Rapport de Stage

Présenté par

Michaël TISSOT

Environnements d'édition de workflow

Effectué dans le cadre du projet Opéra - INRIA

Date :
24 Août 2000

Responsable :
Laurent Tardif

Remerciements

Je tiens tout d'abord à remercier Cécile Roisin pour m'avoir accueilli dans l'équipe Opéra.

Je remercie mon tuteur Laurent Tardif pour m'avoir guidé en me donnant des méthodes efficaces de travail pour la recherche de documents, l'approche du code de Kaomi et la rédaction de ce rapport, pour sa disponibilité, pour ses nombreuses relectures et ses critiques constructives.

Frédéric Séraphine pour avoir joué le rôle ingrat de taxi vers l'INRIA pendant tout l'été, et pour avoir supporté mes coups de téléphone de bon matin :)

Frédéric Bès et Lionel Villard pour leurs éclaircissements sur certaines parties du code de Kaomi

Et toute l'équipe Opéra pour l'ambiance chaleureuse qui y règne.

1 Introduction

C'est récemment que le monde de l'entreprise s'intéresse au rôle que peut jouer l'informatique dans l'organisation du travail. Le workflow est un outil qui apporte dans cette optique une véritable aide à l'organisation, l'exécution et l'optimisation d'un processus de travail.
Mon stage s'est effectué a l'INRIA Rhône-Alpes, dans l'équipe Opéra qui développe plusieurs axes de recherche :

- L'édition de documents structurés (Thot) [16]
- L'édition coopérative (Byzance et Amaya) [15]
- L'édition de documents multimédia (Madeus) [9]

L'édition de documents multimédia et l'édition de workflow présentent des aspects communs, en particulier l'importance de la dimension temporelle dans ces documents. C'est sur cet axe que s'appuie mon travail ; il s'agit de mettre en évidence les similitudes entre l'édition de workflows et de documents multimédia, puis d'étudier la boîte à outils Kaomi, développée par l'équipe Opera, afin de mettre au point un éditeur de documents workflow.
Mon but lors de ce stage a été dans un premier temps d'étudier la boite à outils Kaomi, les services proposés et la structure de l'application. Après cette étude préliminaire, j'étais en mesure d'écrire une grammaire de représentation d'un workflow tenant compte des besoins inhérents à ce type de document, et cohérent dans l'optique d'une intégration dans Kaomi. Enfin j'ai implémenté un module d'édition de documents workflow au sein de Kaomi, en me penchant sur les aspects gestion temporelle et gestion de ressources pour la planification de tâches.

2 Rôle et structure d'un workflow

2.1 Le Business Process Reengineering

Dans le monde de l'entreprise, l'organisation du travail est un problème complexe. On a d'abord eu a mettre en place des modèles d'organisation pour gérer la production et les besoins matériels (industrie traditionnelle). Puis plus tard l'information a pris toute son importance et on a mis au point des systèmes d'information (base de données, communication, tri des informations etc.). Le but du business process reengineering est de mettre en corrélation ces systèmes avec l'aspect humain dans l'entreprise, a savoir mettre au point des systèmes qui tiennent compte des ressources humaines afin d'obtenir une organisation optimale du travail en termes de rendement et de communication entre les gens.
Le BPR permet ainsi a travers une modélisation et un suivi assisté des processus de travail, d'identifier les problèmes dans l'organisation et d'apporter des solutions d'optimisation non triviales.

2.2 Définition du workflow

Un workflow peut être défini comme un modèle informatique pour représenter un processus de travail. On distingue plusieurs aspects au sein de la notion de workflow :
De manière informelle, un workflow est un travail coopératif impliquant un nombre limité de personnes devant accomplir, en un temps limité, des tâches articulées autour d'une procédure définie et ayant un objectif global.
On peut ensuite parler plus spécifiquement du workflow en tant qu'objet pouvant être décrit par un langage descriptif dans un fichier informatique, qu'une application adaptée (« workflow engine ») peut alors interpréter et exécuter. Ainsi on peut automatiser un processus de travail, par exemple :

- Envoi et réception de fiches électroniques à remplir par les différents intervenants humains, chaque fiche remplie provoquera l'envoi de nouvelles fiches à d'autres personnes, en suivant un processus organisationnel défini.
- Démarrage conditionnel de processus de traitement informatique, comme des requêtes dans les bases de données d'information d'une entreprise (dossiers clients, comptabilité interne…).

