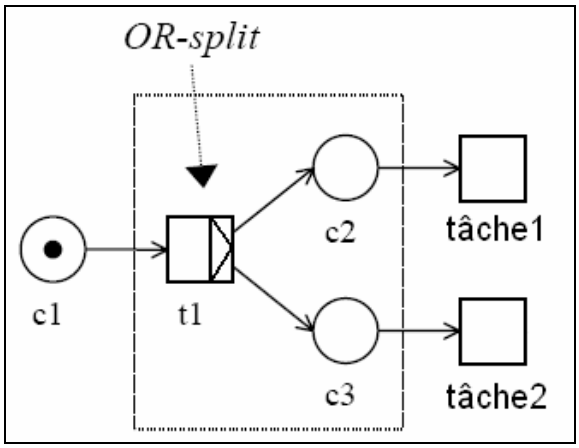


Figure 26. Routage sélectif.

- Routage mixte (And/Or)

Il s'agit d'un cas moins classique, où un choix entre AND et OR sera possible. Après mise à feu du split And/Or il y aura soit

1. un jeton en c2 et aucun en c3
2. un jeton en c3, aucun en c2
3. un jeton en c2 et un en c3

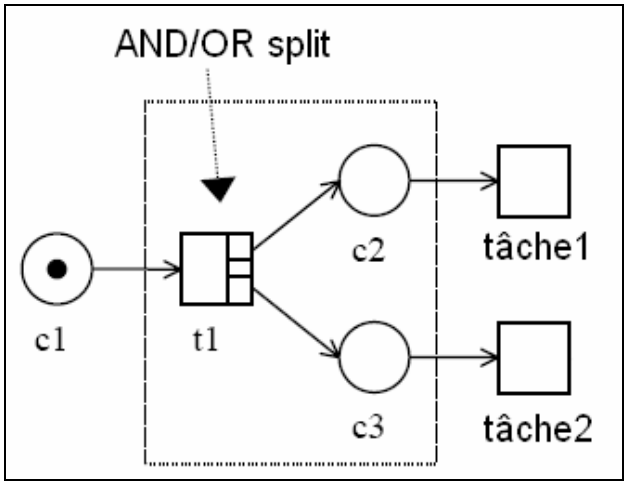


Figure 27. Routage mixte.

Notations de routage

La figure ci-dessous indique à gauche la notation explicite et à droite la signification correspondante avant l'introduction de la notation.

La représentation explicite a été choisie pour le programme réalisé dans ce travail car elle a l'avantage d'être très claire et d'éliminer tout risque d'ambiguïté donc d'erreur. A la gauche de la transition se trouve la condition d'entrée, à droite la condition de sortie. Le triangle contenu par chacune de ces conditions permettant de voir en un coup d'œil qu'il s'agit d'une fusion (join) ou d'une division (split).

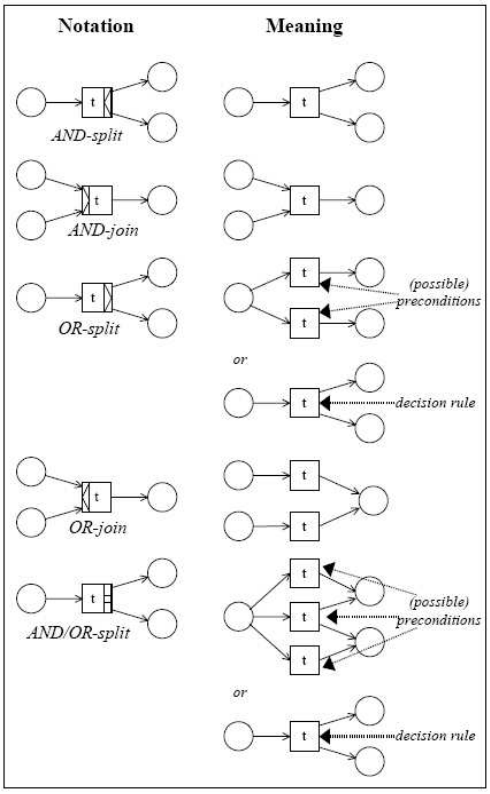


Figure 28. Notation de routage.

Ajoutons que cette notation a pour conséquence d'avoir au plus un arc dont la destination est une place. Il en va de même pour l'arc « sortant » (dont la source est une place) qui sera donc unique. Nous n'obtiendrons donc pas les deux cas de la figure suivante.

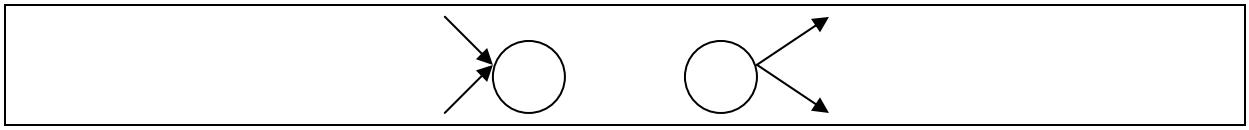


Figure : arcs incorrects.

2.2.2.4 Modèle des processus (process model)

Ce modèle, provenant de [Guer1], comprend les éléments constitutifs du niveau processus ainsi que les relations qui les lient.

Les opérateurs de processus (process operators) sont les patterns de routage que nous venons de définir. Ils lient un processus source à un ou plusieurs processus cibles (target process) dans le cas les cas de séparation (splitting). Pour la jointure (join) nous avons un ou plusieurs processus source qui sont liés au processus cible.

Les différents opérateurs sont les suivants :

- Séquence : les processus sont réalisés les uns après les autres.
- Séparation parallèle : correspond au AND-split.
- Synchronisation : il s'agit du AND-join
- Choix exclusif : OR-split
- Fusion simple (simple merge) : OR-join
- Choix multiple : AND/OR

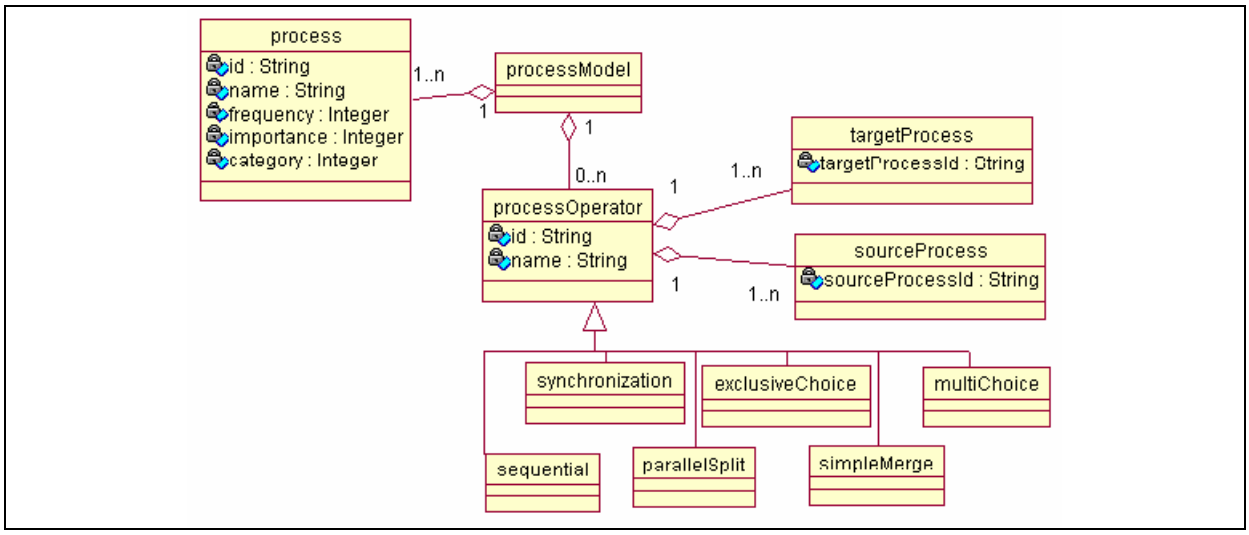


Figure 29. Modèle des processus.

2.2.3 Niveau workflow

La définition d'un workflow est composée de trois parties :

- la définition du processus
- la classification des ressources disponibles
- les règles d'attribution des ressources au travail

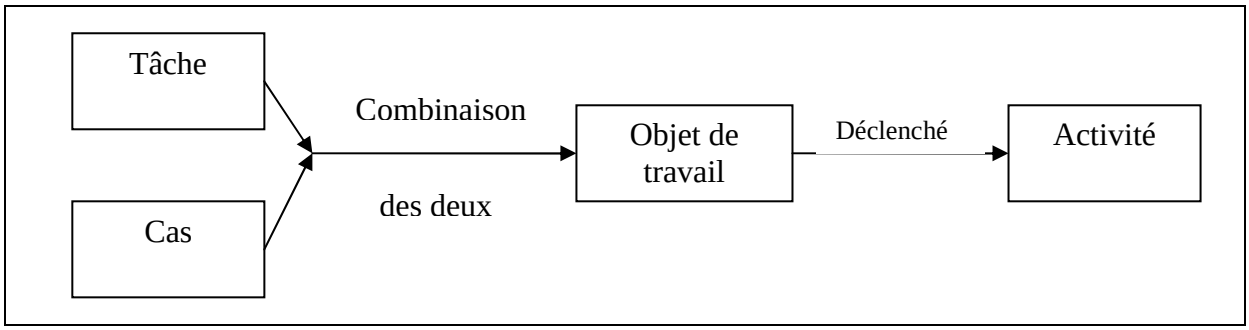
Ayant déjà défini le premier point au niveau processus ainsi que la décomposition des tâches qui lui appartiennent (niveau tâche), il nous reste à aborder les deux autres points. La notion de ressources est donc l'élément clé de cette section. Nous pouvons à présent aborder la dimension de la prise en charge du travail.

2.2.3.1 Le cas (case)

Il s'agit de l'objet sur lequel porte le travail des tâches. Plus concrètement, cela peut être un dossier, un produit ou service qui va aller de tâche en tâche selon l'ordre défini par un processus. Sa représentation est donc le jeton du niveau processus.

La tâche est une définition du travail à réaliser mais non la performance de ce travail. Pour distinguer la tâche de sa réalisation nous avons recours aux notions de d'objet de travail, d'activité et de déclenchement.

- Objet de travail (work item)
  Lorsqu'une tâche peut être réalisée pour un cas et que le travail n'a pas encore commencé, il s'agit d'un objet de travail.
- Activité (activity)
  A partir du moment où le travail est en cours pour ce même cas, la performance de la tâche en cours se nomme une activité.
- Déclenchement (triggering)
  Le passage entre objet de travail et activité ne se fait que lorsque la performance de la tâche a été déclenchée. Ce déclenchement peut émaner de la ressource assignée à la tâche, être dû à un évènement externe ou à un signal temporel.



2.2.3.2 Unité organisationnelle

L'unité organisationnelle est la représentation du lieu dans lequel prennent place les tâches d'un processus. La combinaison du lieu et des ressources qui s'y trouvent donne des informations que nous allons utiliser pour l'attribution du travail aux ressources disponibles.

2.2.3.3 Ressources

La ressource est l'entité capable de réaliser une tâche [Vda1]. Dans une administration il s'agira vraisemblablement d'êtres humains, mais une imprimante ou un robot sont également des ressources. Il existe donc trois types de ressources : travailleur (user stereotype), immatérielle, matérielle. Un travailleur possède un agenda dans lequel sont stockées les tâches qu'il a à accomplir. Nous reviendrons sur cet aspect dans la partie consacrée à l'attribution du travail.

Une ressource est également caractérisée par ses capacités, permettant de savoir quelles sont les tâches qu'elle est capable d'accomplir. Elle appartient à une ou plusieurs unités organisationnelles. Enfin, elle est capable d'effectuer un ou plusieurs jobs (métiers).

La ressource peut donc être vue comme se trouvant dans une union ensembliste reprenant ses jobs et unités organisationnelles. Ci-dessous, le cas d'un travailleur nommé John, combinant les métiers de mathématicien et physicien et appartenant à l'unité scientifique d'une université.

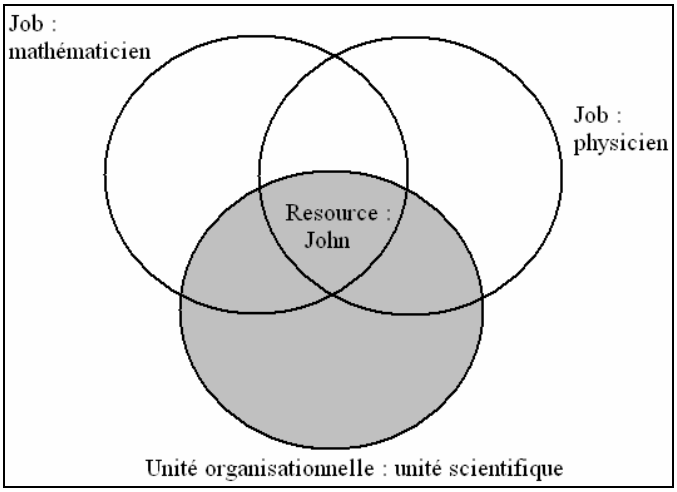


Figure 30. Métiers et unités organisationnelles d'une ressource humaine.

Ajoutons que si le terme rôle est parfois utilisé pour décrire les différents jobs ou métiers, nous lui associerons plutôt une signification propre à l'utilisation du workflow. Les différents rôles sont alors le workflow designer (chargé de la modélisation du workflow), le workflow manager (chargé de son bon fonctionnement) et le process manager (responsable d'un processus en particulier).

2.2.3.4 Attribution du travail aux ressources

L'attribution de la réalisation d'une tâche à un ensemble de ressources se base sur l'utilisation de patterns. Pour visualiser cette attribution, des animations utilisant la technologie flash se trouvent sur [http://www.workflowpatterns.com/]. L'illustration suivante montre l'attribution

de la tâche « open office » à la personne nommée Sue. Il s'agit d'une allocation directe, le nom de la tâche ayant été placé dans le champs « allocated » de la liste de travail (work list) de Sue.

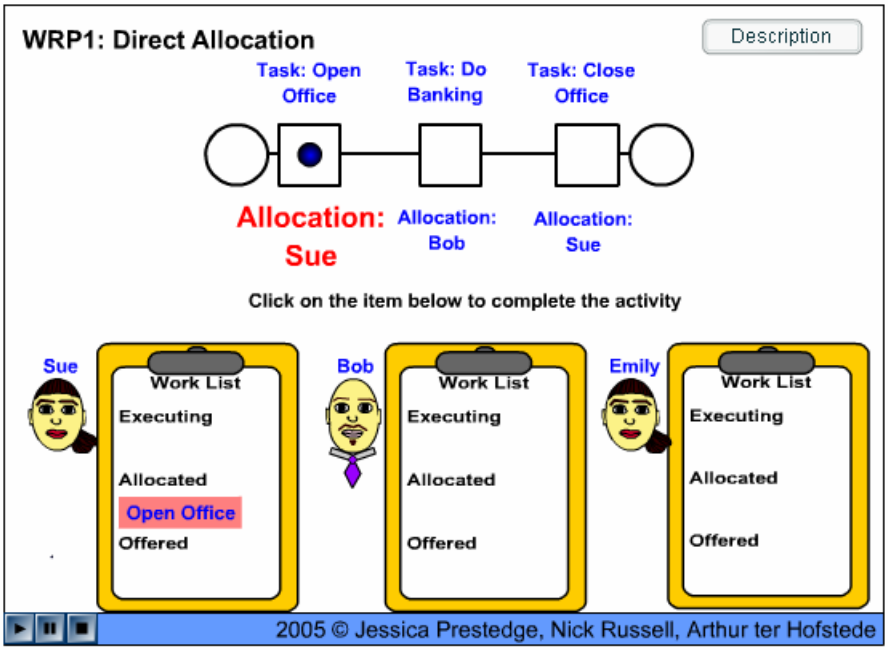


Figure 31. Attribution du travail aux ressources.

La liste de travail (worklist) est donc le moyen par lequel un travailleur est informé des tâches qu'il a à accomplir. Elle est divisée en trois parties : les tâches offertes, allouées et en cours d'exécution.

Nous allons à présent établir la liste des patterns de ressources existants, regroupés par famille afin de faciliter la compréhension.

Patterns de création :
Leur utilisation permet de réduire champs des ressources pouvant effectuer le travail.

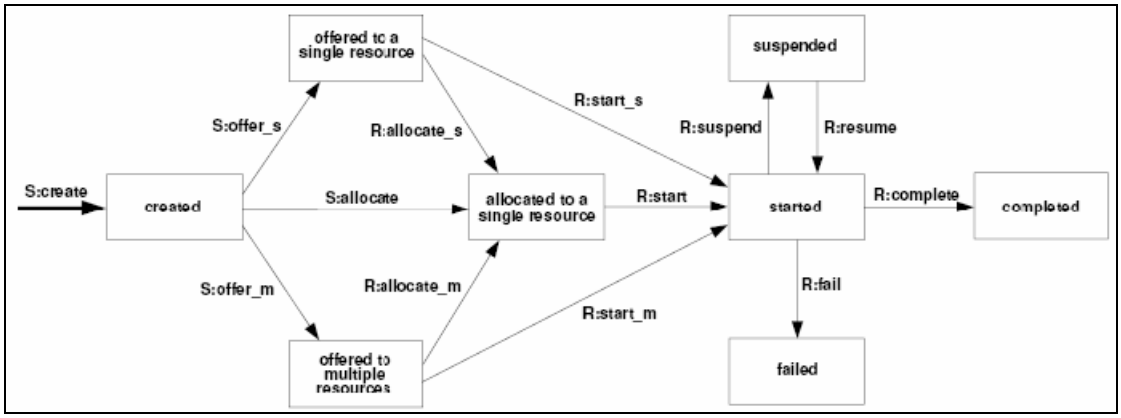


Figure 32. Patterns de création.

- Allocation directe – spécifier au temps de design l'identité de la ressource qui va exécuter la tâche.

- Allocation différée – différer la spécification de l'identité de la ressource qui va exécuter une tâche jusqu'au temps d'exécution.
- Basé sur l'autorisation – spécifier un ensemble de ressources qui sont autorisées à exécuter une tâche.
- Séparation des devoirs – spécifier que deux tâches réalisées sur un même cas doivent être réalisées par des ressources différentes. Cela est utile dans les processus bancaires par exemple, afin d'éviter les fraudes.
- Gestion de cas – allouer toutes les tâches réalisées sur un même case à la même ressource. Si le cas est un dossier à l'administration, il sera pris en charge par la même personne pour l'ensemble des tâches effectuées.
- Retenir les familiers – quand cela est possible, allouer la tâche à une ressource ayant déjà travaillé sur le cas.
- Basé sur la capacité – offrir ou allouer un objet de travail à des ressources sur base de certaines capacités qu'elles ont.
- Basé sur l'historique – offrir ou allouer des objets de travail aux ressources sur base de leur historique.
- Basé sur le niveau hiérarchique – attribuer la réalisation d'une tâche à des ressources vérifiant une condition posée sur leur niveau hiérarchique.

Patterns de distribution de type « push » :
Le système réalise proactivement l'attribution des ressources.

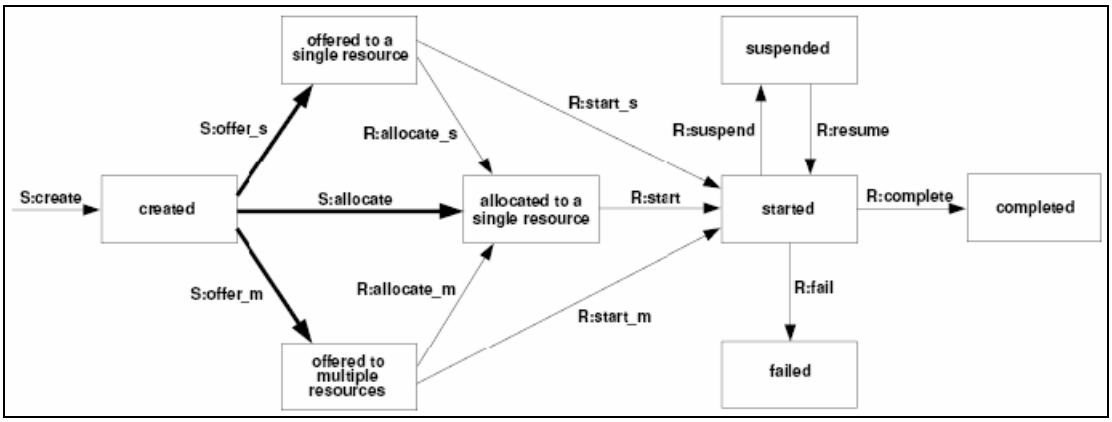


Figure 33. Patterns de distribution de type push.

- Offre à une ressource unique – un objet de travail est proposé à une seule ressource.
- Offre à un ensemble de ressources – l'objet de travail est proposé à plusieurs ressources.
- Allocation à une ressource unique – la performance d'une tâche est attribuée à un travailleur.
- Distribution anticipée – la ressource est avertie avant le moment auquel l'activité a été créée.
- Distribution au temps de démarrage – la ressource est avertie au moment où elle est censée commencer à travailler.
- Distribution tardive – la ressource est avertie après que la performance de la tâche ait commencé.
- Allocation aléatoire – le système attribue le travail aux ressources possibles de manière aléatoire.
- Allocation round-Robin – le système attribue le travail aux ressources selon l'ordre round-Robin.

- Allocation à la file la plus faible – le système alloue le travail à la ressource ayant le plus petit nombre d'objets de travail alloués.

Patterns de distribution de type «pull » :
Il s'agit de la situation où des ressources individuelles sont mise au courant d'objets de travail spécifiques requérant exécution, soit par une offre directe du système soit indirectement via l'utilisation d'une liste de travail partagée.

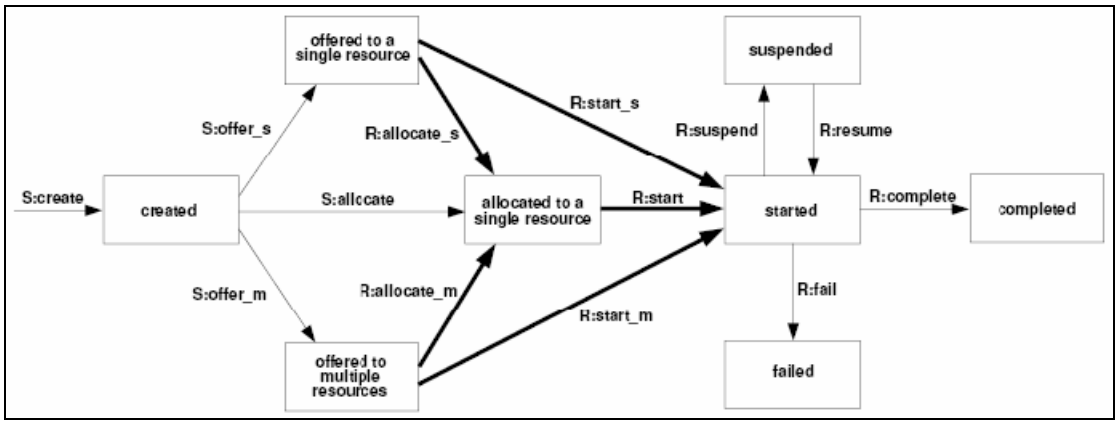


Figure 34. Patterns de distribution de type pull.