L'approche du Business Process Reengineering est la suivante :
On effectue un audit dans l'entreprise :

- Il faut alors interroger toutes les personnes impliquées, pour définir quel est leur rôle dans un processus de travail, quels sont les facteurs intervenant dans l'exécution de leurs tâches, quel est le temps qu'il leur faut pour les accomplir, avec quelles autres personnes sont elles en communication directe pour les besoins du travail. Cela permet de dégager la structure générale du processus de travail.
- On inventorie les ressources (humains, machines) et les tâches effectuées, ainsi que les contraintes qui portent dur ces tâches (temps, facteurs externes).
- On effectue un découpage des tâches impliqués dans le processus global (Hiérarchisation).
- On modélise le workflow à l'aide d'un outil d'édition ; une vue graphique du workflow peut alors mettre en évidence certains dysfonctionnements simples. On peut alors mettre en place le workflow, application qui s'exécutera sur une machine serveur reliée à toutes les cellules de l'entreprise.
- Après exécution d'un certain nombres de cycles du processus de travail, on peut observer un échantillon caractéristique de résultats réels (les temps effectifs, les blocages éventuels).

Une étude poussée effectuée par des personnes compétentes en organisation à partir des résultats permet alors d'identifier les problèmes.
Après un second audit proposant alors diverses orientations, on est en mesure d'apporter des solutions viables aux problèmes d'organisation.
Le workflow est alors un outil mettant en place l'organisation définie. A chaque fois qu'on problème se présente, on peut alors reprendre les deux derniers points.

2.3 Domaines d'application

Les workflows ont de multiples applications dans le monde d'aujourd'hui. L'évolution des processus organisationnels de l'entreprise conduisent à utiliser cet outil. Il répond a un besoin d'optimisation des processus de travail en termes d'utilisation des ressources et de temps effectif.

Le workflow est amené a jouer un rôle important dans les entreprises du monde financier comme les systèmes bancaires, les assurances (délivrer un prêt, opérer un remboursement…). On peut l'étendre a tout processus de travail cyclique dans le monde de l'entreprise.

On s'intéresse aussi a ses applications dans le monde informatique, comme le processus de développement d'un logiciel ; En intégrant l'aspect travail coopératif au sein du workflow, on peut lier l'intégration progressive des éléments d'un logiciel avec l'organisation prévue. Le chef de projet dispose ainsi d'un outil de contrôle sur l'avancement du projet et la cohérence du système en terme de délais.

Les workflows peuvent également être utilisés dans des organisations autres que l'entreprise, comme dans le monde médical : suivi du dossier médical d'un patient (on peut le mettre a jour automatiquement selon les traitements médicaux effectués), planification des opérations chirurgicales (salles d'opérations, chirurgiens)…
On peut imaginer des applications des workflows dans l'éducation par exemple la mise en place de processus de contrôle continu de l'apprentissage via le web.

2.4 Modélisation d'un workflow

Je présente ici un modèle de représentation interne d'un workflow. Ce modèle doit tenir compte des dimensions logique et temporelle de la problématique du workflow.
On peut spécifier un workflow par un ensemble d'objets (les **tâches**) et de règles (les relations entre les tâches).
Un workflow est soit une tâche atomique, on parle alors de tâche élémentaire, soit un sous-workflow (imbrication), on parle alors de tâche composite (ou composant).

2.4.1 Définition d'une tâche élémentaire.

Une tâche élémentaire est décrite par un nom, les ressources impliquées (intervenants humains ou informatiques), une durée si celle-ci est prévisible, et éventuellement des contraintes temporelles.

Les contraintes temporelles propres à une tâche élémentaire portent sur sa date de début ou de fin, et on peut définir pour chaque date des contraintes de type 'au plus tôt le' et 'au plus tard le'. C'est ce que nous définirons comme les relations unaires sur la tâche.
Chaque tâche disposera donc d'un couple (Istart,Ifinish) où I start et I finish sont les intervalles de temps acceptés pour les instants start et finish. Si la tâche n'a pas de contrainte, sa relation unaire est donc ([-∞,+∞],[-∞,+∞]). Le positionnement de la tâ-

che dans le temps sera alors fonction des relations avec les autres tâches sur lesquelles nous nous étendrons plus loin.
Ainsi si l'on veut représenter une tâche T qui doit débuter entre les instants 2 et 5 et finir avant l'instant 9, on obtient :

([2,5],[- ∞,9])
Les dates de début et de fin de tâches peuvent aussi être fixées ; on représente alors l'instant t par [t,t]

La tâche doit commencer après le 1er janvier et finir le 31 décembre 2000: ([01/01/2000, +∞],[31/12/2000, 31/12/2000])

2.4.2 Définition de tâches composites

2.4.2.1 Composition des tâches

Un workflow est représenté comme un ensemble de tâches (composites ou simples) liées par des relations de composition. Cette représentation induit un arbre dont les feuilles, les tâches atomiques, sont liées par des relations de composition, les nœuds.
On note la relation de composition binaire comp(tâche1,tâche2) ; chaque tâche peut être une tâche élémentaire ou composite.

Ainsi avec deux tâches élémentaires T1 et T2, il est possible de définir le workflow X suivant :

X = comp (T1,T2)
X est constitué des deux tâches T1 et T2

En ajoutant une tâche élémentaire T3, il est possible de définir W :

W = comp (X,T3)
W est constitué du composant X et de la tâche élémentaire T3.

On déduit de cette composition l'arbre décrit dans la Figure 1.

W = comp(comp(T1,T2),T3)

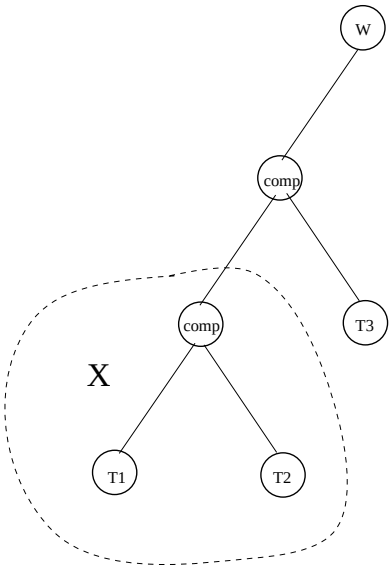


Figure 1 : Structure hiérarchique d'un workflow

Cet arbre ne permet pas de représenter le workflow mais seulement son squelette général. L'orientation dans le temps des tâches de chaque composant est explicitée par l'ajout de relations temporelles entres ces tâches.

2.4.2.2 Mise en relation des tâches

Les relations de composition permettent de définir le «squelette» du workflow. Il faut alors modéliser l'agencement temporel des tâches.
Pour chaque composant séquence, ses sous-composants et ses tâches élémentaires peuvent être contraints par des relations temporelles.

Il faut disposer d'un ensemble de relations pour modéliser les différents liens temporels entre les tâches.
On utilise les relations sur les intervalles de Allen [1] dont la signification est donnée dans la Figure 9. (on peut assimiler une tâche à un intervalle [début, fin])

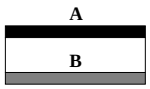
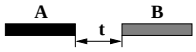
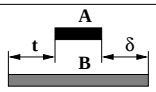
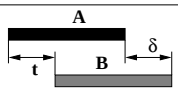
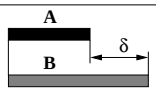
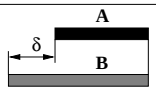
Relations	Sémantique Graphique	Contraintes Numériques
A equals B		dur(A) = dur(B)
A before B		t > 0
A during B		t < dur(B) - dur(A) & t > 0 & δ > 0
A overlaps B		t > dur(A) - dur(B) & t > 0 & δ > 0
A meets B		____
A starts B		dur(A) < dur(B)
A finishes B		dur(A) < dur(B)

Figure 2 : Les opérateurs d'Allen

Avec ces 7 relations, on peut modéliser les liens temporels entre objets.
Chaque relation entre tâches se traduit en relation d'instants sur les 'start' et 'finish' de chaque tâche.
Ainsi A before B se traduit en A.finish < B.start
A overlaps B se traduit en A.start < B.start, A.finish < B.finish

Pour représenter un composant séquence on donnera donc la liste de ses tâches (élémentaires et composites) ainsi que les relations qui lient ces tâches.
Enrichi des relations temporelles, le modèle induit alors un graphe orienté dans le temps.

On a donc défini un formalisme de description pour les workflows avec contraintes temporelles. (Voir [5][12])
Nous allons à présent nous intéresser à l'intégration d'un éditeur de workflow à la boîte à outils Kaomi développée par l'équipe Opera. Dans cette perspective nous présenterons l'aspect document multimédia et la structure de la boîte à outils.

3 Présentation de Kaomi

L'équipe Opéra travaille depuis plusieurs années sur les documents multimédia. Pour cela l'équipe a développé une boîte à outils appelée Kaomi [10], dont l'architecture ouverte a permis de mettre au point un éditeur de documents multimédia s'appuyant sur Madeus, le modèle de description propriétaire, mais aussi sur les documents SMIL (standard établi par le W3C en matière de documents multimédia) ou encore MHML. En présentant l'environnement d'édition, la structure du programme et le modèle de description des objets, nous mettrons en évidence les analogies entre l'édition de documents multimédia et l'édition de workflow.