- Allocation initiée par la ressource – la ressource prend en charge un objet de travail sans pour autant commencer directement à travailler dessus.
- Exécution initiée par la ressource (allocation) – la ressource commence immédiatement le travail lorsqu'il lui est alloué.
- Exécution initiée par la ressource (offre) - la ressource commence immédiatement le travail lorsqu'il lui est offert.
- File de travail gérée par le système – le système de gestion de workflow organise le contenu et la séquence dans laquelle les objets de travail seront livrés à la ressource.
- File de travail gérée par la ressource.
- Autonomie de sélection – la ressource choisit le prochain objet de travail à exécuter selon ses préférences.

Patterns de détour :
Ces patterns permettent de modéliser la situation où l'allocation du travail est interrompue par le système de gestion de workflow ou à l'instigation de la ressource.

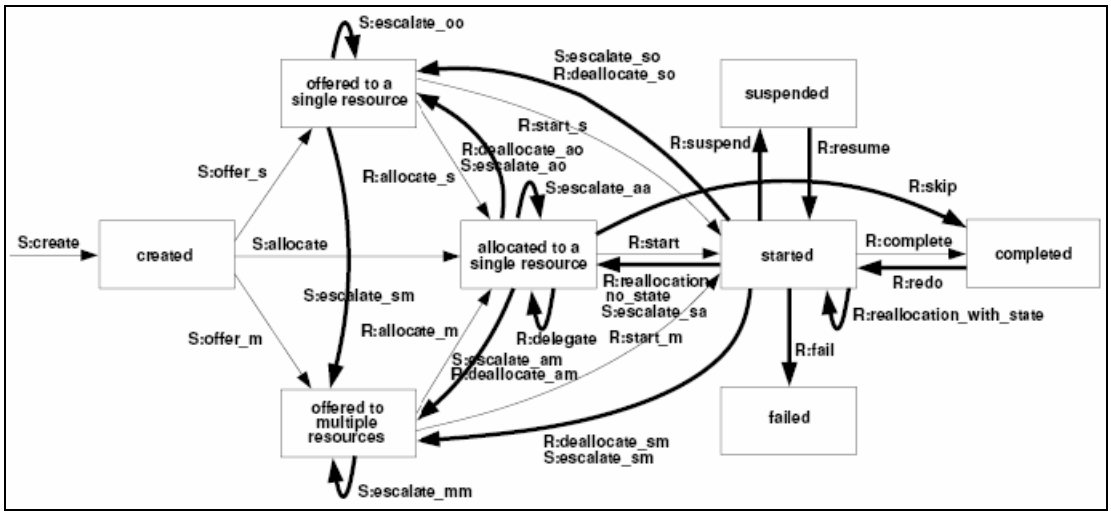


Figure 35. Patterns de détour.

- Délégation – la ressource réalloue un objet de travail qui lui avait été alloué à une autre ressource.
- Escalade – le système de gestion de workflow offre ou alloue un objet de travail à un nouveau groupe de ressources.
- Désallocation – la ressource abandonne un objet de travail qui lui a été alloué. Le travail devra donc être redistribué à d'autres ressources.
- Réallocation avec état – la ressource confie la réalisation d'une tâche à une autre ressource sans perte de l'état d'avancement.
- Réallocation sans état – la ressource confie la réalisation d'une tâche à une autre ressource, l'état d'avancement étant perdu.
- Suspension/reprise – la ressource peut arrêter et reprendre l'exécution d'un objet de travail
- Couper (skip) – la ressource marque un objet de travail comme terminé
- Refaire – la ressource recommence un objet de travail déjà réalisé
- Pré fabriquer – une ressource peut prendre de l'avance et réaliser une tâche avant d'être informé du fait qu'elle doive la réaliser.

Patterns d'exécution automatique :
Il s'agit de la situation où l'exécution d'un objet de travail est déclenchée par des évènements spécifiques du cycle de vie de la tâche en cours ou de la définition du processus associé.

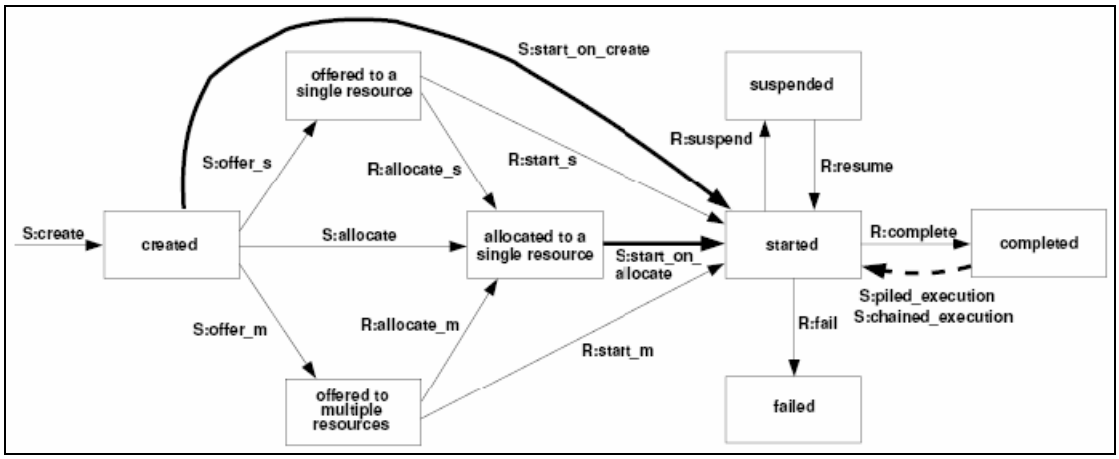


Figure 36. Patterns d'exécution automatique.

- Commencement à la création – la ressource exécute un objet de travail dès qu'il est créé
- Commencement à l'allocation – la ressource exécute l'objet de travail dès qu'il a été alloué
- Exécution empilée – le système de gestion de workflow initie la prochaine instance d'une tâche (même sur un cas différent) dès que la précédente a été complétée.
- Exécution chaînée – le système démarre automatiquement l'exécution d'un objet de travail appartenant à un cas dès que le précédent a été terminé.

Patterns de visibilité :
Leur rôle est de déterminer les objets de travail visibles ou non par une ressource en particulier.

- Visibilité des objets de travail non-alloués configurable.
- Visibilité des objets de travail alloués configurable.

Patterns de multiplicité :
Ces patterns permettent d'exécuter plusieurs objets de travail en même temps ou d'avoir plus d'une ressource en charge d'un objet de travail.

- Exécution simultanée – une ressource exécute plus d'un objet de travail à la fois.
- Ressources additionnelles – plusieurs ressources sont attachées à la réalisation d'un objet de travail.

2.2.3.5 Analyse du workflow

Différentes méthodes d'analyse de workflow existent, permettant de déterminer la correction de sa définition [Vda1]. Plusieurs aspects peuvent être vérifiés de manière informatique, nous allons les aborder en les répartissant parmi plusieurs familles.

Analyse de l'accessibilité

L'accessibilité est le critère déterminant si une place ou transition d'un réseau de Petri pourra recevoir un jeton, partant de l'état de commencement et respectant les règles de routage établies. Lorsque nous avons affaire à un réseau de Petri de petite taille, il est assez facile de vérifier sa correction. Par contre ce test s'avère très utile lorsque nous augmentons la complexité du workflow. L'accessibilité se vérifie via l'utilisation d'algorithmes récursifs. On part de l'état de départ, tous les éléments pointés par une flèche provenant de l'état de départ sont inclus dans les éléments atteignables. Ensuite, on répète l'opération à partir de ces éléments atteints.

Analyse structurelle

L'utilisation des patterns de routage mène à de possibles incohérences. En premier lieu il est important qu'une transition d'un réseau de Petri soit dotée d'un pattern de jointure lorsque plusieurs places d'input lui sont liées. Il en va de même pour les patterns de séparation. Par défaut le pattern de routage liant une unique place d'input ou d'output à une transition est le routage séquentiel.

Six erreurs communes apparaissant dans la définition d'un processus sont recensées dans [Vda1] :

1. Tâches n'étant pas liée à la fois à au moins une place d'input et au moins une place d'output.
2. Tâches mortes, qui ne seront jamais réalisées à cause de la configuration du Petri Net.
3. Deadlock : un cas n'atteignant jamais la place représentant l'état final.
4. Cycle infini, lorsqu'un cas ne peut sortir d'une boucle.
5. Des tâches sont effectuées à nouveau sur un jeton ayant atteint l'état final.
6. Certains jetons restent dans la définition du processus alors que le cas est terminé. En d'autres termes lorsqu'un cas s'est séparé en deux jetons identiques à des places différentes, ces jetons n'ont pas fusionné avant la fin.

La détection de ces erreurs passe par une analyse des éléments constitutifs du schéma du workflow. Par exemple, le fait d'avoir un lien sortant à partir de la place de terminaison sera interprété comme une erreur.

Analyse de la performance

Les aspects quantitatifs de l'utilisation d'un workflow peuvent être mesurés. Il s'agit des données temporelles relatives aux cas, du nombre de cas transitant par le workflow par unité de temps et de l'utilisation des ressources. Cette analyse peut s'appuyer sur des modèles probabilistes Markoviens, ainsi que la théorie des files étudiant les temps d'attente et la simulation. L'implémentation d'une solution logicielle consiste à stocker les données temporelles lors de l'exécution du workflow et à en interpréter les résultats. En dehors d'une situation réelle il faut associer des valeurs probabilistes aux évènements (choix de routage par exemple).

2.2.3.6 Modèle de workflow

Les différents composants abordés à ce stade du niveau workflow sont repris dans le diagramme suivant [Guer1].

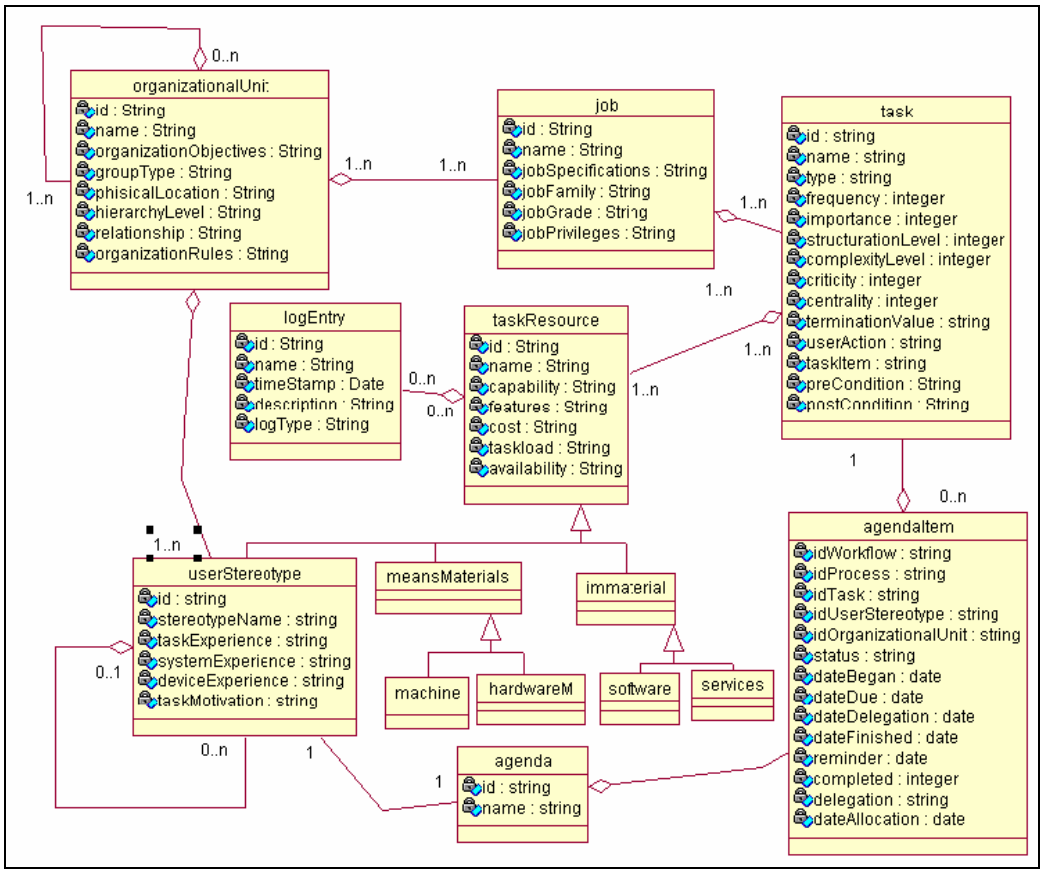


Figure 37. Modèle conceptuel du workflow.

- TaskResource identifie les différentes ressources séparées en trois groupes : userStereotype (travailleur), moyens matériels et immatériels.
- Un travailleur est lié à son agenda reprenant les différents objets de travail qu'il a à accomplir
- Dans une unité organisationnelle se trouvent des travailleurs. Chaque job est effectué dans une ou plusieurs unités organisationnelles. Une tâche peut être réalisée par des ressources capables d'effectuer un des jobs requis.

Chapitre 3 Implémentation logicielle d'un système de gestion de workflow

Ce chapitre se consacre à l'application concrète qui a été développée dans le cadre de ce travail. Il repose sur les fondements théoriques abordés au second chapitre.

Cette application a été divisée en deux entités distinctes, appelées « éditeur de workflow » et « gestionnaire de workflow ». Cette division se base sur les fondements théoriques de l'architecture logicielle établie par la Workflow Management Coalition, qui fait l'objet de la première section.

Dans la seconde section nous rappellerons les attentes formulées lors de l'établissement de l'objectif du mémoire.

La troisième section sera dédiée aux fonctionnalités offertes par le programme. Il s'agira d'expliquer en détail ce que l'utilisateur peut attendre de l'application.

Quant à la quatrième, elle décrira plus avant les packages ainsi que les classes qui constituent ces packages.

Nous terminerons par une estimation de l'effort réalisé dans la mise en œuvre de l'implémentation du logiciel.

3.1 Composantes de l'approche logicielle

La structuration d'un système de gestion de workflow repose en premier lieu sur le constat de la séparation du programme en deux entités temporelles : le temps de design et le temps d'exécution. Les premiers systèmes de gestion de workflow n'étaient pas décomposés en modules. Nous verrons pourquoi c'est intéressant de le faire et ensuite le modèle architectural proposé par la Workflow Management Coalition [Wfmc1].

3.1.1 Relation au temps

Deux moments principaux sont à distinguer dans le cadre de l'utilisation d'un système de gestion de workflow. Le temps de construction (build time, aussi appelé design time) et le temps d'exécution (run time).

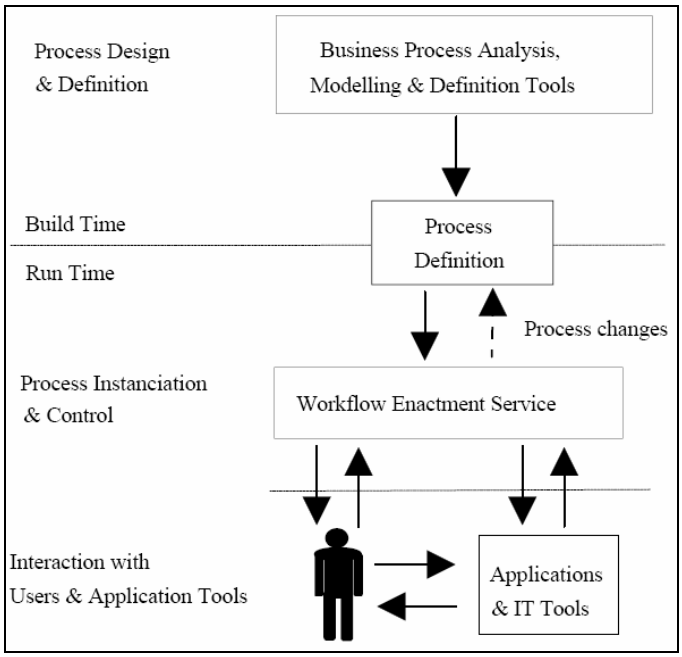


Figure 38. Séparation temporelle de l'architecture.

Le temps de construction est utilisé pour la définition et la modélisation des processus business intervenant dans le workflow. A cette étape un processus est traduit depuis le monde réel vers sa version informatisée, la définition du processus. Cela peut se faire sur base d'un éditeur graphique. La standardisation est importante pour l'output de cette activité, chaque programme de définition étant libre des moyens mis en œuvre (interfaces graphiques notamment) pour parvenir à cette spécification.

La seconde étape dans le cycle de vie d'un workflow est le temps d'exécution. A ce stade la définition des processus est utilisée par le système de gestion de workflow (workflow enactment service) afin de gérer dynamiquement le workflow. Il s'agit de la gestion des cas, des tâches qui seront exécutées sur ce cas et de l'attribution des ressources à ce travail. Nous en détaillerons le fonctionnement dans la section dédiée au modèle de référence les différentes composantes d'un système de gestion de workflow et la communication établie entre eux.

3.1.2 Séparation de la gestion et de l'exécution

Historiquement, le système d'information formait une entité non-décomposée en modules. La première étape menant à la structuration actuelle d'un système de gestion de workflow a été d'en extraire la partie applicative. Cette dernière consiste en l'ensemble des programmes permettant la réalisation du travail, par opposition aux aspects organisationnels. Nous obtenons alors la représentation de la figure suivante.

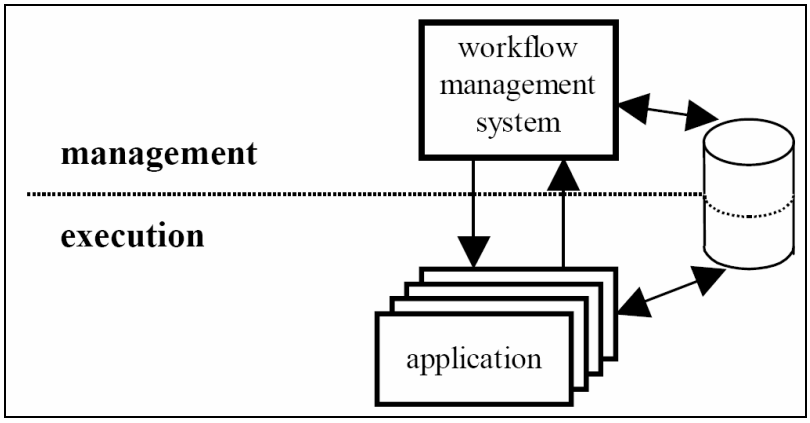


Figure 39. Séparation de la gestion et de l'exécution.

Cette séparation permet de faire évoluer les deux parties indépendamment l'une de l'autre. Il est donc possible de changer les processus business sans avoir à modifier les applications permettant la réalisation effective du travail. L'inverse est vrai également, ce qui apporte des avantages tels que la possibilité de changer les ordinateurs, systèmes d'exploitation et programmes utilisés par les travailleurs indépendamment.

3.1.3 Modèle de référence

La construction d'un système de workflow fait l'objet d'un modèle de référence décrit par la Workflow Management Coalition [Wfmc1]. Ce modèle décrit l'architecture d'un tel système, en se basant sur des composants et leurs interfaces. Il s'agit de la figure40.

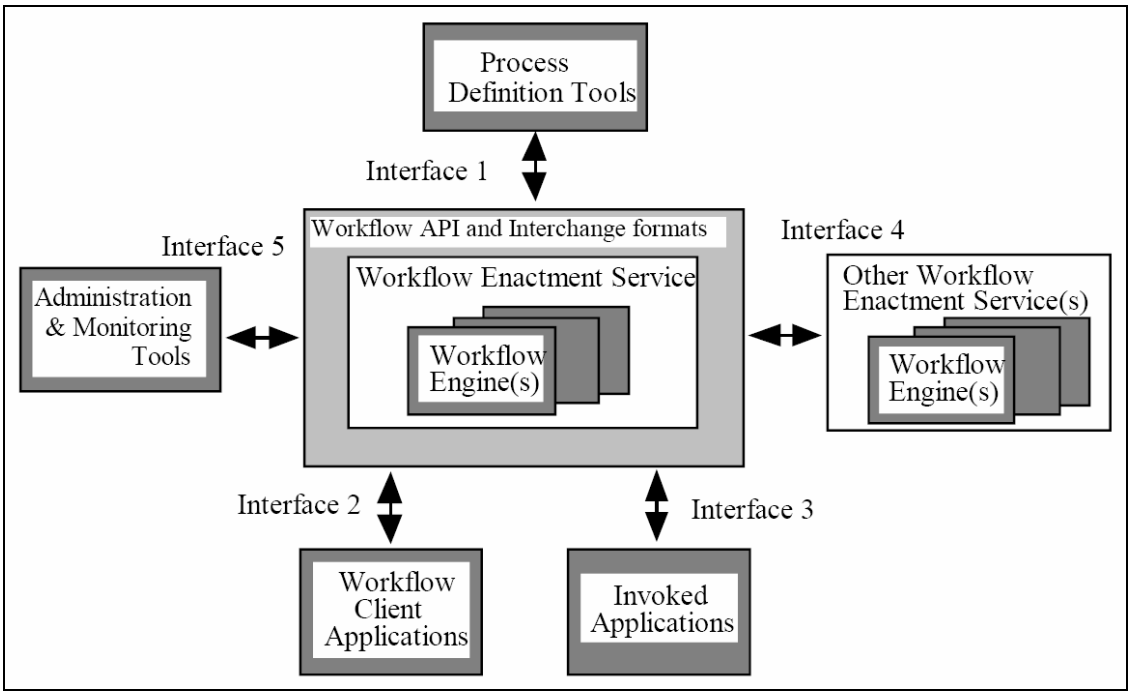


Figure 40. Modèle architectural de référence du WFMC.

Nous allons maintenant définir un à un les composants de ce schéma.

3.1.3.1 Service de gestion de workflow

Le centre de cette architecture est le service de gestion de workflow (workflow enactment service). Il possède un ou plusieurs moteurs de workflow (workflow engines), dont le rôle est de traiter la gestion des cas. Il s'agit des opérations suivantes :

- création de nouveaux cas
- routage des ces cas selon la définition du processus correspondant
- gestion des informations des cas (attributs et historique)
- suivi de la consistance du workflow
- gestion des utilisateurs
- interaction avec les différents composants

3.1.3.2 L'outil de définition de processus.

Cet outil a pour output la génération d'une spécification de processus qui sera interprétée par le service de gestion de workflow (un de ses moteurs en réalité). La séparation de ces deux entités permet en premier lieu de créer une distinction entre le temps de design et le temps de gestion/exécution. Cela s'inscrit également dans la logique de standardisation des modèles puisqu'il sera possible de remplacer les outils de définition de processus pour autant que leur output réponde aux mêmes normes que l'ancien.

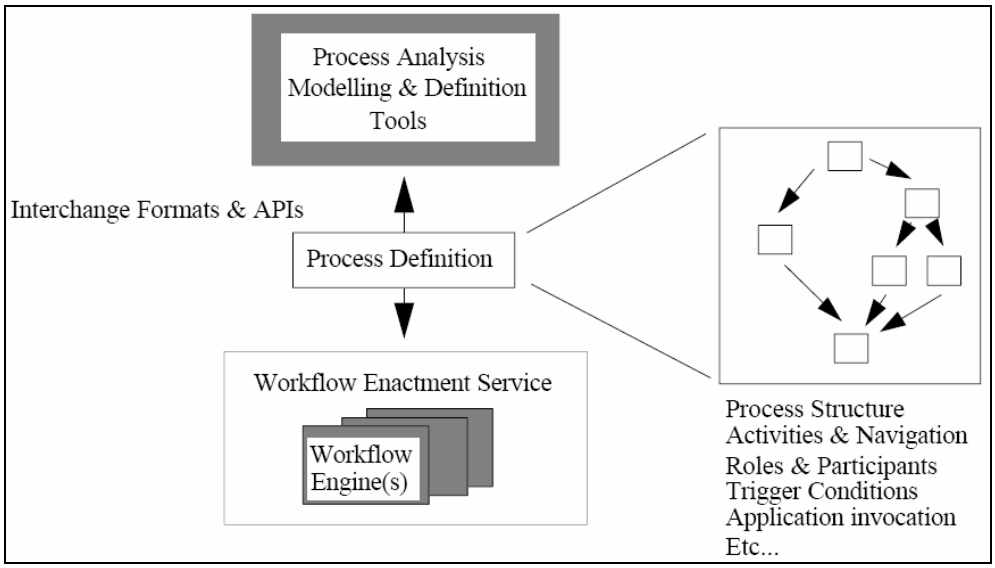


Figure 41. Outil de définition de processus.

L'utilisation de l'outil de définition de processus consiste à spécifier les tâches et leur ordre, vérifier la validité du schéma réalisé ainsi que recenser les ressources.

Des méthodes d'analyse sont disponibles afin de vérifier la validité du schéma. Cet aspect est important car l'utilisation d'un workflow mal spécifié mènerait forcément à des incohérences et à de nombreux problèmes.

Le dernier aspect, consistant en la définition des ressources disponibles, permettra de déterminer par les moyens qui prendront en charge la réalisation du travail. Il s'agit en premier lieu de ressources humaines, possédant des qualifications et capables d'accomplir un ou plusieurs jobs. Mais également de ressources matérielles et immatérielles telles que les équipements informatiques.

3.1.3.3 Les applications client du workflow

Le moyen de communication du travailleur avec un moteur de workflow est l'application cliente. C'est la seule chose que ce travailleur percevra du système de workflow. Cette application est dotée d'une interface, ce qui nous mène à la représentation du schéma (figure 42).

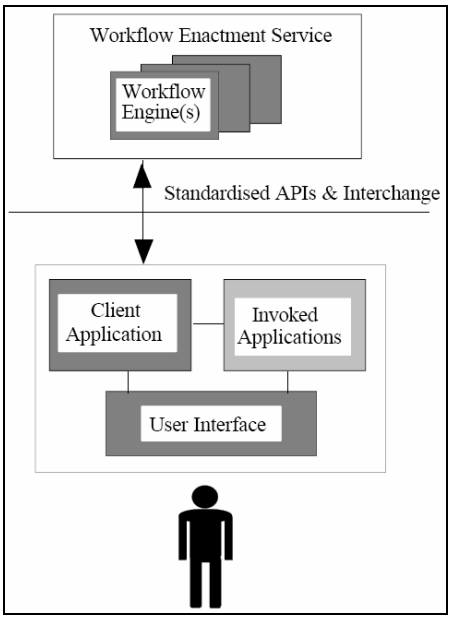


Figure 42. Application client.

Cette application consiste principalement en une liste de travail (worklist). L'utilisateur recevra donc des offres de travail qu'il pourra accepter ou refuser. Une fois que le travail lui est alloué, il reste à réaliser son exécution effective. Le suivi de ces opérations est réalisé par l'application.

3.1.3.4 Applications invoquées

Il s'agit des applications utilisées dans le cadre de la réalisation des tâches. Elles n'appartiennent pas au service de gestion de workflow car il ne s'agit pas de définir comment le travail est réalisé mais bien de l'exécuter.

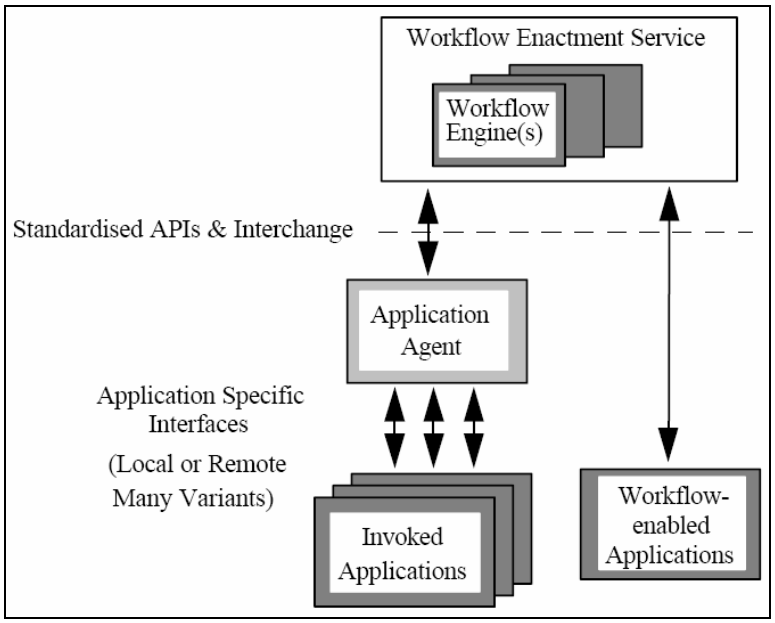


Figure 43. Applications invoquées.

Les applications invoquées peuvent communiquer avec le système via un agent ou être simplement démarrées par le service de gestion de workflow. Certaines applications sont interactives, et donc dirigées par l'utilisateur. Elles correspondent alors à un objet de travail

attribué au travailleur qui utilise l'application. D'autres sont automatisées et ne requièrent pas d'interaction avec un utilisateur.

3.1.3.5 Outils d'administration et de suivi

La supervision et la gestion opérationnelle du workflow sont réalisées grâce à ces outils. La gestion opérationnelle consiste à configurer et gérer l'exécution des processus pour les cas qui y cheminent. C'est également par le biais de ces programmes que se fait la gestion des ressources (ajouts, suppressions et modifications). Enfin, il est possible de charger une définition de processus dans un moteur de workflow.

Le suivi du workflow se fait via l'enregistrement continu de données concernant le workflow. Il est alors possible de connaître le temps moyen passé dans le workflow pour un cas déterminé, le temps d'attente moyen avant de réaliser un objet de travail, le taux d'utilisation des ressources ou toute autre information de ce type. Ces informations permettront aux gestionnaires de prendre les décisions qui s'imposent afin de rectifier des situations problématiques telles que les goulots d'étranglements constatés à certaines tâches par exemple.

3.2 Spécification du logiciel réalisé

Nous allons maintenant aborder les attentes formulées pour le programme réalisé dans le cadre de ce travail. La réalisation de l'objectif principal nous mène à rappeler les points suivants.

La première partie du programme est un éditeur graphique de workflow. Cet éditeur permet de manipuler les concepts de tâches, processus et workflow. L'utilisateur peut alors lier ces concepts entre eux et gérer leurs attributs.

Nous avons vu au chapitre 2 les déterminants de la modélisations d'un workflow. Les tâches sont créées et décomposées grâce à l'arbre des tâches concurrent. Les processus sont définis grâce à la modélisation des réseaux de Petri. Les patterns de routage sont alors utilisés pour déterminer la manière dont un cas peut évoluer au sein du processus. Les unités organisationnelles sont définies et liées aux processus. Les patterns de ressource sont utilisés pour définir le cadre de l'attribution du travail aux ressources. Enfin, les travailleurs sont créés ainsi que leurs jobs.

Tous les éléments peuvent être inclus au schéma, déplacés, redimensionnés et supprimés. La liste de ces éléments est la suivante.

- Place
- Transitions
- Boîte à ressource (JobBox).
- Groupe de boîtes à ressource (JobGroup).
- Unité organisationnelle.

Des liens existent entre places transitions, comme définis dans la section dédiée à l'utilisation des réseaux de Petri. Les jobs et ressources sont spécifiés.

Selon [Vda1], les fonctionnalités dont un outil de définition de processus doit être doté sont les suivantes :

- Capacité à établir la définition des processus (nom, description, …)
- Capacité à modéliser le routage par des moyens graphiques (And split, and join, …)

- Support du versionnage (différentes versions d'un même processus)
- Définition des attributs des cas d'un processus
- Spécification des tâches
- Vérification syntaxique de la validité de la définition d'un processus

La définition du workflow peut être sauvée, sous deux formats différents. En premier lieu un format interne au programme, en second lieu sous la forme XML. Il est aussi possible de réaliser des captures d'écran.

Ensuite il est possible de créer des cas et de les faire évoluer au sein du workflow défini par l'éditeur. Pour cela la gestion des ressources est prise en compte via l'attribution du travail aux ressources.

3.3 Fonctionnalités implémentées

Nous allons aborder ici les opérations permises par le programme. Ces opérations sont scindées en deux groupes : en premier lieu celles auquel l'utilisateur aura accès au temps de design, via « l'éditeur de workflow » (4.1.1). Le fonctionnement de cet éditeur est statique, il n'inclut pas d'opérations relatives au temps. Son utilisation permet de spécifier le workflow en le dessinant et en effectuant le paramétrage des différentes entités utilisées. Il s'agit notamment de la gestion des attributs de ces éléments, de la définition des ressources et de la décomposition des tâches.

Nous verrons ensuite la partie dynamique du programme, que nous appellerons le « Gestionnaire de workflow » (4.1.2). C'est là que nous pourrons créer des cas et voir leur cheminement au sein du workflow.

3.3.1 Editeur de workflow

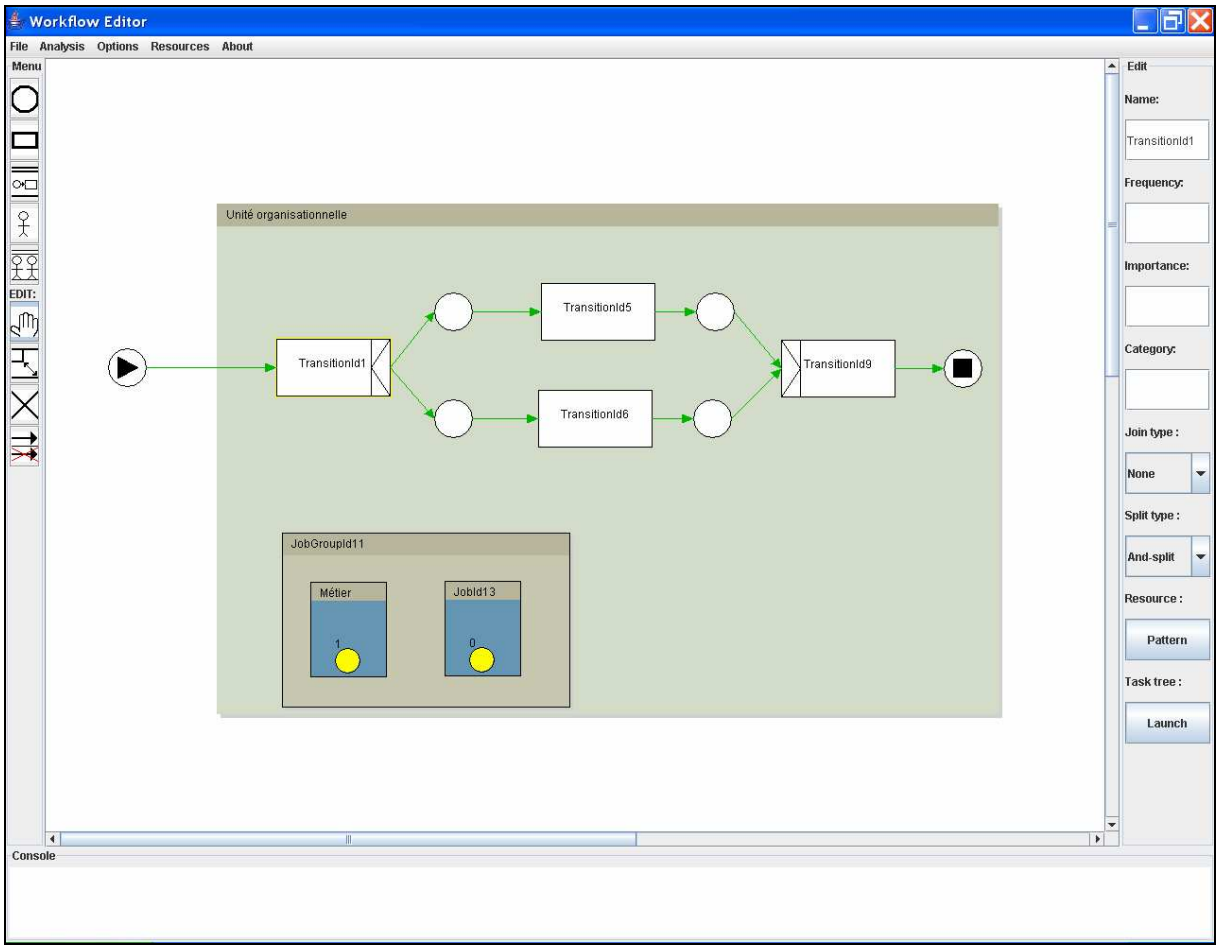


Figure 44. Editeur de workflow.

L'éditeur permet de spécifier un workflow. Il s'agit de l'outil de définition de processus du modèle architectural WFMC. La représentation est basée sur celle des Petri Nets évoqués plus haut, et a été enrichie avec d'autres éléments. Le programme a une disposition des éléments proche d'un éditeur graphique tel que Microsoft Paint.

- Sur la gauche une barre d'outils permettant d'insérer les éléments graphiques, de les sélectionner, déplacer, redimensionner, supprimer ainsi que les relier par une flèche.
- En haut se trouve le menu permettant d'accéder aux options.
- A droite le menu d'édition de l'élément sélectionné.
- Au centre la représentation graphique du workflow, dont les éléments sont accessibles selon les options déterminées par la barre d'outils.
- Enfin, une console permettant de communiquer des messages à l'utilisateur se trouve en bas.

3.3.1.1 Représentation du workflow

Le dessin du workflow repose sur l'utilisation d'éléments que nous allons maintenant décrire. Afin de pouvoir lier ces définitions à leur visualisation veuillez vous référer à la figure suivante.

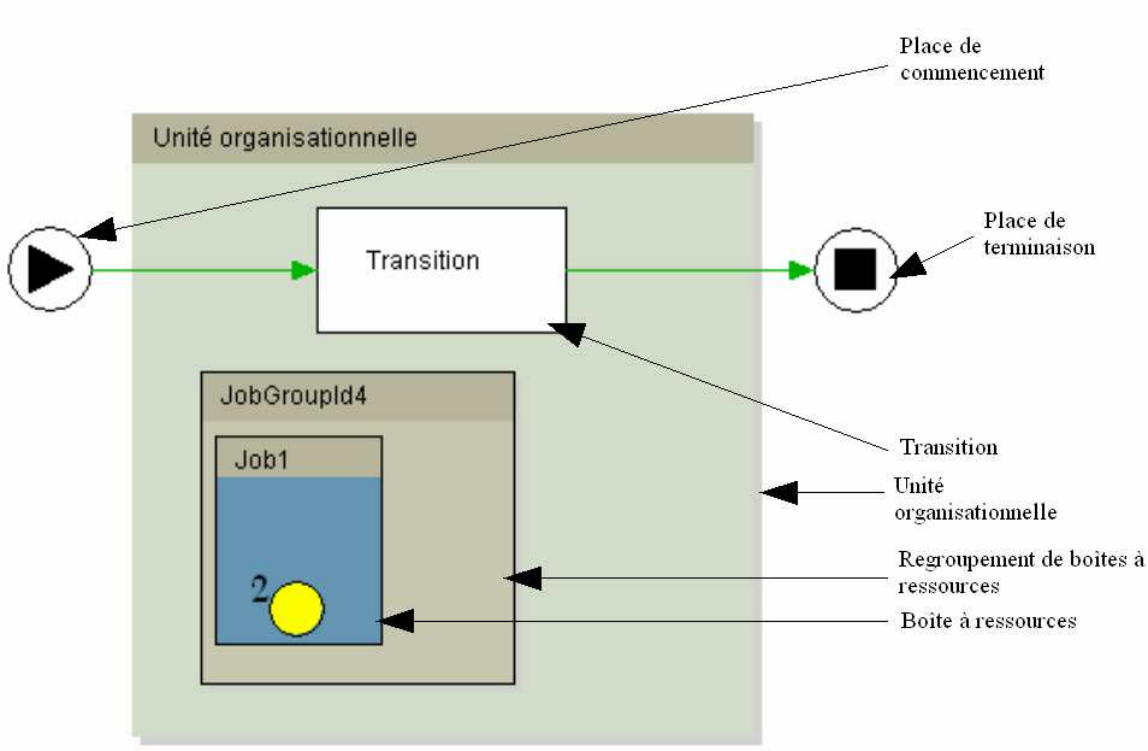


Figure 45. Eléments constitutifs d'un workflow.

1. Les places

Il existe trois types de places : commencement (start), terminaison (end) et normale. La place de commencement permet de déterminer le point d'entrée du workflow. C'est là que seront placés les cases qui viennent d'être créés. Quand à la place de terminaison elle permet d'identifier la fin du processus.

2. Les transitions

Ces processus atomiques, ne contenant qu'une tâche, sont dotés d'un pattern de routage en entrée et d'un autre en sortie. C'est par le biais de leur édition que l'on accède à la fois à l'arbre de décomposition de la tâche et que l'on peut assigner les patterns de ressources associés à la tâche.

3. Les arcs

C'est par l'utilisation des flèches que s'effectue la liaison entre places et transitions.

4. La boîte à ressources (job box)

Cette boîte contient une série de travailleurs exerçant un même métier (job). Cela permet de localiser des ressources. Pour cela nous considérons que la portée d'un travailleur est l'unité organisationnelle à laquelle appartient sa boîte à ressource. Le choix du job, l'ajout et le retrait de travailleurs se font via le menu d'édition.

5. Le regroupement de boîtes à ressources (job group)

Il s'agit d'un container permettant d'accueillir plusieurs boîtes à ressources afin de clarifier la représentation.

6. L'unité organisationnelle

Ces unités permettent de contenir une partie du workflow afin de déterminer concrètement dans quel lieu elle prendra place. Une unité organisationnelle dispose de ressources,

indiquées par les boîtes à ressources. Concrètement, pour qu'un travailleur puisse être attribuable à une tâche il faut qu'il se trouve soit dans la même unité organisationnelle, soit dans une unité parente.

3.3.1.2 Gestion des fichiers

Le menu « file » permet de réaliser les opérations suivantes :

- Créer un nouveau workflow
- Sauver/charger une définition de workflow existante
- Enregistrer le dessin du workflow au format « .jpg »
- Exporter la spécification de ce workflow au format Usi-XML
- Quitter l'application

3.3.1.3 Analyse du workflow

Plusieurs tests permettent au concepteur du workflow de vérifier la validité de celui-ci. Il est en effet assez classique que l'utilisateur ait introduit des erreurs dans le schéma du workflow. Une vérification automatisée devient alors une aide importante dans le cadre d'un programme ayant pour but non seulement d'aider l'utilisateur par une interface qui lui soit adaptée mais également de prodiguer certaines corrections.

- Test start/end

Il est ici question d'inspecter le workflow afin de déterminer si les états de commencement et de sortie sont présents. L'état "start" est celui d'où partiront les jetons (tokens), son absence aurait pour effet de ne pas pouvoir tracer les exécutions des cases dans le schéma et donc l'utilisation dynamique du programme. Quand à l'état "end" il permet de déterminer qu'un case particulier a atteint l'état final et que la procédure de travail est donc terminée.

- Test d'accessibilité (reachability)

Par ce test le programme permet de savoir si tout élément du schéma, place ou transition, est atteignable. La vérification consiste à partir de l'état "start" et de suivre les liens du graphique. Tout élément atteint sera marqué comme "atteignable" et une méthode récursive est appliquée afin de voir tous les éléments effectivement atteignables. Ajoutons qu'un simple test consistant en la vérification d'un lien entrant pour tout élément n'aurait pas suffi. En effet, un groupe d'éléments connectés entre eux mais non reliés au reste du graphe vérifierait cette condition alors que les éléments ne sont pas atteignables depuis l'état "start".

- Test des patterns de routage (join/split)

Le choix a été posé d'utiliser des patterns de flux explicites. Qu'il s'agisse de la jointure (join) ou de la division (split), les patterns de routage que sont and, xor, et and/or sont utilisés pour modéliser les différents chemins à suivre par les tokens. Certaines combinaisons de choix de l'utilisateur pourraient être irréalistes, par exemple le fait de choisir un pattern de join alors que la transition n'a qu'un seul lien entrant. Le test répond à ce type de problème en corrigeant l'erreur si la solution est triviale (dans le cas précédemment évoqué, enlever le pattern de jointure). Pour les cas où un plusieurs choix sont possibles, tels que le manque d'un pattern de division, l'utilisateur sera averti de la nécessité d'inclure le pattern.

3.3.1.4 Gestionnaire des métiers (job handler)

C'est par le biais de ce gestionnaire que les métiers sont définis. L'utilisateur entre le nom, les spécifications, la famille, le grade et les privilèges définissant le job. L'accès à ce gestionnaire se fait via le menu situé au nord de la fenêtre de l'éditeur, dans la partie « ressources ».

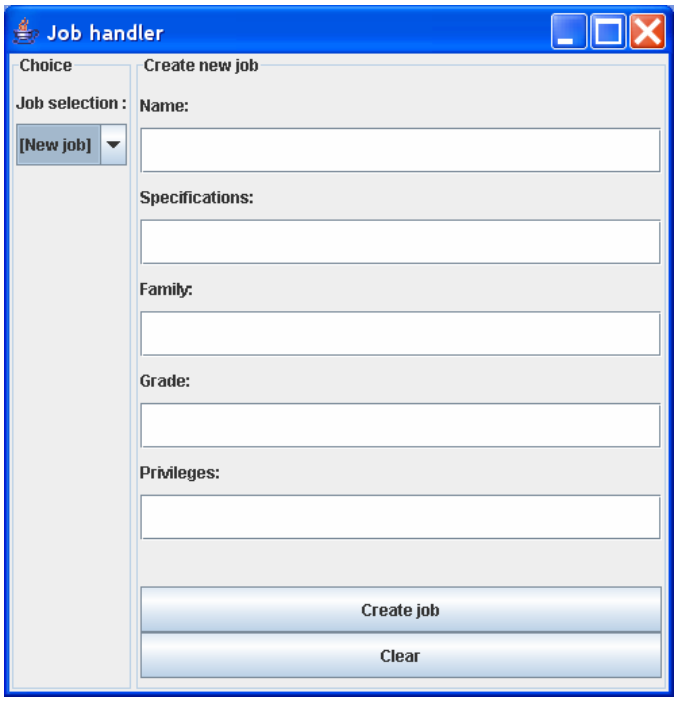


Figure 46. Gestionnaire des métiers.

3.3.1.5 Editeur de travailleurs

La fenêtre de cet éditeur s'active dans la partie « ressources » du menu. Il permet de créer un nouveau travailleur, et de déterminer les métiers que ce travailleur est capable d'accomplir. Ces jobs se trouvent dans le champs « job list » et sont éditables par les opérations add job et remove job. Enfin, deux valeurs (arbitrairement définies entre un et cinq) permettent de déterminer le niveau d'expérience et de hiérarchie du travailleur. Elles seront utilisées dans le cadre de l'utilisation des patterns de ressources permettant de savoir si un travailleur convient pour réaliser une tâche.

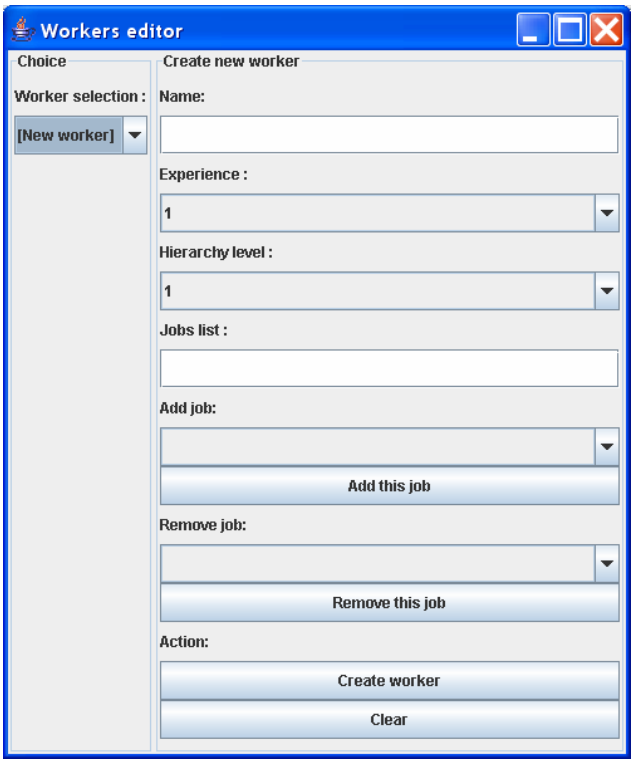


Figure 47. Editeur de travailleurs.

3.3.1.6 Console

La console est le moyen textuel d'informer l'utilisateur pour le programme. Lorsqu'un nouvel événement se produit, la console sera vidée de son contenu et les informations successives seront écrites ligne par ligne afin d'obtenir la meilleure visibilité possible. La zone de texte est scrollable (défilement du texte exercé par l'utilisateur) afin de pouvoir stocker une quantité d'information aussi importante que nécessaire.

Ci-dessous un exemple d'output de la console suite à un test sur les patterns de routage effectué par l'utilisateur.

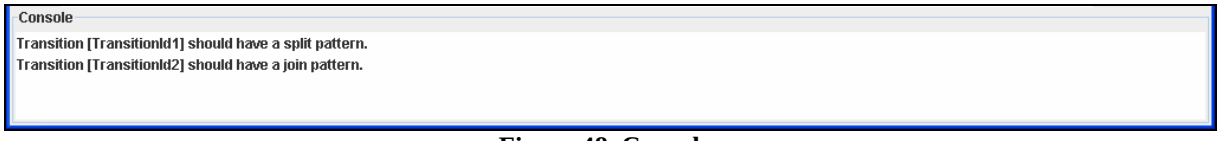


Figure 48. Console.

3.3.1.7 Gestionnaire des ressources (resource patterns handler)

A ce stade nous définissons l'étendue des ressources allouables à une tâche, la manière dont la ou les ressources seront prévenues de l'offre ou allocation de travail ainsi que le moment auquel ces ressources seront prévenues. Il s'agit donc de l'utilisation des patterns de ressource, en se limitant bien sûr à ceux disponibles au temps de design.

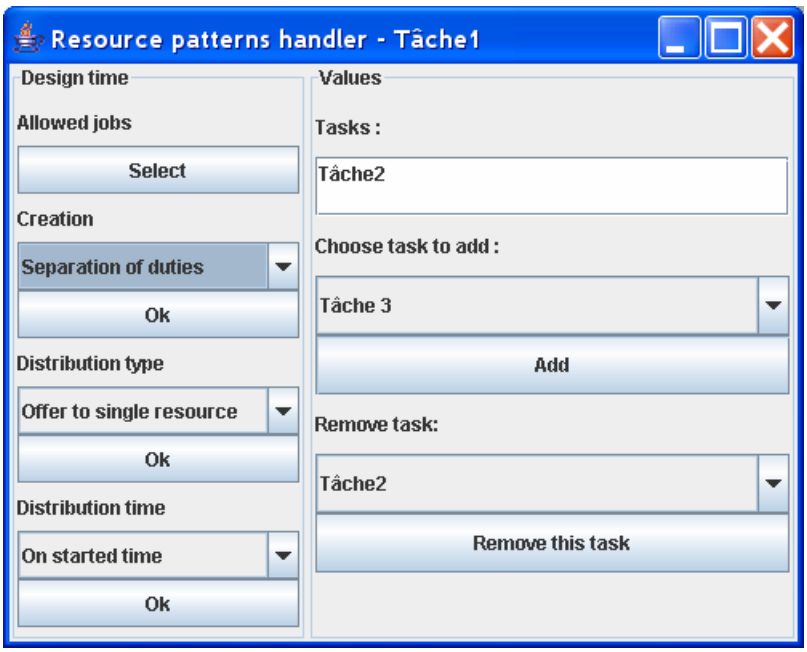


Figure 49. Gestionnaire des ressources.

C'est grâce au gestionnaire des ressources que sont définis les métiers qui permettent d'effectuer la tâche. Il s'agit de réaliser une liste de ces jobs.

Les dix patterns de création ont été implémentés. Leur rôle est de déterminer quels seront les travailleurs aptes à effectuer le travail. Il s'agit d'allocations/offres de type

1. Direct : allocation au temps de design.
2. Différé : offre/allocation au temps de gestion.
3. Basé sur l'autorisation : le travailleur doit avoir une autorisation avant de pouvoir effectuer la tâche.
4. Séparation des devoirs : deux tâches pour un même case ne peuvent se voir attribuer le même travailleur.
5. Gestion de cas : donner toutes les tâches effectuées sur un même cas au même travailleur.
6. Retenir les familiers : utiliser si possible un travailleur ayant déjà travaillé sur le cas.
7. Basé sur la capacité : un niveau minimum d'expérience est requis.
8. Basé sur l'historique : utiliser une ressource ayant déjà réalisé la tâche auparavant.
9. Basé sur le niveau hiérarchique.

L'utilisateur a la possibilité de déterminer si le travail sera alloué ou offert, le travailleur devant donner son accord dans ce dernier cas. De plus, une offre peut être envoyé à plusieurs ressources à la fois. Nous avons donc affaire à trois possibilités : offre à une ressource unique, offre à un ensemble de ressources, allocation à une ressource unique.

Enfin, on détermine le moment auquel l'utilisateur sera informé qu'il faut exécuter la tâche, grâce aux trois patterns dévolus à ce rôle. Cette information pourra arriver avant la création de l'activité (càd lorsque le jeton atteint une tâche), à sa création ou plus tard.

La figure 49 montre un exemple où le pattern de création est la séparation des devoirs (separation of duties). Etant donné que nous éditons la tâche numéro 1 (task1), l'utilisateur a spécifié via le volet à droite qu'il souhaite que la ressource ayant effectué la tâche 2 ne soit pas autorisée à réaliser la performance de la tâche courante. Le pattern de distribution est «offre à ressources multiples », ce qui signifie que le travail pourra être offert à une ou plusieurs ressources. Enfin, le temps de distribution spécifié est le moment de commencement de la performance de la tâche.

3.3.1.8 Editeur d'arbre de tâche (Task Tree Editor)

Accessible via le menu d'édition d'une transition, l'éditeur Ideal-XML permet de réaliser l'arbre de décomposition de la tâche. Son auteur, Francisco Montero, a accepté l'intégration de son programme dans ce travail. La représentation utilisée est la modélisation par arbre de tâche concurrent (Concurrent Task Tree - CTT). La tâche principale est décomposée en sous-tâches elles-mêmes décomposables. Les tâches possédant chacune un type parmi les suivants: abstrait, interactif, application, utilisateur, groupe. De plus les opérateurs LOTOS permettent de définir les règles temporelles liant les tâches deux à deux.

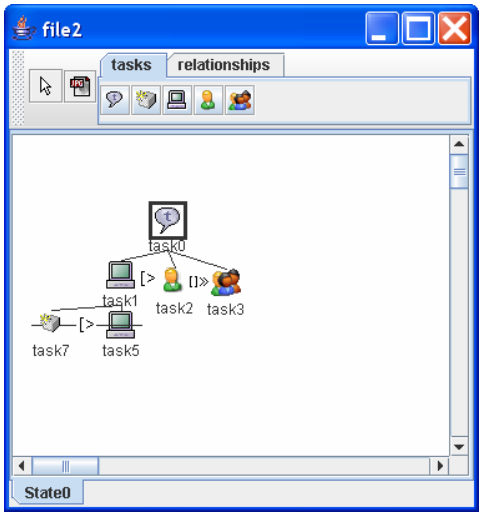


Figure 50. Editeur d'arbre de tâche.

Ideal-XML (Interface Development Environment for AppLications specified in usiXML) est un outil permettant une approche du design d'interface utilisateur basée sur la tâche. Il permet de réaliser facilement des prototypes à partir de la spécification réalisée à l'aide d'un outil graphique. Une fois que les attributs de la tâche sont spécifiés, ainsi la manière dont l'utilisateur va la remplir (saisie d'un champ textuel par exemple), une seconde fenêtre permet d'obtenir automatiquement une interface utilisateur. Cette interface pourra être revue et corrigée afin d'y apporter des améliorations éventuelles.

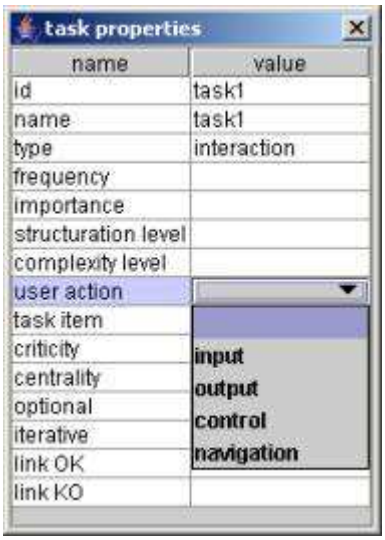


Figure 51. Propriétés de la tâche.

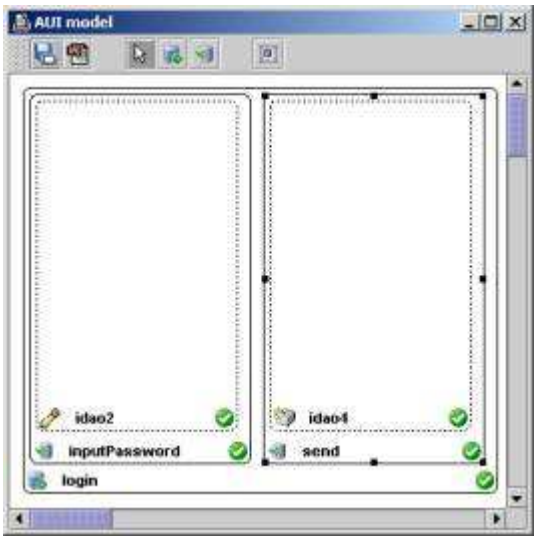


Figure 52. Interface utilisateur abstraite.

3.3.1.9 Sauvegarde au format Usi-XML

L'utilisation d'un système de gestion de workflow repose sur des interactions avec des agents humains. Pour cela nous utilisons des interfaces utilisateur, qui sont donc le moyen de communication entre le système et cet utilisateur.

Il est possible de dériver des interfaces utilisateur à partir d'informations codées dans le langage XML (eXtensible Mark-up Language). Il s'agit d'un langage reposant sur l'utilisation de balises, au même titre que le HTML que l'on retrouve dans la programmation des sites internet. Une balise est un composant du type suivant :

```
<type attribut1= « contenu2 » attribut2= « contenu2 » …>
```

La structuration des balises forme un arbre, dans lequel sont insérés d'autres balises ou un contenu spécifique. Pour insérer des nœuds fils, on a recours à une balise ouverte, suivie par le contenu et enfin une balise fermée. Ce qui nous mène à la représentation suivante :

```
<Balise>
     <fils>
     Contenu
</Balise>
```

Le langage XML offre donc une structuration générique, permettant de stocker tout type de données. Il est simple d'accès et est conforme à la norme ISO 8879. En l'occurrence nous l'utiliserons dans le cadre du design d'interfaces utilisateurs. Pour cela nous aurons recours à un langage de description d'interface utilisateur (user interface description langage - UIDL) reposant sur la norme XML [Stav].

Notre choix s'est porté sur le langage UsiXML (User Interface eXtensible Markup Language). Le but de ce langage est d'exprimer des interfaces utilisateur construites avec plusieurs modalités d'interaction, et qui soient indépendantes de ces modalités [Usi]. Sa conformité à XML permettant de « réaliser des échanges flexibles d'information et une communication puissante entre modèles et outils utilisés dans l'ingénierie des interfaces utilisateur. » UsiXML est défini comme étant un ensemble de schémas XML, chacun de ces schémas correspondant à un modèle appartenant au langage. Il est indépendant de toute technologie, modalité d'interface (textuelle, vocale, …) ou logiciel.

D'autres langages de description d'interface utilisateur auraient pu être utilisés, tels XIML (eXtensible Interface Mark-up Language), UIML (User Interface Mark-up Language), XUL (XML-based User Interface Language), ou encore AUIML (Abstract User Interface Mark-up Language).

Les composants d'UsiXML sont les modèles suivants [Usi] :

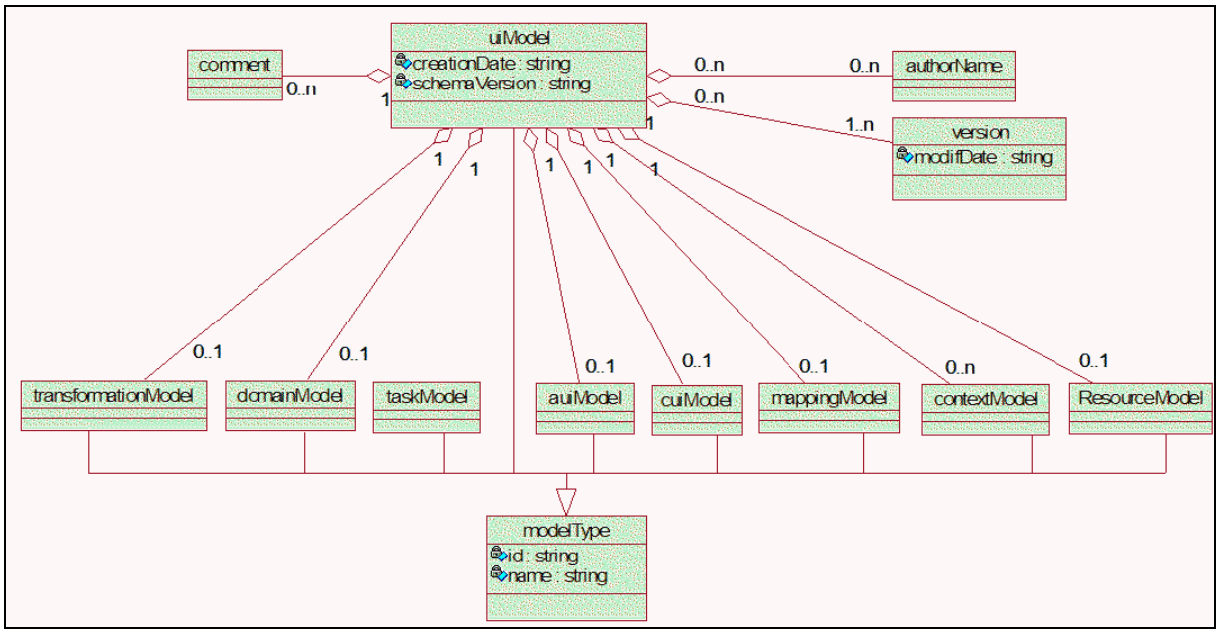


Figure 53. Composants UsiXML.

1. Modèle UI – modèle principal, incluant toutes les modélisations suivantes.
2. Modèle de mappage – servant à réaliser des mappages inter-domaines
3. Modèle du domaine – description des classes et objets manipulés par l'utilisateur lorsqu'il interagit avec le système.

4. Modèle AUI (Abstract User Interface) – modèle représentant les rendus et manipulations des domaines conceptuels
5. Modèle CUI (Concrete User Inteface) – représente la concrétisation du modèle AUI.
6. Modèle des tâches – représentant la décomposition CTT des tâches.
7. Modèle contextuel – décrit les aspects du contexte d'utilisation dans lequel un utilisateur final réalise une tâche interactive avec une plateforme informatique.
8. Modèle de transformation – contient les règles permettant la transformation d'une spécification vers une autre.
9. Modèle des ressources – modélise la définition des ressources (tout contenu pouvant être attaché à un objet interactif).

L'utilisation de la fonction d'exportation au format XML du logiciel implémenté pour ce travail ne réalise pas l'ensemble de ces modèles. Il prend en charge les modèles suivants :

- Modèle des tâches. Défini grâce à l'éditeur CTT inclus au programme, il permet de définir l'ensemble des tâches et leurs attributs respectifs.
- Modèle des processus (définition des processus). Dans ce modèle nous retrouvons le détail de l'ensemble des processus.
- Modèle des ressources : il s'agit des unités organisationnelles et jobs y qui sont réalisés.
- Un modèle de mappage au niveau processus, permettant d'enregistrer les patterns de routage utilisés (jointures et divisions).
- Un modèle de mappage au niveau tâche : il définit les relations inter-tâches du niveau tâches (relations temporelles LOTOS et relations de décomposition).

Il est possible de dériver automatiquement les modèles concernant l'interface utilisateur à partir de ces modèles. Le fonctionnement de ce processus est le suivant.

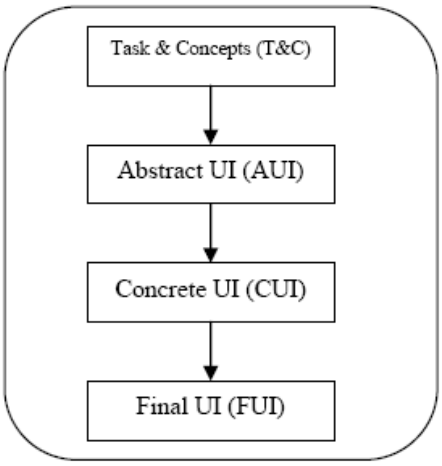


Figure 54. Dérivation d'interface utilisateur.

La première étape est la définition des tâches et concepts, à partir de là est dérivée une interface utilisateur abstraite puis concrète et enfin l'interface utilisateur finale.

3.3.2 Gestionnaire de workflow (case manager)

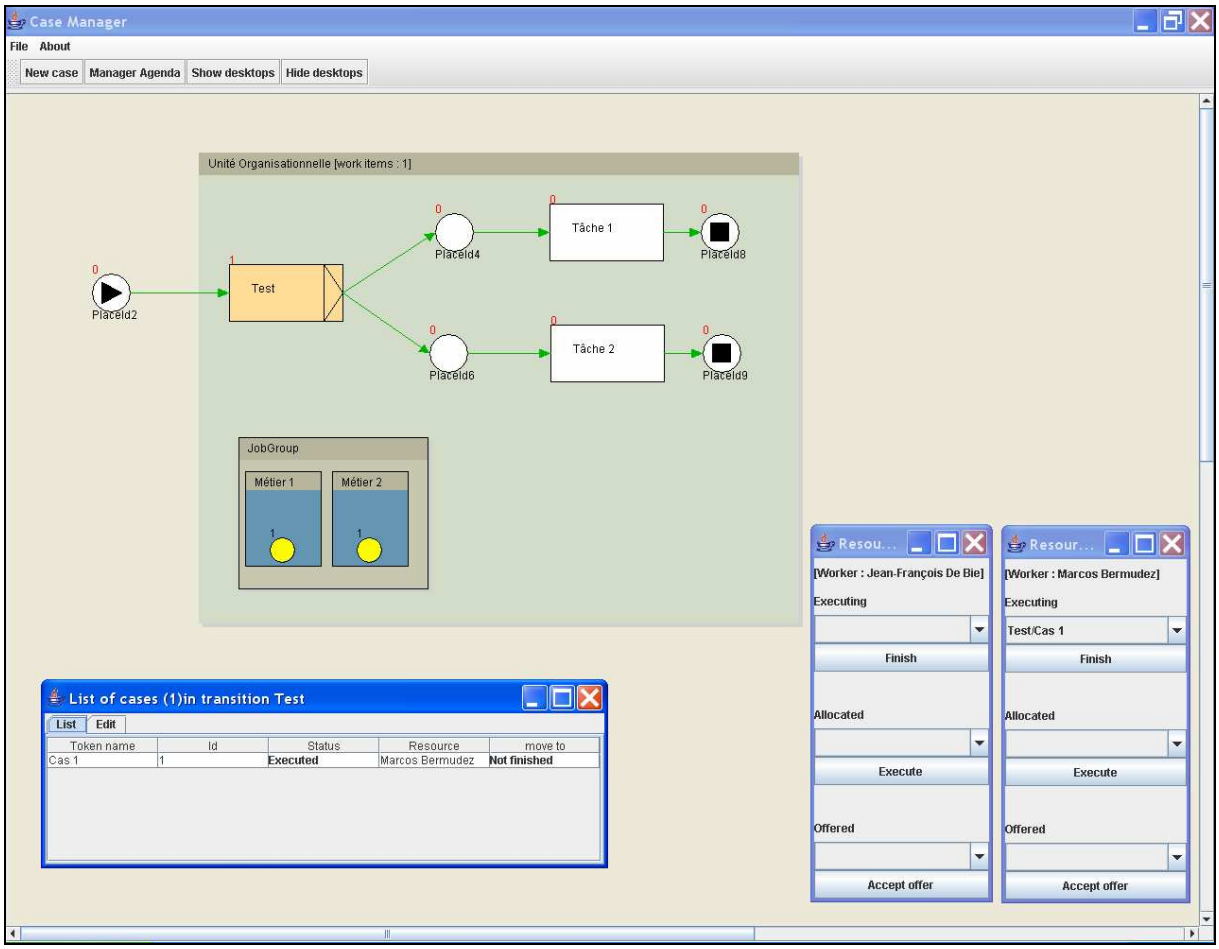


Figure 55. Gestionnaire de workflow.

Le gestionnaire de workflow permet la simulation du fonctionnement d'un service de gestion de workflow tel que celui évoqué dans le modèle architectural WFMC.

Cette seconde partie du programme est la partie dynamique : elle permet de tracer les cas de travail tout au long de leur cheminement dans le workflow. C'est pour cette raison qu'un nombre est associé à toute place et transition : il s'agit des jetons qui se trouvent dans cet élément à un moment donné.

Nous aborderons la manière dont le routage des cas a lieu et les changements graphiques selon le nombre de jetons se trouvant dans une transition. Ensuite nous verrons les différentes interfaces permettant de déterminer la ressource en charge d'un travail. Puis d'effectuer une simulation de la traçabilité de l'état d'avancement via des passages d'état commandés par le travailleur. Enfin, nous parlerons de l'agenda du manager, résumant la situation des jetons dans un tableau.

3.3.2.1 Routage des cas

L'utilisateur commence par charger la spécification d'un workflow réalisée précédemment grâce à l'éditeur. A ce stade on crée un ou plusieurs nouveaux cas via le menu (new case).

Ces cas se retrouvent au départ dans la place d'entrée. Leur cheminement dépendra de la cartographie établie avec l'éditeur.

Lorsqu'un cas est dans une place, cela signifie que la prochaine tâche va pouvoir être associée à une ressource. Pour cela nous cliquons sur la place, afin d'obtenir la fenêtre suivante.

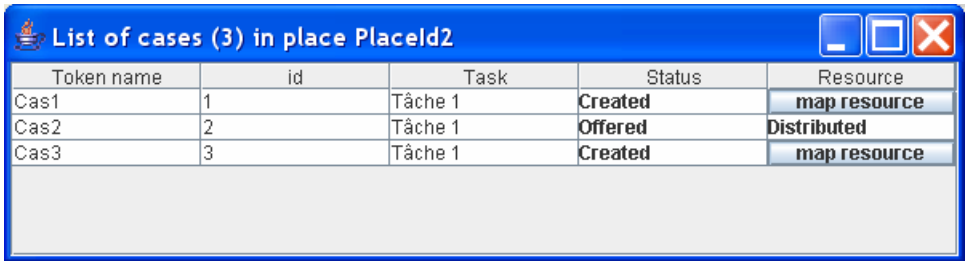


Figure 56. Liste des cas appartenant à une transition.

Outre le nom et l'id du cas (token) se trouvant dans la place, il y a le nom de la prochaine tâche et l'accès à la fenêtre d'assignation de ressources grâce au bouton « map resources » (veuillez vous référer à la section s'y rapportant pour plus de détails). De plus, le statut est l'information permettant de déterminer si l'assignation des ressources doit encore être réalisée ou si par contre le travail a été offert ou alloué.

Une fois qu'une ressource a été chargée de la réalisation de la tâche et que l'exécution de cette dernière a eu lieu, le jeton est déplacé dans la transition correspondante. En cliquant sur cette transition nous obtenons la fenêtre suivante.

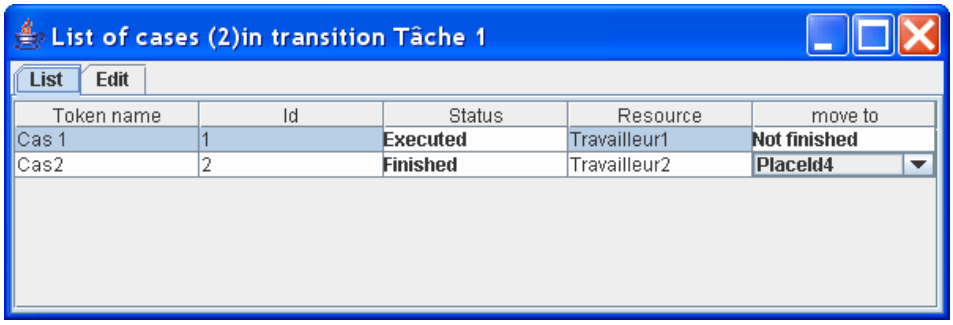


Figure 57. Mise à jour d'une liste des cas appartenant à une transition.

L'exemple se trouvant dans la fenêtre est l'illustration des deux situations auxquelles sont confrontés les cas se trouvant dans une transition.

- Un premier cas (id n°1) dont la tâche est en cours d'exécution, son statut est « executed », la ressource indiquée est le travailleur1 et le champs « move to » indique « not finished ». Cela signifie que le jeton ne peut quitter la transition puisque l'exécution de la tâche est en cours.
- Un second cas dont le travail à réaliser pour la tâche est terminé (status : « finished »). Il est donc désormais possible de déplacer le jeton vers la ou les prochaines places, qui seront déterminées par les patterns de routage pour le splitting de la transition dans laquelle se trouve actuellement le jeton.

3.3.2.2 Nombre de cas et goulot d'étranglement

Pour aider les responsables en leur permettant de prendre des décisions sur base du dessin du workflow, on associe toute transition à deux valeurs paramétrables. Ces deux seuils sont des nombres de cas se trouvant dans la transition. Si ce nombre atteint le premier seuil, la transition sera colorée en orange. Et en rouge dans le cas du second. Il s'agit donc d'une visualisation des goulots d'étranglements.

La paramétrisation de ces deux seuils se fait à l'aide de la fenêtre suivante.

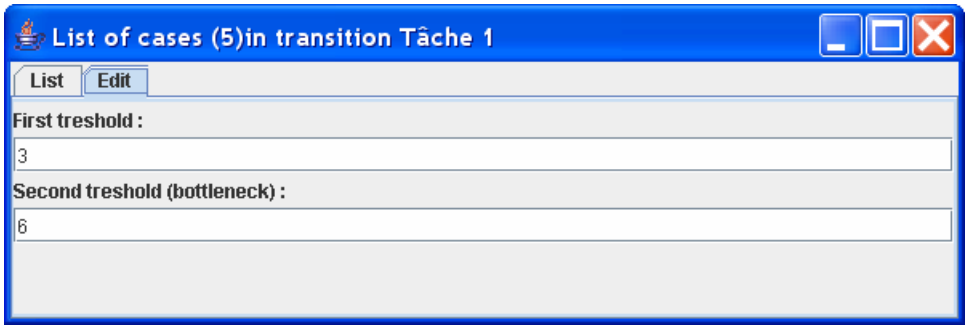


Figure 58. Spécification de seuils.

La figure suivante représente un workflow possédant les trois niveaux possibles.

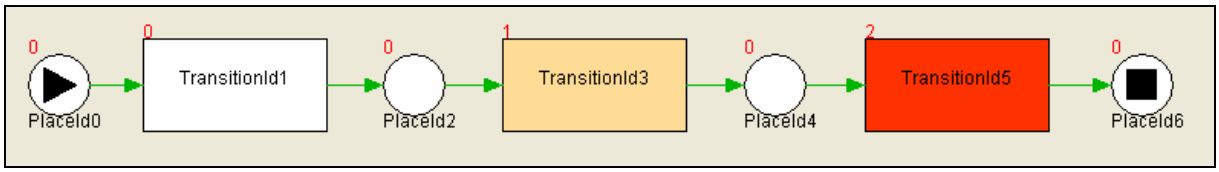


Figure 59. Trois états possibles d'une transition.

Les seuils étant ici fixé à 1 et 2. La première transition ne contient pas de jeton et n'est donc pas colorée. La seconde atteint le premier seuil et est en orange. Quand à la troisième les deux jetons lui appartenant font qu'elle égale le second seuil et donc la couleur rouge est signe que le gestionnaire du workflow doit être attentif à la situation.

3.3.2.3 Agenda du manager

Il s'agit d'un résumé de l'ensemble des jetons, indiquant dans quelle unité organisationnelle ils se trouvent et dans quelle place ou transition. Cela offre une vue générale, sous la forme de la fenêtre ci-dessous.

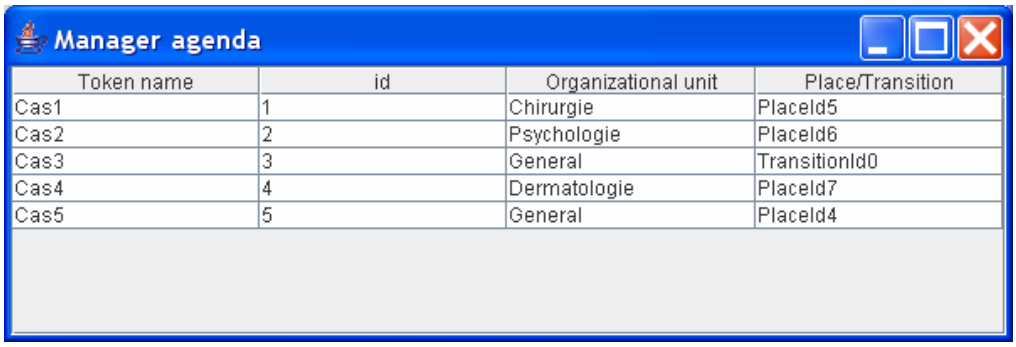


Figure 60. Agenda du manager.

3.3.2.4 Assignation des ressources (resource mapper)

C'est ici que se fait l'assignation du travail aux ressources. Chaque fois qu'une tâche doit être réalisée pour un cas donné nous aurons recours à ce gestionnaire de ressources. A ce stade le jeton se trouve encore dans la place reliée à la tâche à venir. Ce ne sera que plus tard, lorsque la tâche sera en cours d'exécution et deviendra par là même une activité, que le jeton sera déplacé dans la transition contenant la tâche.

Lorsque l'utilisateur clique sur le bouton « resource mapper » accessible via l'agenda correspondant à la transition, il obtient alors la fenêtre d'assignation des ressources suivante.

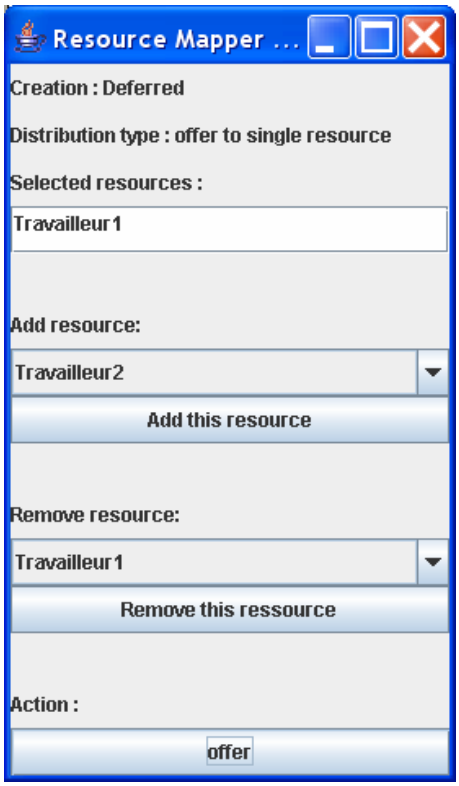


Figure 61. Assignation de ressources.

La première information est le pattern de création choisi. Il restreint le nombre de travailleurs pouvant effectuer la tâche. Ensuite nous avons le type de distribution : une offre ou une allocation, et dans le cas d'une offre elle pourra être propagée à plusieurs travailleurs. Le champ « selected resources » indique la liste de ressources courante. Il est possible d'ajouter et de supprimer des ressources de cette liste, le programme ne permettant bien sûr que l'ajout de travailleurs répondant aux critères établis par le pattern de création.

Une fois que l'utilisateur a indiqué sa liste définitive de ressources il clique sur le bouton « offer » dans le cas d'une offre ou « allocation » si il s'agit de ce second cas de figure. Le travailleur est alors averti de cette assignation. Cela fait l'objet de la section suivante.

3.3.2.5 Bureau de la ressource (resource desktop)

Cette fenêtre permet la simulation de la communication d'assignation du travail aux ressources. En situation réelle, ces fenêtres se trouveraient distribuées parmi les ordinateurs

des travailleurs et l'information serait propagée par un réseau. Dans notre cas le but était de mettre en lumière la façon dont un travailleur réagit à l'offre ou à l'allocation du travail, et dont l'état d'un objet de travail (l'association d'un jeton et d'une tâche particulière) évolue au cours du temps. Nous pouvons donc voir tout le cheminement du travail depuis sa création en passant par la distribution du travail aux ressources, son exécution et sa terminaison.

La fenêtre permettant au travailleur de résumer son rapport aux différents cas et aux tâches associées est la suivante (figure 62). Il s'agit d'une liste de travail, ou worklist.

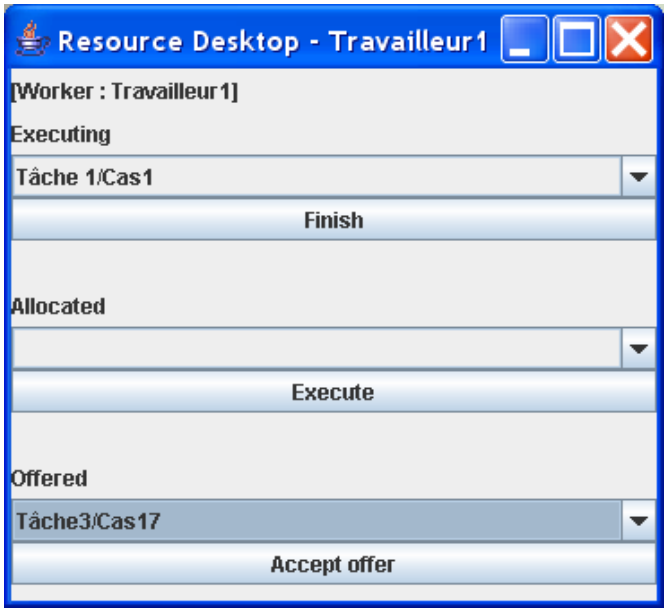


Figure 62. Bureau de la ressource.

En bas se trouvent les tâches qui lui sont offertes. En l'occurrence on voit qu'il s'agit de la tâche3 pour le cas17. En cas d'offre multiple, toutes les fenêtres contenant des bureaux de travailleurs concernés par l'offre seront de la même configuration. Le premier travailleur acceptant enverra un signal au système par le biais du bouton « accept offer » et l'information se propagera de sorte que l'offre ne soit plus disponible pour les autres.

Lorsqu'un objet de travail lui est alloué (allocated), la simulation de son exécution se fait grâce au bouton « execute ». C'est également cela qui informe le système de la modification de l'état, ce qui lui permettra de changer les données reprises dans les autres fenêtres afin d'être cohérentes avec les modifications réalisées.

Enfin, le dernier volet est consacré au travail en cours, la ressource peut à ce stade informer le système de la terminaison de ce travail, ce qui permettra au cas de poursuivre sa route dans le workflow selon les conditions établies par les patterns de routage.

3.4 Architecture de l'implémentation

Le langage utilisé pour réaliser l'implémentation du logiciel de ce mémoire est Java. Il s'agit d'un langage portable, qui peut être utilisé dans un environnement windows, linux, mac os ou encore unix. Java est orienté objet. Un objet représente un concept, une idée ou toute entité du monde physique. Il est doté d'attributs et de méthodes définissant les actions qui lui sont applicables. La spécification d'un ensemble d'objets similaire est la classe.

Dans les sections suivantes nous allons définir les classes principales du programme ainsi que les relations qui les lient. Nous commençons par le niveau le plus haut : celui des packages.

3.4.1 Répartition en packages

L'implémentation repose sur trois regroupements de classes, appelés packages :

- Package Elements
- Package WorkflowEditor
- Package CaseManager

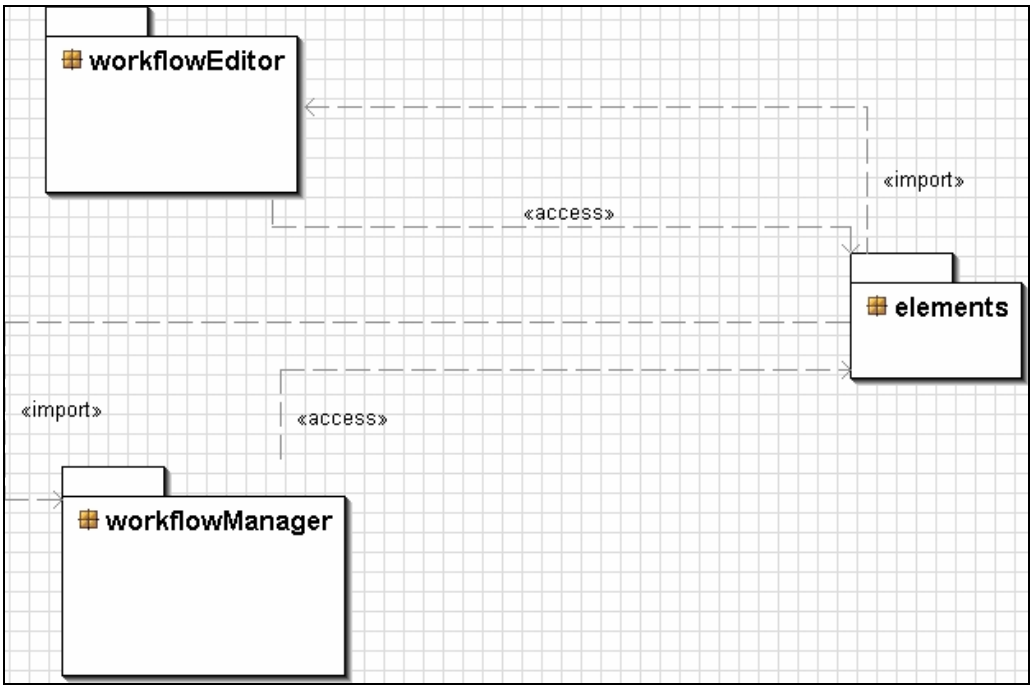


Figure 63. Répartition en packages.

Nous allons détailler le contenu de ces packages dans les sections suivantes, en expliquant leurs relations. Nous montrerons l'objectif poursuivi par chacune des classes composant les différents packages.

3.4.2 Package Elements

Dans ce package se trouvent toutes les classes qui seront partagées par l'éditeur de workflow et le gestionnaire des cas. Nous allons les décrire ainsi qu'expliquer leurs relations. Afin d'obtenir un schéma lisible les attributs et méthodes ont été enlevées.

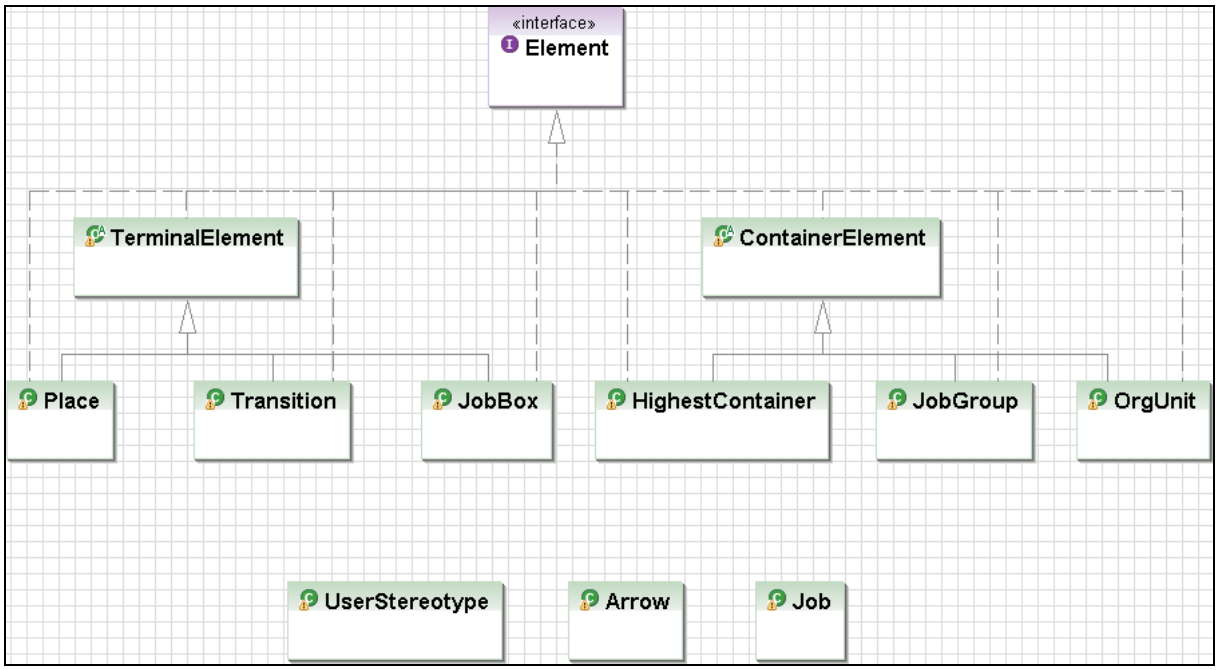


Figure 64. Package Elements.

La structuration reprend différents groupes :

- Une interface générale : Element

Toutes les classes qui correspondent à un objet graphique servant au dessin du workflow implémentent cette interface.

- Deux classes de structuration : TerminalElement et ContainerElement

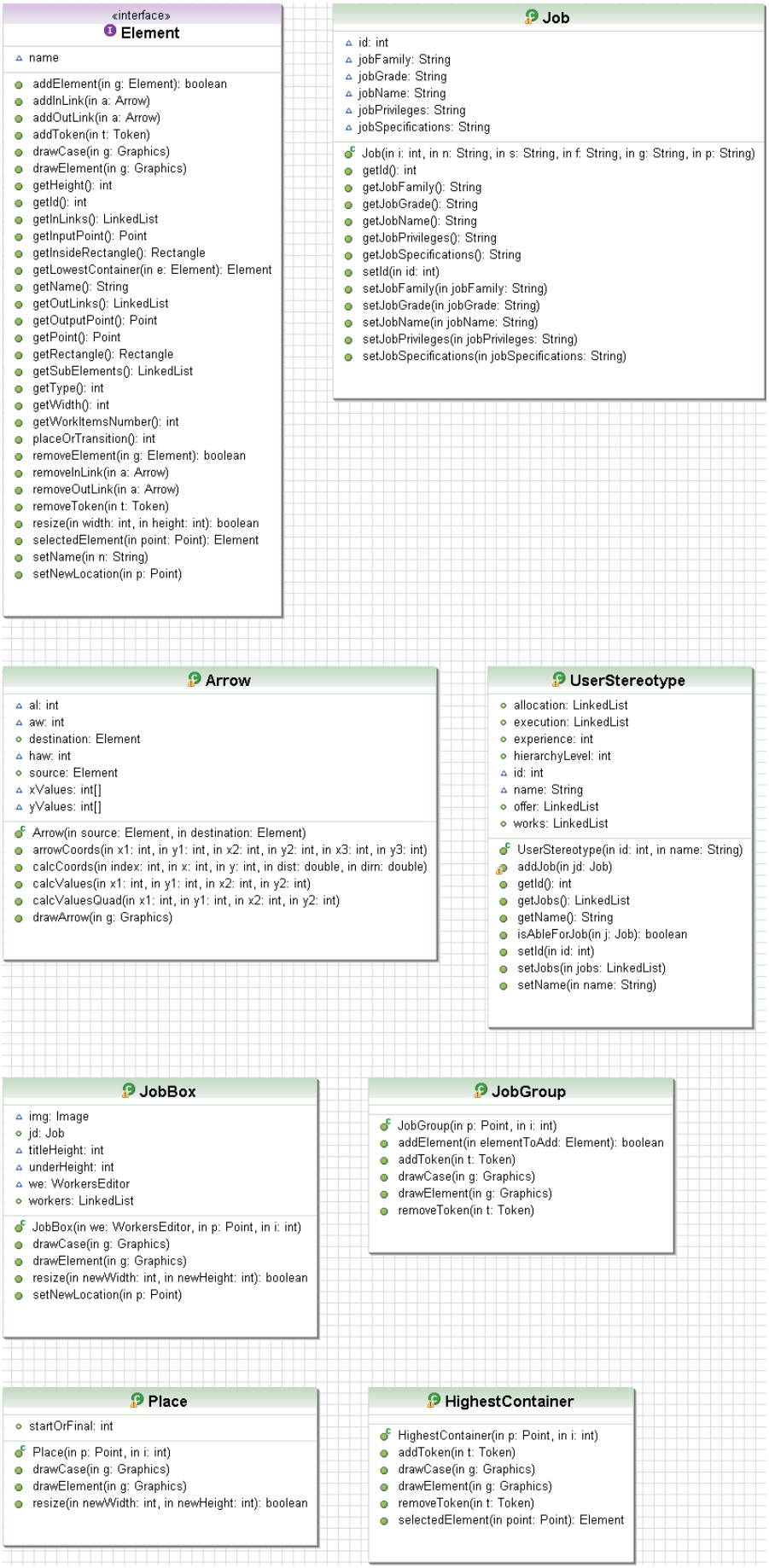
Cette séparation permet de travailler différemment selon qu'un élément est lui-même un container d'autres éléments ou non. Les éléments terminaux sont les places, transitions et boîtes à jobs (JobBox). Ils héritent donc de la classe TerminalElement.
Par contre, JobGroup (container à boîtes à jobs), OrgUnit (unité organisationnelle) et HighestContainer (le container général, qui est celui de plus haut niveau) héritent logiquement de la classe ContainerElement.

- Eléments isolés

La classe UserStereotype sert à modéliser les travailleurs. Arrow crée un lien dirigé entre une place et une transition. Quand à Job il s'agit de la classe représentant un métier. Ces classes n'ont pas de représentation graphique de type élément et n'implémentent donc pas l'interface du même nom.

Les attributs et méthodes de ces classes se trouvent sur les figures suivantes.

Figure 65. Classes du package Elements


«interface»
Element
name
addElement(in g: Element): boolean
addInLink(in a: Arrow)
addOutLink(in a: Arrow)
addToken(in t: Token)
drawCase(in g: Graphics)
drawElement(in g: Graphics)
getHeight(): int
getId(): int
getInLinks(): LinkedList
getInputPoint(): Point
getInsideRectangle(): Rectangle
getLowestContainer(in e: Element): Element
getName(): String
getOutLinks(): LinkedList
getOutputPoint(): Point
getPoint(): Point
getRectangle(): Rectangle
getSubElements(): LinkedList
getType(): int
getWidth(): int
getWorkItemsNumber(): int
placeOrTransition(): int
removeElement(in g: Element): boolean
removeInLink(in a: Arrow)
removeOutLink(in a: Arrow)
removeToken(in t: Token)
resize(in width: int, in height: int): boolean
selectedElement(in point: Point): Element
setName(in n: String)
setNewLocation(in p: Point)
Job
id: int
jobFamily: String
jobGrade: String
jobName: String
jobPrivileges: String
jobSpecifications: String
Job(in i: int, in n: String, in s: String, in f: String, in g: String, in p: String)
getId(): int
getJobFamily(): String
getJobGrade(): String
getJobName(): String
getJobPrivileges(): String
getJobSpecifications(): String
setId(in id: int)
setJobFamily(in jobFamily: String)
setJobGrade(in jobGrade: String)
setJobName(in jobName: String)
setJobPrivileges(in jobPrivileges: String)
setJobSpecifications(in jobSpecifications: String)
Arrow
al: int
aw: int
destination: Element
haw: int
source: Element
xValues: int[]
yValues: int[]
Arrow(in source: Element, in destination: Element)
arrowCoords(in x1: int, in y1: int, in x2: int, in y2: int, in x3: int, in y3: int)
calcCoords(in index: int, in x: int, in y: int, in dist: double, in dirn: double)
calcValues(in x1: int, in y1: int, in x2: int, in y2: int)
calcValuesQuad(in x1: int, in y1: int, in x2: int, in y2: int)
drawArrow(in g: Graphics)
UserStereotype
allocation: LinkedList
execution: LinkedList
experience: int
hierarchyLevel: int
id: int
name: String
offer: LinkedList
works: LinkedList
UserStereotype(in id: int, in name: String)
addJob(in jd: Job)
getId(): int
getJobs(): LinkedList
getName(): String
isAbleForJob(in j: Job): boolean
setId(in id: int)
setJobs(in jobs: LinkedList)
setName(in name: String)
JobBox
img: Image
jd: Job
titleHeight: int
underHeight: int
we: WorkersEditor
workers: LinkedList
JobBox(in we: WorkersEditor, in p: Point, in i: int)
drawCase(in g: Graphics)
drawElement(in g: Graphics)
resize(in newWidth: int, in newHeight: int): boolean
setNewLocation(in p: Point)
JobGroup
JobGroup(in p: Point, in i: int)
addElement(in elementToAdd: Element): boolean
addToken(in t: Token)
drawCase(in g: Graphics)
drawElement(in g: Graphics)
removeToken(in t: Token)
Place
startOrFinal: int
Place(in p: Point, in i: int)
drawCase(in g: Graphics)
drawElement(in g: Graphics)
resize(in newWidth: int, in newHeight: int): boolean
HighestContainer
HighestContainer(in p: Point, in i: int)
addToken(in t: Token)
drawCase(in g: Graphics)
drawElement(in g: Graphics)
removeToken(in t: Token)
selectedElement(in point: Point): Element

3.4.3 Package workflowEditor

Dans ce package se trouvent les classes qui sont utilisées par l'éditeur de workflow, à l'exception de celles provenant du package Elements évoqué plus haut. La classe WorkflowEditor en est le point central, créant les instances des autres classes.

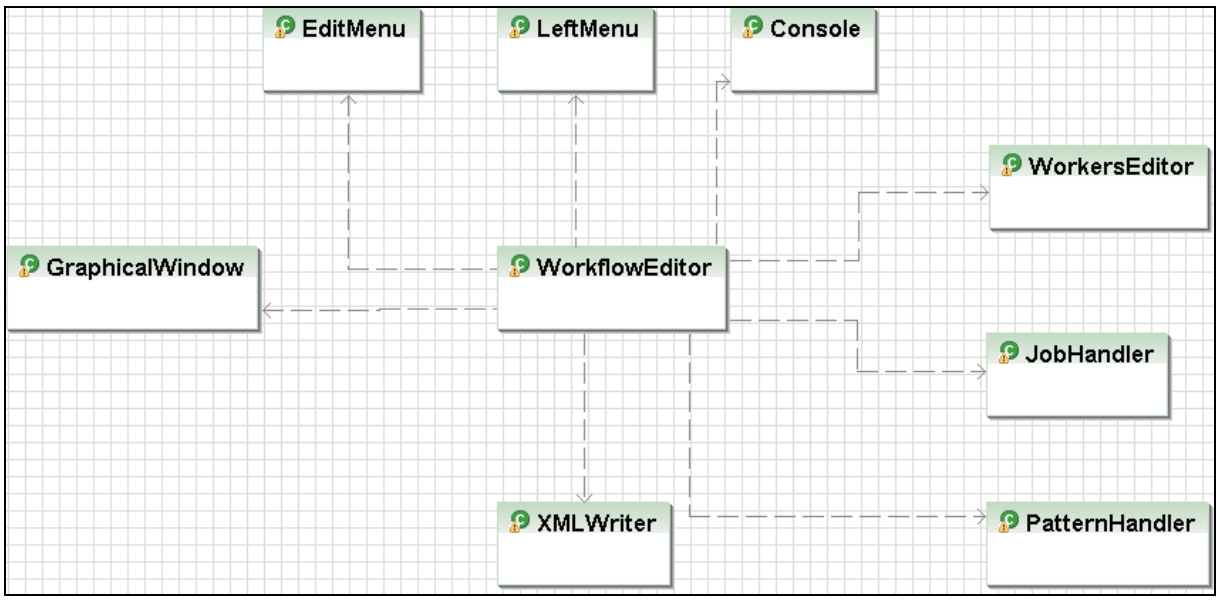


Figure 66. Package workflowEditor.

Nous décomposons alors l'architecture en plusieurs groupes :

- Les classes d'entrée/sortie qui sont implémentées par les classes suivantes
  - EditMenu – classe permettant de définir les attributs de chacun des éléments graphiques une fois sélectionnés. Il s'agit du menu apparaissant à droite de l'éditeur.
  - LeftMenu – par ce menu l'utilisateur a accès aux opérations consistant à ajouter les éléments graphiques, les lier, redimensionner éditer et supprimer.
  - Console – classe permettant d'afficher des messages, il s'agit de la partie sud de la fenêtre principale de l'éditeur de workflow.
- La tablette graphique : classe GraphicalWindow. Il s'agit de la classe gérant tout le contenu graphique. Elle est détentrice de l'ensemble des éléments et c'est donc là que se trouvent toutes les méthodes relatives aux sauvegardes et à la mise en relation d'éléments (liaison par une flèche par exemple).
- Les gestionnaires :
  - WorkersEditor – permet la création et l'édition de travailleurs via une fenêtre spécifique.
  - JobHandler – sert à définir les jobs via une fenêtre dédiée à tous les aspects de création et modification d'un job.
  - PatternHandler – lors de l'édition d'une transition il est possible de choisir les patterns de ressource qui lui seront associés dans une fenêtre spécifique.
  - CTT éditeur – n'appartient pas à proprement parler aux classes car il s'agit de l'import d'un fichier .jar
- L'écriture du fichier XML : XMLWriter.

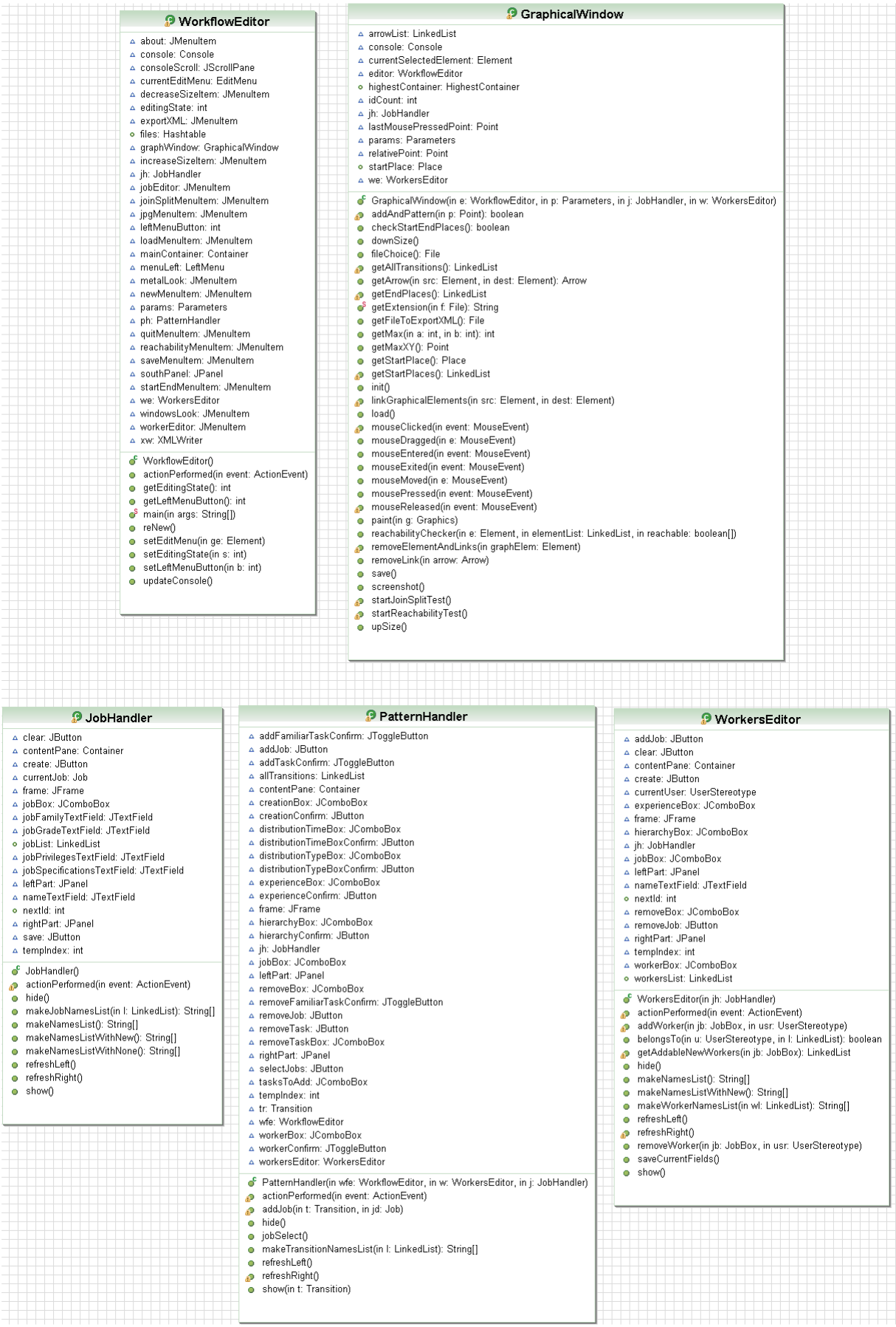


Figure 67. Classes du package workflowEditor (1).

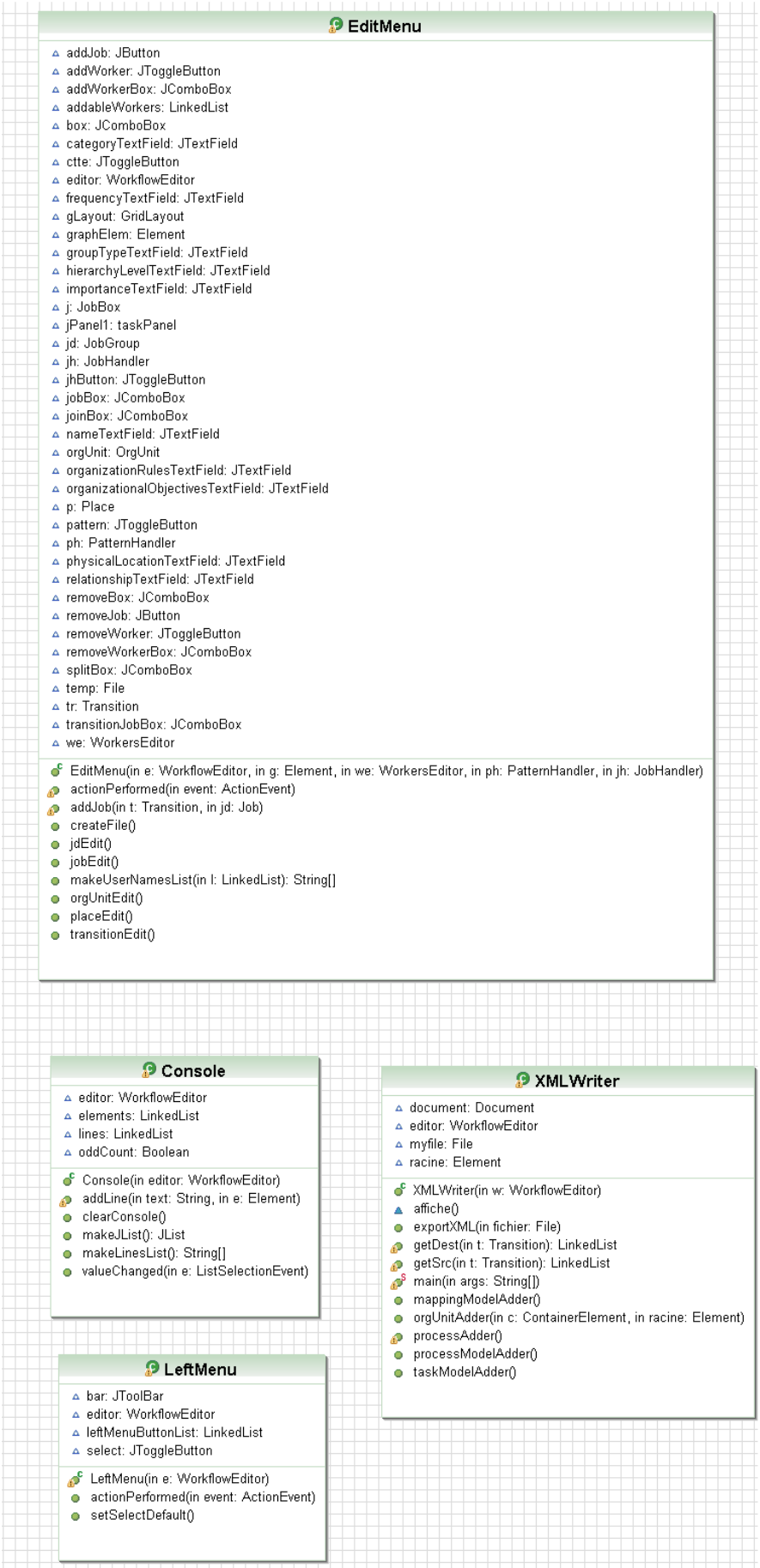


Figure 68. Classes du package workflowEditor (2).

3.4.4 Package workflowManager

Ce package fait le regroupement des classes servant à définir la partie dynamique du programme, celle de la gestion des cas. Le point central est ici la classe CaseManager. Comme indiqué sur le schéma suivant, elle possède des instances de CaseManagerFrame, GraphicalAgenda et ResourceManager.

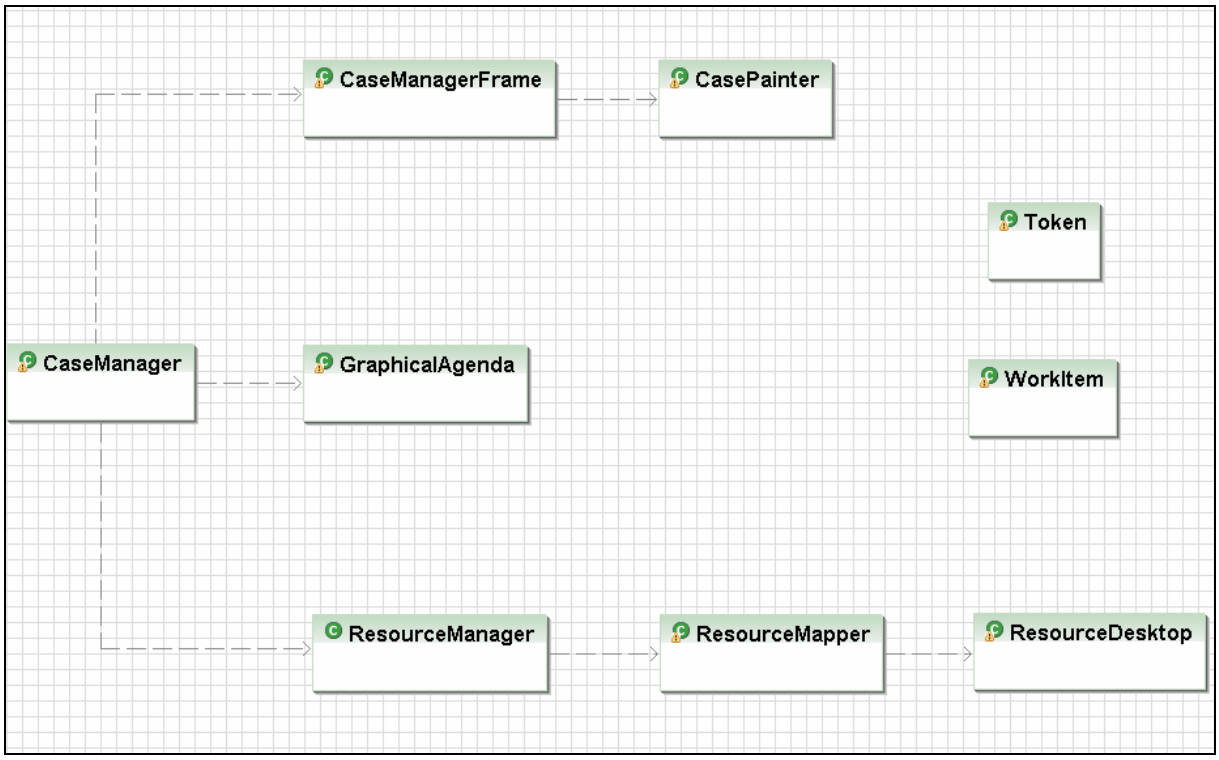


Figure 69. Package workflowManager.

Nous pouvons détailler trois groupes architecturaux :

- Les classes permettant de représenter le workflow tel qu'il a été défini dans l'éditeur :
  - CaseManagerFrame est implémente la fenêtre contenant le dessin du workflow, et la gestion des clics effectués.
  - CasePainter gère l'affichage de ce dessin.
- Le gestionnaire d'agendas : GraphicalAgenda. Lorsque l'utilisateur clique sur une place ou une transition, une fenêtre s'ouvre indiquant les jetons qui lui appartiennent et les actions possibles qui lui sont associées. Cette classe gère aussi l'agenda du manager, vue d'ensemble de tous les cas cheminant au sein du workflow.
- Le gestionnaire de ressources : ResourceManager.
  - Il possède une instance de ResourceMapper, classe chargée de l'attribution du travail aux ressources sur base de l'utilisation des patterns de ressource.
  - Pour réaliser une simulation d'interaction avec un travailleur, la classe ResourceDesktop implémente la liste de travail (worklist) et permet d'accepter un travail offert, d'exécuter un travail alloué et de signaler la fin de ce travail.
- Les classes Token et WorkItem, représentant respectivement un jeton et la réalisation d'un objet de travail.

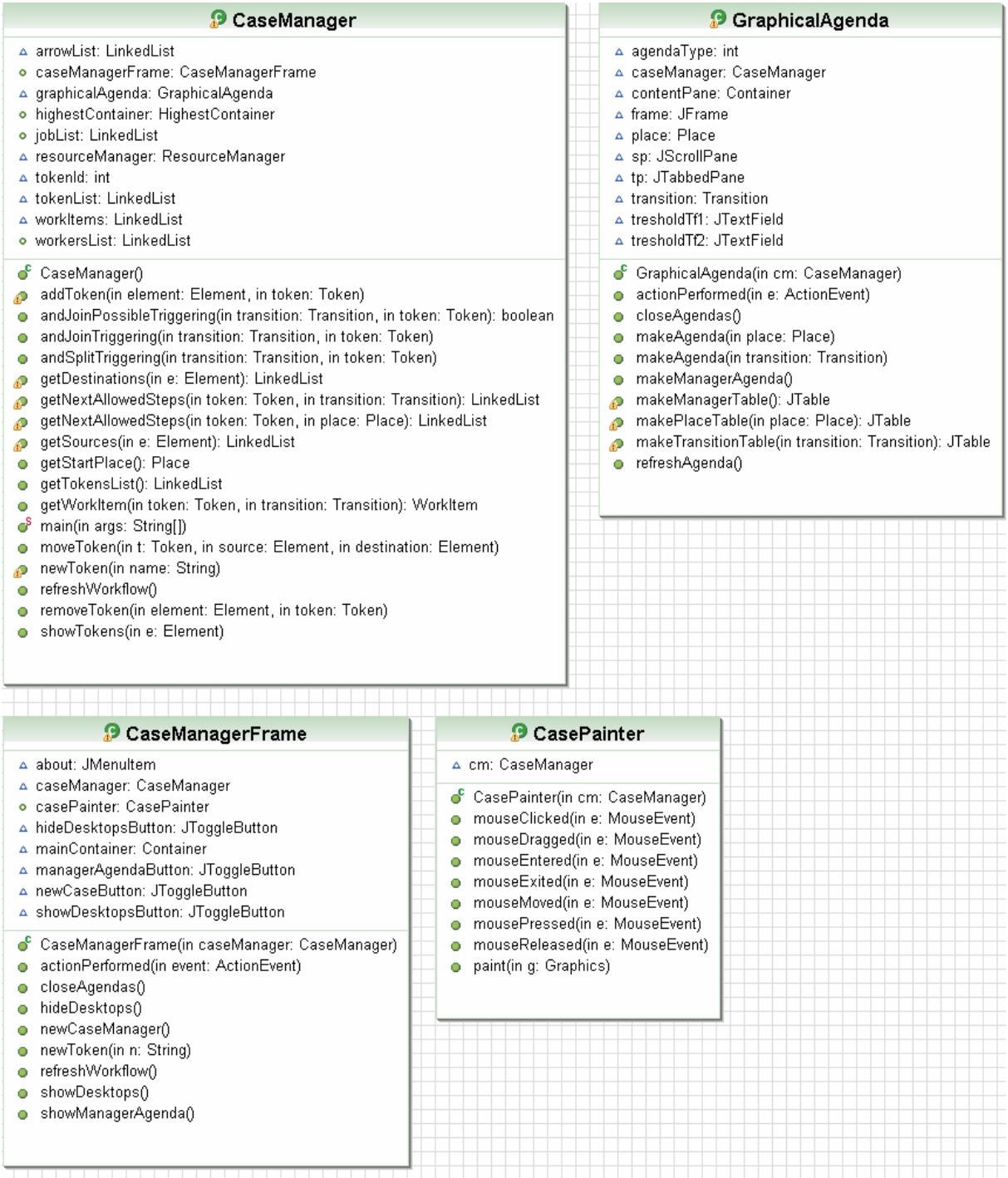


Figure 70. Classes du package workflowManager (1).

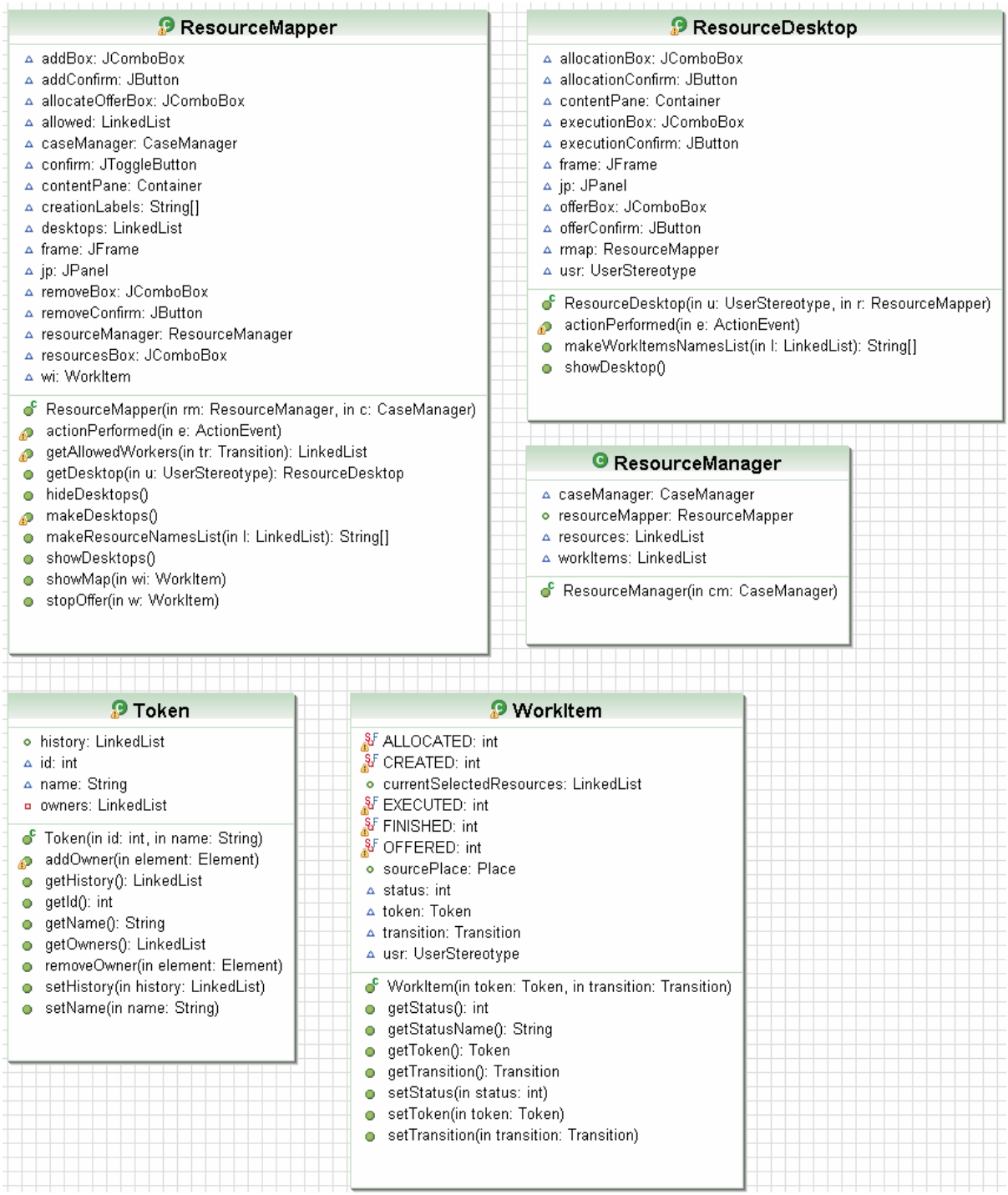


Figure 71. Classes du packages workflowManager (2).

3.5 Effort de l'implémentation

L'implémentation a été une partie conséquente de la réalisation de ce mémoire. L'aspect graphique, réalisé en Java/Swing nécessite de se former à l'API (application program interface) correspondante. De plus les attentes en terme de logiciel, que ce soit du point de vue graphique ou de son utilisation, ont nécessité de créer des objets graphiques inédits. Il était important de pouvoir imbriquer les éléments graphiques, les redimensionner, déplacer et

lier ainsi que de pouvoir les superposer. Cela offre au final une représentation « user-friendly », aspect important lorsque l'on développe ce type d'outil. Le logiciel a été optimisé afin de ne pas souffrir des lenteurs auxquelles font face de nombreux outils d'édition graphique développés en java. J. Guerrero et J. Montero ont apporté de nombreux conseils afin d'améliorer la prise en main du logiciel par l'utilisateur et l'aspect graphique de celui-ci.

Le fait de pouvoir utiliser effectivement le workflow défini via l'éditeur pour tracer l'exécution de cas a nécessité d'interpréter la spécification du workflow.

Le développement d'un éditeur graphique passe par une approche globale, basée sur l'utilisation de plusieurs techniques de programmation (parsing, algorithmes récursifs, interfaces graphiques, listeners, sérialisation, …) La phase d'implémentation de l'application a donc pris plusieurs mois. Le code (6000 lignes) a été optimisé afin de disposer de méthodes génériques aussi réutilisables et évolutives que possible.

Chapitre 4 Etude de cas

Ce chapitre se consacre à l'utilisation du programme dans le cadre de la modélisation d'un cas concret. Nous allons voir comment fonctionne le programme, depuis la définition des entités formant le workflow jusqu'à son utilisation dans le cadre d'une simulation gérée par la seconde partie du logiciel implémenté.

Nous utiliserons la modélisation d'un hôpital, ses services, les ressources qui s'y trouvent et les processus qui y prennent place. L'utilisation du workflow dans le système de gestion permettra de créer le cas d'un patient se rendant à l'hôpital et nécessitant une prise en charge.

4.1 Unités organisationnelles et ressources

La première étape consiste à déterminer les unités organisationnelles de l'hôpital. Comme nous l'avons vu dans le chapitre 2, ces unités sont le lieu où prennent place les processus, et sont dotés de ressources en charge du travail. Nous utilisons une répartition classique en services.

- ❖ Accueil : lorsqu'un patient arrive à l'hôpital, il se rend au guichet de ce service. C'est là qu'il sera dispatché vers l'une des autres unités organisationnelles afin d'être traité ou de prendre un rendez-vous.
- ❖ Centre de dépistage : cette unité a pour objectif le dépistage de certaines affections dont souffrent les patients, et qui nécessitent une approche spécifique. Trois types de tests y sont effectués, ceux propres à la détection d'allergies, de maladies infectieuses et de carences alimentaires.
- ❖ Dermatologie : les patients ayant des problèmes dermatologiques s'y rendent, pour obtenir la consultation d'un spécialiste ou une opération de petite chirurgie. Il s'agit d'opérations légères ne nécessitant pas d'anesthésie telles que l'ablation d'un grain de beauté ou la correction d'un ongle incarné.
- ❖ Médecine générale : dans ce département ont lieu des consultations, actes médicaux légers et des mesures biométriques telles que la mesure de la pression sanguine.
- ❖ Chirurgie : il s'agit de l'unité dans laquelle les patients sont opérés. Ils y seront anesthésiés, opérés et alités afin d'être suivis durant le temps post-opératoire.
- ❖ Gestion des rendez-vous : c'est là que les rendez-vous sont pris, et les démarches administratives et médicales nécessaires sont effectuées.
- ❖ Service financier : dans ce service, le patient effectue un paiement pour la prise en charge médicale dont il a fait l'objet.

Les ressources chargées d'effectuer les tâches sont réparties en trois familles. En premier lieu celle du domaine médical. On retrouve des médecins répartis selon leur spécialisation : généralistes, allergologues, dermatologues, chirurgiens et anesthésistes. Outre les médecins, l'hôpital emploie des infirmières. En dehors du cadre médical, des personnes effectuant des tâches de type administratif, financier et d'accueil travaillent pour l'hôpital. Il s'agit d'hôtesses, de secrétaires et de caissiers.

Au centre de dépistage travaillent des infirmières, allergologues et généraliste. En dermatologie un spécialiste de la discipline, de même pour l'unité de médecine générale. L'unité chirurgicale est dotée d'anesthésistes, chirurgiens et infirmières. La gestion des rendez-vous emploie des secrétaires. Enfin, une ou plusieurs hôtesses se trouvent à l'accueil et des personnes chargées de l'encaissement dans le service financier.

Si nous nous centrons sur l'unité organisationnelle de dépistage, nous commençons par l'insérer graphiquement grâce au menu d'éléments graphiques. Le nom de « Centre de dépistage » lui est donné et ses attributs sont définis.

- Objectifs : réaliser le dépistage d'affections dont souffre le patient.
- Group type : domaine médical.
- Localisation : bâtiment principal, $3^{ème}$ étage.
- Niveau hiérarchique : moyen.

- Relations : indépendance vis-à-vis des autres unités organisationnelles.
- Règles : règles spécifiques à l'hygiène et aux restrictions d'accès des personnes.

Nous définissons alors les ressources. En premier lieu, les jobs sont définis en utilisant le Job Handler disponible dans le menu situé au nord du logiciel. La figure ci-dessous (à gauche) montre la spécification du job d'infirmière. Lorsque les jobs sont encodés, nous pouvons définir les travailleurs étant capable d'accomplir ce job. Dans notre exemple (voir figure suivante - partie droite), il s'agit de Sophie Grandjean, ayant trois ans d'expérience et un niveau hiérarchique d'infirmière en chef (niveau 2).

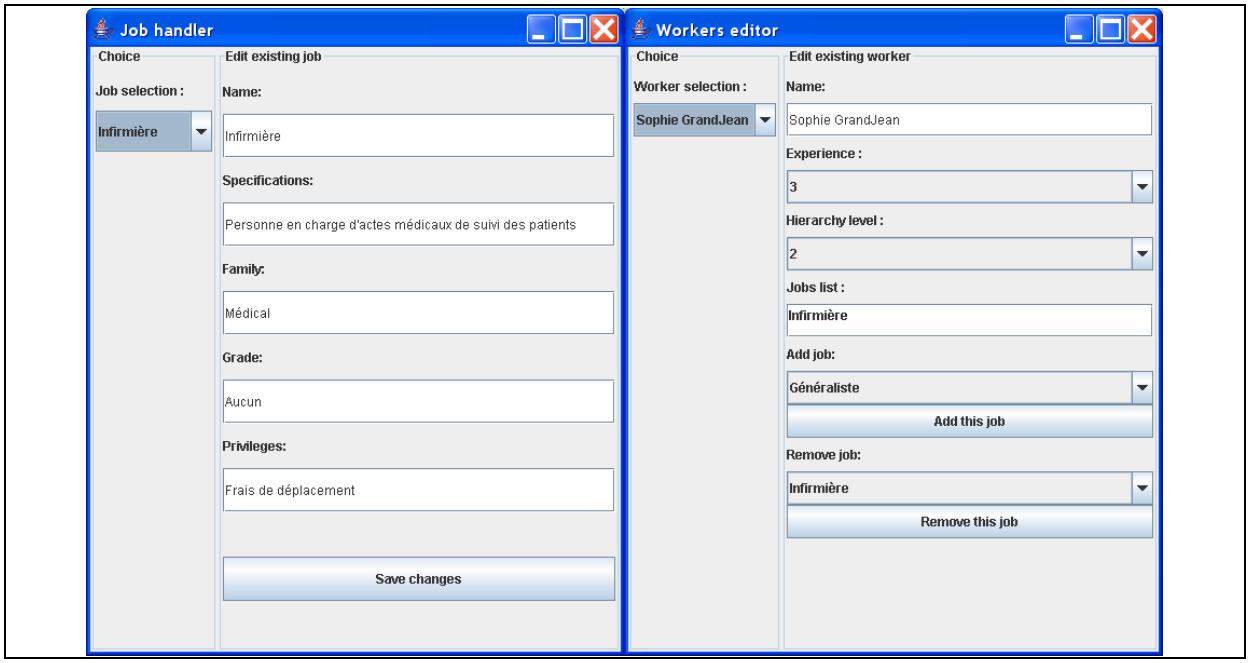


Figure : gestionnaire de métiers et éditeur de travailleur.

Maintenant que les ressources sont définies, nous pouvons placer une boîte à ressource correspondant au job d'infirmière dans notre unité organisationnelle. Pour cela nous insérons une boîte puis nous choisissons le job qui la concernera. Enfin, nous instancions cette boîte avec l'infirmière Sophie GrandJean, ce qui a pour effet d'incrémenter le chiffre dans la boîte à ressource.

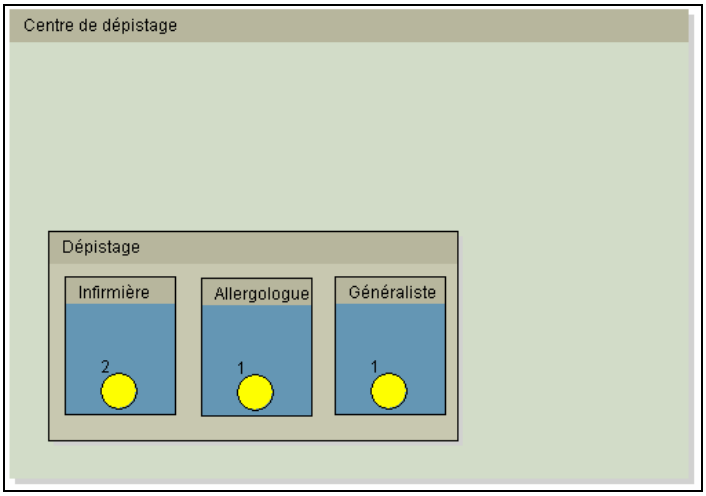


Figure 72. Unité organisationnelle "Centre de dépistage".

4.2 Spécification des processus

A ce stade nous définissons les tâches effectuées dans chacune des unités organisationnelles, ainsi que leur agencement.

Dans le centre de dépistage, la première tâche est la prise de sang. Ensuite il est possible de réaliser l'un des trois tests suivants : allergies, maladies infectieuses, carences. Il est également possible de réaliser les trois. Pour terminer, une prescription médicale a lieu. Le pattern de séparation de la prise de sang est un And/Or, de même que pour la jointure de la tâche de prescription.

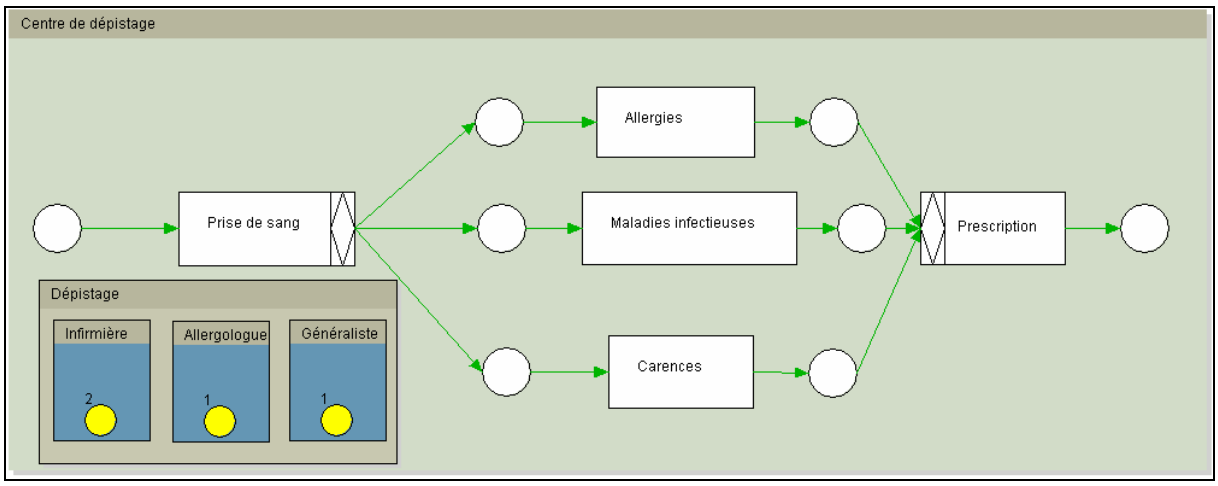


Figure 73. Processus du Centre de dépistage.

Un patient arrivant en dermatologie sera d'abord dirigé soit vers une opération de petite chirurgie si il avait un rendez-vous le permettant, soit en consultation. Suite à cette consultation il recevra une prescription.

La première tâche de l'unité de médecine générale est une consultation. Ensuite il est possible de réaliser un check-up complet, d'effectuer un acte médical ou aucune de ces deux tâches. Pour finir, le médecin réalise une prescription qu'il remet au patient.

L'unité de chirurgie contient un processus entièrement séquentiel. Les tâches sont effectuées les unes à la suite des autres : l'anesthésie, l'opération, la réanimation et enfin l'hospitalisation.

Le service de gestion des rendez-vous a pour première tâche la prise de rendez-vous avec le patient, ce qui permettra d'obtenir les données qui y sont relatives. Ensuite deux tâches sont effectuées en parallèle, l'enregistrement des informations qui mettre à jour le planning de l'hôpital et l'envoi d'un courrier de confirmation.

L'accueil et le paiement sont les tâches uniques de leurs unités respectives.

Lorsque nous plaçons toutes les unités organisationnelles sur un même schéma, et que nous ajoutons les places de commencement et de terminaison, nous obtenons la représentation graphique du workflow (figure).

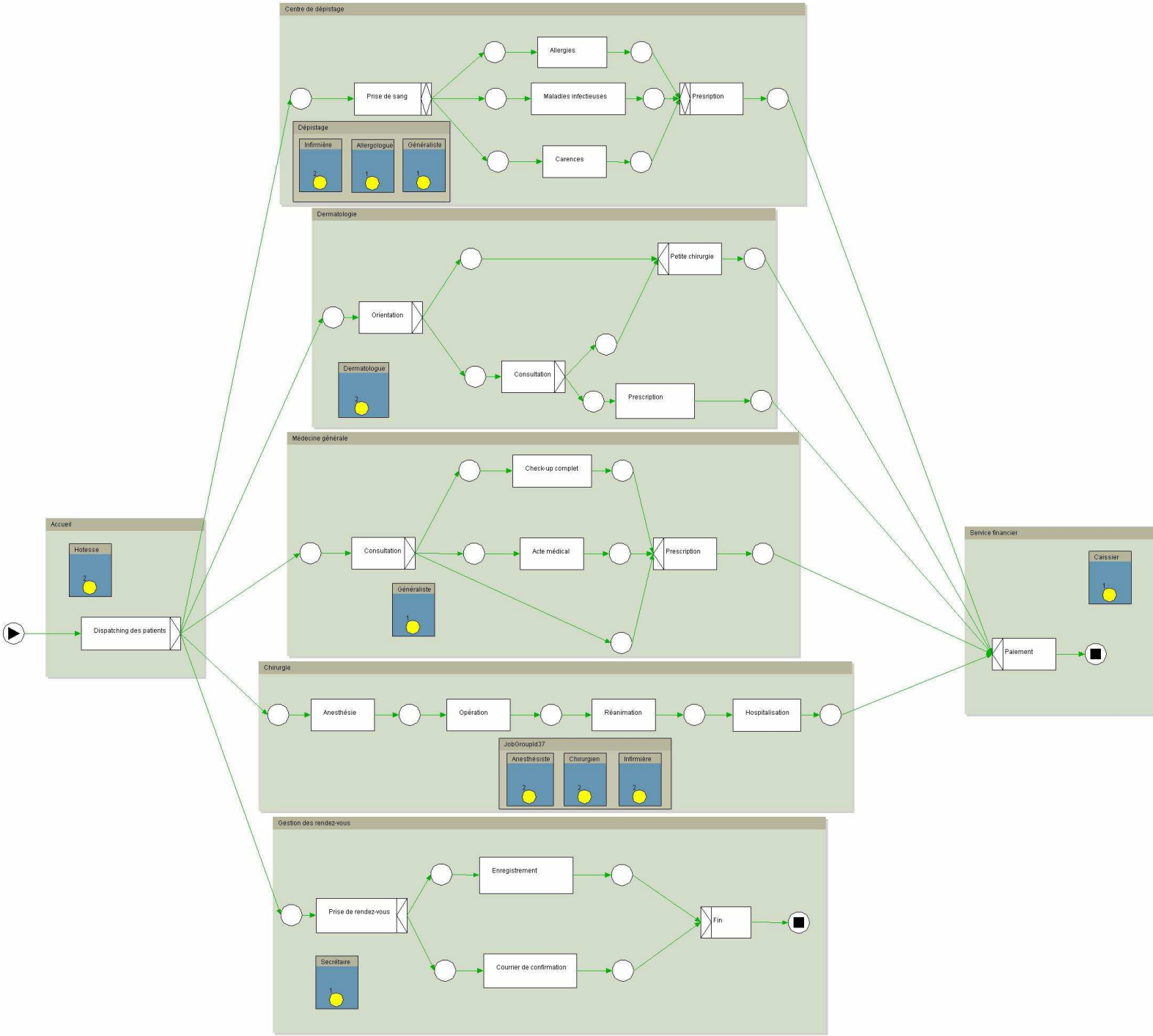


Figure 74. Schéma du workflow modélisant l'hôpital.

Les trois tests (accessibilité de tous les éléments du schéma, présence des places start/end et patterns de routage appropriés) sont lancés et se terminent en indiquant que le schéma est correct.

4.3 Arbre de tâche

A présent nous réalisons l'arbre de tâches concurrent pour chacune des tâches du schéma. Il s'agit de l'éditeur IdealXML. Nous allons donner l'illustration de la tâche de prise de sang réalisée au sein du centre de dépistage. Figure.

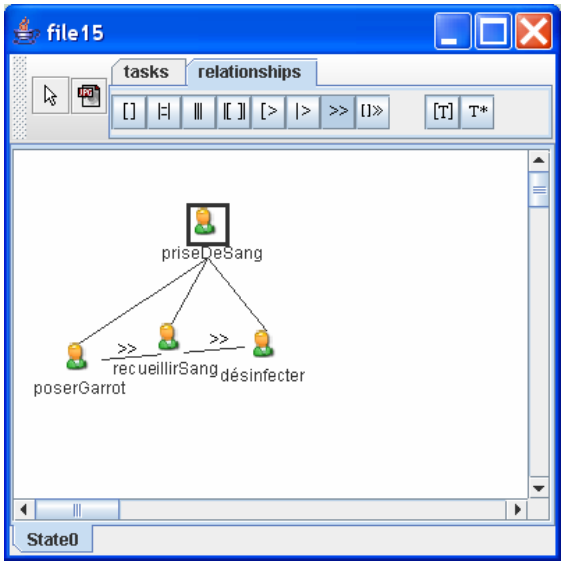


Figure 75. Arbre de la tâche de prise de sang.

La prise de sang est décomposée en trois sous-tâches : poser garrot, recueillir sang et désinfecter. La relation temporelle utilisée est l'activation, marquée >> et consistant à dire que les sous-tâches doivent ici être réalisées l'une après l'autre.

4.4 Utilisation des patterns de ressource

Il nous reste à définir les ressources qui pourront prendre en charge la réalisation des tâches. Pour cela nous avons recours au Resource patterns handler. Dans le cas présent de la tâche de prise de sans, le métier nécessaire pour sa réalisation est le job d'infirmière. Le pattern de création est celui basé sur la capacité : la ressource qui prendra en charge le travail aura une expérience minimum requise. Dans notre cas nous fixons une expérience d'au moins 3 ans. Le pattern de distribution consiste à offrir le travail à une ressource unique, quant à celui du temps de distribution il s'agit de la distribution lorsque la tâche commence. Figure.

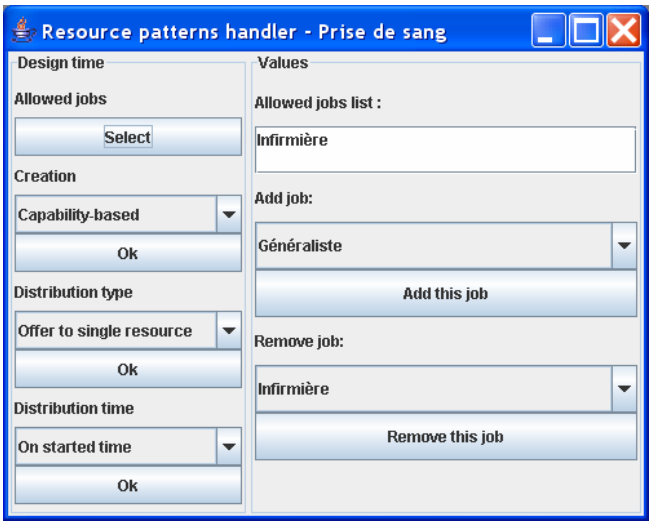


Figure 76. Gestionnaire des patterns de ressource pour la prise de sang.

4.5 Sauvegarde

La définition du workflow est maintenant terminée, il reste à le sauver dans le format Usi-XML et dans le format interne afin de pouvoir passer à la partie dynamique du programme. Cette dernière permet de réaliser une simulation de gestion du workflow défini, en réalisant le suivi des cas.

4.6 Utilisation du caseManager

Le caseManager, seconde partie du programme, est alors exécuté. Après avoir chargé la définition du workflow, un nouveau cas est créé au nom de « Mr Durand ». Ce patient se rend à l'hôpital pour effectuer un dépistage d'allergies. Il est orienté par l'accueil, ce qui mène le jeton représentant le cas à la première place appartenant à l'unité organisationnelle de dépistage. Un arc relie cette place à la tâche de la prise de sang. Ceci est résumé par la fenêtre qui s'ouvre lorsque l'utilisateur clique sur la place où se trouve le jeton (figure).

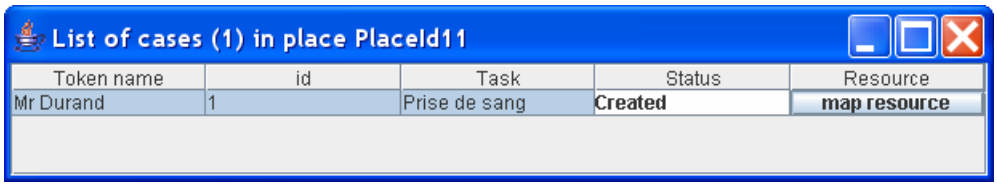


Figure 77. Jetons dans la place précédant la tâche de prise de sang.

4.7 Attribution des ressources

L'attribution des ressources peut alors se faire pour la réalisation de la tâche. Les ressources prises en considération sont celles qui se trouvent dans l'unité de dépistage. A savoir deux infirmières, un allergologue et un généraliste. Nous avons défini la tâche de prise de sang comme devant être réalisée par une infirmière, ce qui restreint l'ensemble des ressources. De plus, le pattern de création associé limite le choix aux ressources ayant plus de 2 ans d'expérience. Des deux infirmières, seule Sophie GrandJean remplit ce critère. La réalisation de la tâche lui est alors offerte via la fenêtre d'attribution des ressources (figure).

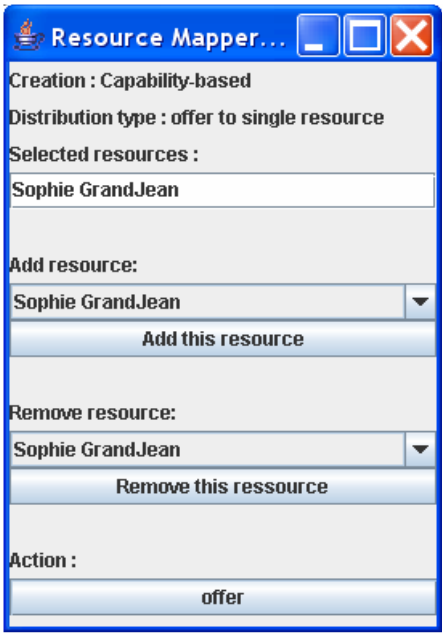


Figure 78. Sélection des ressources.

4.8 Bureau de la ressource

Le bureau de la ressource, qui permet la simulation de la liste de travail (work list) de l'infirmière Sophie GrandJean, est alors utilisé pour accepter l'offre, lancer l'exécution de la tâche et signaler sa terminaison (figure).

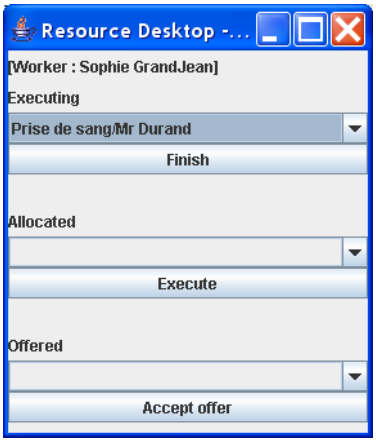


Figure 79. Liste de travail de la ressource.

Lorsque la tâche est terminée, le jeton peut être déplacé vers les places de destinations. Etant donné que la transition contenant la prise de sang est dotée du pattern de routage And/or, il est possible de déplacer le jeton vers l'une des trois places ciblées ou d'obtenir une instance du jeton à chacune des places.

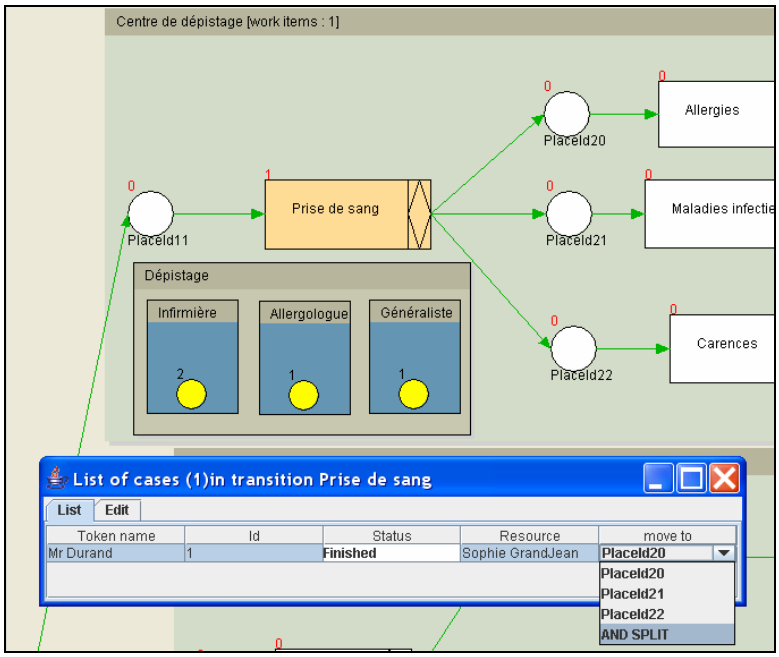


Figure 80. Gestionnaire des cas.

4.9 Enseignements de l'étude de cas

Cette étude de cas nous a montré l'ensemble des possibilités du programme via la modélisation d'un hôpital. Nous sommes partis de la spécification des unités organisationnelles, ainsi que de la définition des métiers et des ressources. Les processus ont ensuite été créés. Enfin, nous avons montré la manière dont chemine un cas au sein du workflow. L'attribution des ressources étant réalisée dynamiquement, en se basant sur la disponibilité des travailleurs disponibles et en satisfaisant aux conditions émises par les patterns de ressource.

Chapitre 5 Conclusion

Il est maintenant temps d'aborder la conclusion du mémoire. Nous allons dresser un bilan du travail réalisé. En premier lieu nous évoquerons les contributions réalisées. En termes conceptuels, par l'approche des workflows que nous avons sélectionnée, ainsi que dans le domaine de l'implémentation. Ces considérations feront l'objet de la première section.

Dans la seconde section nous évoquerons les possibles développements futurs. Nous verrons ce qu'il est possible d'améliorer ainsi que les ajouts qui pourraient compléter le logiciel réalisé.

La troisième et dernière section sera consacrée au mot final de ce mémoire.

5.1 Contributions de ce mémoire

Afin d'évoquer les contributions réalisées, nous allons rappeler l'objectif principal, provenant du premier chapitre. Son intitulé est le suivant :

> En vue de mettre en oeuvre un outil de workflow, un éditeur graphique de son comportement devrait permettre la génération aussi automatique que possible de ses spécifications fonctionnelles à partir desquelles une interface homme-machine centrée sur l'utilisateur peut être obtenue.

Le développement d'un outil de workflow a nécessité de spécifier l'approche qui fut utilisée dans le cadre de sa modélisation. C'est pourquoi la première partie du travail a consisté à déterminer la problématique de l'utilisation des outils de workflow actuels. Des problèmes importants tels que le manque de standardisation dans la définition des workflow ont été mis en lumière. Forts de ce constat, nous avons réalisé un état de l'art afin de dégager les points positifs et négatifs des programmes se trouvant sur le marché. Il en est ressorti qu'il existe en réalité un certain nombre de manières d'aborder d'une part la notion de workflow, et d'autre part la manière de le modéliser.

Nous avons alors effectué une sélection des modélisations aussi cohérente que possible, puis établi la jonction entre elles afin d'obtenir un produit logiciel homogène. Un modèle des tâches a été choisi, basé sur l'utilisation de l'arbre de tâche concurrent. Le modèle des processus s'appuie quant à lui sur l'utilisation des réseaux de Petri. Il permet d'établir les différentes routes composées de tâches qui seront parcourues par les cas. Enfin, le modèle de workflow nous a permis d'ajouter le concept de ressources et la manière dont elles sont attribuées à la performance du travail.

Cette définition du workflow se devait de favoriser une utilisation graphique des concepts impliqués. Dans cette optique, l'utilisation des réseaux de Petri et de l'arbre de tâche concurrent offre des représentations très parlantes à l'utilisateur. Les spécifications fonctionnelles en sont issues. On pense notamment aux patterns de routage, qui seront formellement déduits de la représentation graphique du workflow.

La génération d'interfaces hommes-machines centrées sur l'utilisateur est permise par l'utilisation du langage de spécification d'interfaces Usi-XML. La définition du workflow peut en effet être sauvée dans ce langage ou sous une forme spécifique à l'application.

L'implémentation de la partie dynamique du programme permet de réaliser une simulation afin de voir évoluer des cas au sein du workflow. L'utilisateur peut alors effectuer l'attribution du travail aux ressources. Ces dernières répondants aux limitations posées par les patterns utilisés.

5.2 Développements futurs

Le présent travail peut être prolongé et il est possible de lui adjoindre un certain nombre de fonctionnalités. Nous allons recenser ici plusieurs d'entre elles.

1. La simulation du fonctionnement dynamique d'un workflow dans le cadre de la gestion des cas peut être remplacée par une implémentation supportant une

mise en situation réelle. Dans cette optique, il est important de découpler les applications client et le système de gestion de workflow. Ces applications sont le moyen de communication du travailleur avec le système.

2. Un système de gestion de workflow entier se doit de respecter le modèle architectural du WFMC (voir chapitre 3). L'indépendance entre l'éditeur de workflow et le gestionnaire des cas ne serait garantie que si le passage de l'un à l'autre se faisait via l'utilisation d'un format d'encodage tel que Usi-XML. Dans sa version actuelle, le logiciel permet de produire un tel code mais le chargement d'une définition de workflow dans le gestionnaire de cas se fait grâce à un format interne.
3. L'aspect des données n'a pas été pris en compte dans le logiciel. Il est possible de poser des conditions sur la manière dont les données du système sont utilisées. [Russ02] identifie différents patterns permettant leur gestion. Concrètement il s'agit par exemple de définir quels cas ont accès à des informations bancaires relatives au découvert des clients.
4. Le gestionnaire des cas fait appel aux patterns de ressource ayant trait à la sélection des travailleurs. Par contre il n'est par exemple pas possible de réaliser une simulation sur le fait de réallouer le travail d'une ressource à une autre ou encore de suspendre la réalisation d'une tâche et de la reprendre par la suite.
5. Les tests effectués dans l'éditeur de workflow quant à la validité du schéma peuvent être améliorés et plus nombreux. Il serait également possible d'ajouter une dimension de mesure sur le temps moyen passé par un cas dans le gestionnaire de cas.

5.3 Mot final

La réalisation de ce mémoire a permis de fusionner plusieurs approches, dotées chacune de ses propres avantages. La manière dont un workflow est spécifié met en relation des éléments tels que les unités organisationnelles, les tâches ou encore les ressources. Ceci fait de l'éditeur une contribution unique. Le choix des différents modèles et leur synergie mènent à une approche cohérente de la notion de workflow. Cette dernière a également l'avantage d'être plus précise que ce que d'autres programmes existants ont implémenté. Nous avons entre autres eu recours à l'arbre de tâche concurrent, rarement utilisé ailleurs.

L'édition graphique permet à l'utilisateur de maîtriser la complexité inhérente à la plupart des projets du domaine du workflow. Quant à la partie dynamique du programme, permettant de gérer les cas, elle a l'avantage de proposer une simulation de l'utilisation d'un workflow défini avec l'éditeur.

Enfin, du point de vue des interfaces utilisateur, le recours au langage UsiXML fait de l'éditeur un produit permettant d'exporter ses spécifications. Ce qui est d'une part un avantage lorsque l'on considère le cadre posé par le modèle architectural de référence de la Workflow Management Coalition, et permet d'autre part la génération automatique des interfaces hommes-machines se rapportant au système.

Bibliographie

[Andr] Andre, S., MDA (Model Driven Architecture) principes et état de l'art, Lyon, 2004.

[Cons] Conspectus, The IT report for directors and decision makers, Workflow, Document & business process management, Angleterre, 2004.

[Guer1] Guerrero Garcia, J., Conceptual Modeling of User Interfaces to Workflow Information Systems, Louvain-La-Neuve, 2006.

[Guer2] Guerrero Garcia, J., Vanderdonckt, J., Modeling User Interfaces to Workflow Information Systems, Louvain-La-Neuve, 2007.

[Mand] Mandviwalla, M., Olfman, L., What do groups need? A proposed set of generic groupware requirements, 1994.

[Mont1] Montero, F., Lopez-Jaquero, V., Vanderdonckt, J., Gonzalez, P., Lozano, M., Limbourg, Q., Solving the Mapping Problem in User Interface Desing by Seamless Integration in IdealXML, Albacete, Louvain-La-Neuve, 2006.

[Mont2] Montero, F., Lopez-Jaquero, V., IdealXML : an Interaction Design Tool, Albacete, 2006.

[Russ1] Russel, N., Ter Hofstede, A., Edmond, D., Van der Aalst, W., Workflow Ressource Patterns, Australie, Eindhoven, 2006.

[Russ2] Russel, N., Ter Hofstede, A., Edmond, D., Van der Aalst, W., Workflow Data Patterns, Australie, Eindhoven, 2006.

[Stav] Stavness, N., Schnieder, K., Supporting Workflow in User Interface Description Languages, Saskatchewan, 2004.

[Tiss] Tissot, M., Environnements d'édition de workflow, France, 2000.

[Usi] UsiXML Consortium, "UsiXML, User Inteface eXtensible Markup Language", Louvain-La-Neuve, 2007.

[Vda1] Van der Aalst, W., Van Hee, K., Workflow Management – Models, methods and systems, Eindhoven, 2000.

[Vda2] Van der Aalst, W., Workflow Management Systems : functions, architecture, and products, Eindhoven.

[Vdd] Vanderdonckt, J., cours de Travail Collaboratif Assisté par Ordinateur, Louvain-La-Neuve, 2006.

[Veyb] Veyble, L., Prieur, P., Le knowledge management dans tous ses états : la gestion des connaissances au service de la performance, France, 2003.

[Wfmc1] Workflow Management Coalition, Terminology & glossary, Hampshire, février 1999.

[Wfmc2] Workflow Management Coalition, The Workflow Reference Model, UK, 1995.

[Wfmc3] Workflow Management Coalition, The Workflow Reference Model – 10 years on, UK.

[Zur] Zur Muehlen, M., Resource Modeling in Workflow Applications, Muenster, 1999.

Manuel utilisateur

Editeur de workflow

- L'éditeur se lance via la classe WorkflowEditor du package portant le même nom.
- Pour ajouter un élément il faut d'abord cliquer sur sa représentation dans le menu à gauche. Ensuite cliquer sur la zone centrale et maintenir le bouton enfoncé aussi longtemps qu'on le souhait pour réaliser un « drag&drop » de l'élément nouvellement créé. Si l'endroit ne peut accueillir l'élément, l'utilisateur en est informé.
- Lorsqu'un élément vient d'être inséré dans le schéma ou que l'option choisie dans le menu à gauche est « select », il est possible d'éditer les attributs et de faire apparaître les fenêtres associées à l'élément graphique dans le menu d'édition sur la droite.
- Pour modifier la valeur d'un champ d'attribut il faut terminer la saisie par enter, autrement l'information ne sera pas enregistrée.
- Pour déplacer un élément, cliquer sur cet élément et maintenir le bouton enfoncé durant le déplacement. Relâcher pour accepter la nouvelle position.
- Lier deux éléments se fait en cliquant sur le premier, puis garder le bouton enfoncé jusqu'à l'élément destination. Une place n'accepte qu'un lien en entrée et un seul en sortie.
- Lorsqu'un bouton de confirmation est présent il faut l'enclencher pour effectuer la confirmation. Si aucun bouton de confirmation n'est présent, l'information est sauvegardée dès que le changement est réalisé (sélection dans un menu déroulant ou modification d'un champs textuel).
- Redimensionner un élément se fait en cliquant sur l'élément et en maintenant le bouton enfoncé jusqu'à l'endroit désiré.
- Sélection et suppression se font en un clic sur l'élément voulu.
- La gestion des jobs et ressources se fait dans le menu nord.
- Pour pouvoir utiliser les ressources il faut créer des boîtes à ressources (jobBoxes) situées dans la même unité organisationnelle que la transition contenant la tâche voulue.
- L'utilisation d'une boîte à ressources passe par le choix du métier concerné dans le menu d'édition situé sur la droite. Ensuite les travailleurs possédant le job dans leurs attributions pourront être ajoutés à la boîte à job.
- Les différents tests se lancent par ce même menu.
- Ainsi que la sauvegarde, le chargement, la création d'un fichier jpg et l'exportation XML.

Gestionnaire de cas

- Le gestionnaire se lance via la classe CaseManager du package workflowManager.
- Le menu nord permet de charger un fichier .wed réalisé avec l'éditeur de worfklow.
- Créer un nouveau cas se fait par la barre de menu.
- Les bureaux des ressources peuvent être montrés ou cachés en cliquant dans ce même menu.
- Une fois un cas créé il se trouve dans la place de départ.
- Cliquer sur une place/transition pour accéder aux opérations réalisables pour les jetons s'y trouvant
- Il est important d'avoir des ressources disponibles dans des boîtes à ressource appartenant à la même unité organisationnelle. Si aucun travailleur n'est présent le travail ne pourra pas être réalisé.