3.1 Les documents multimédias

On dispose a l'heure actuelle d'outils d'édition puissants pour les documents structurés n'intégrant pas la dimension temporelle, comme les images ou le texte : Thot s'inscrit par exemple dans ce cadre. Les documents actuels possèdent une dimension spatiale, à savoir le placement des objets dans le document. L'aspect logique est également très présent, car la démarche d'édition nécessite une hiérarchisation des objets. Enfin la notion de dimension hypertexte pour un document est de plus en plus présente, notamment depuis l'explosion du web : en effet certaines parties du texte, les liens hypertexte, permettent d'accéder à d'autres documents [3].
La caractéristique principale du document multimédia est l'ajout de la dimension temporelle.
Ceci permet d'intégrer au document classique fait de texte et d'images des présentations animées, des sons ou encore des processus d'interaction avec l'utilisateur.
Avec le développement des technologies de transfert de données à haut débit, l'évolution du document html « classique » vers le document multimédia est de plus en plus d'actualité [8], des standards sont déjà définis comme SMIL. Les applications possibles de tels documents sont diverses ; la présentation d'un produit serait par exemple enrichie d'une démonstration animée et interactive de celui-ci. Les méthodes d'enseignement a distance trouveraient également dans les documents multimédia un outil d'interactivité accrue avec l'élève.

L'une des caractéristiques importantes des objets est la granularité : on peut décomposer une vidéo en scènes qui sont elles-mêmes des séquences d'images. Un élément audio peut quant à lui être découpé en échantillons de son. Cela induit la hiérarchisation du document et de ses éléments, ce qui est conceptuellement très proche de l'imbrication de workflows.

On retrouve également deux notions essentielles du workflow dans la problématique du document multimédia : la contrôlabilité et la flexibilité.
En effet si les unités de présentation sont bornées dans le temps, elles seront représentées par des intervalles contrôlables ; les bornes définiront alors la flexibilité du document. Une animation peut par exemple être jouée avec un « framerate » oscillant quelque peu autour d'une valeur idéale sans que cela ne choque l'utilisateur.
Si les unités de présentation ne sont pas bornées dans le temps, on parle d'incontrôlabilité car la durée de présentation dépend de facteurs externes, et on définit alors des bornes admissibles pour la durée de présentation (montant d'indéterminisme). Jouer une vidéo en transfert continu depuis un serveur (format

RealPlayer par exemple) est une présentation dont on ne peut déterminer la durée a priori.

La phase d'édition d'un document multimédia va consister entre autres à décrire ce qu'on appelle un scénario temporel : ordonnancement temporel des unités entre elles.
Lors de la phase de présentation les objets seront alors placés dans le temps, et le programme orchestre alors l'exécution des différentes unités.
C'est sur ces bases que s'appuie le logiciel Kaomi et le modèle de documents multimédia Madeus, développés par l'équipe Opéra.

Il est important de noter que le modèle de description Madeus ne permet pas de représenter des contraintes de dates absolues sur les instants (relations unaires), car dans un document multimédia ce besoin n'apparaît pas, tout s'organise autour de dates relatives au lancement de la présentation du document. Dans un workflow la datation absolue est un besoin important (jours fériés, délais imposés par les partenaires de l'entreprise).

3.2 Présentation de la boite à outils

Kaomi est la boîte à outils développée par l'équipe Opéra qui sert de supports aux logiciels Madeus Editor, le modèle de description de documents multimédia établi par l'équipe, et SMIL Editor [11], le modèle standard établi par le W3C.
Cette boîte à outils fournit un ensemble de services pour le concepteur d'environnements auteur de documents multimédia. Ces services sont de plusieurs niveaux :

- Les services qui seront utiles pour la construction de notre application :
  - Un ensemble de vues pour la visualisation du document et la navigation à l'intérieur des différentes dimensions du document.
  - Un ensemble de structures de données pour manipuler des documents multimédia
- Les services utiles pour la gestion temporelle et la gestion de ressources
  - Un ensemble de services d'aide à l'auteur tels que la vérification de cohérence et le formatage du document.

Kaomi est une couche logicielle portable qui dispose d'un parser XML, c'est a dire un interpréteur de code XML ([2]) vérifiant la grammaire spécifiée par une DTD (Document Type Definition). Il est donc envisageable de l'adapter a l'édition de workflows, il faudra alors commencer par spécifier une grammaire de représentation d'un workflow par une DTD en XML.
Nous allons présenter plus en détails les différents services offerts par Kaomi au programmeur.

3.2.1 Visualisation

Dans le but de faciliter l'auteur dans sa description de document, Kaomi dispose de plusieurs modules de visualisation :

Visualisation spatiale :
La fenêtre principale présente le document en cours de composition, et la disposition spatiale des éléments, à la manière d'un éditeur graphique HTML. Afin de gérer facilement les relations entre objets, l'utilisateur dispose d'une palette d'opérateurs spatiaux et d'une palette d'opérateurs temporels. Le détail de ces relations est donné en 2.3.
La Figure 3 présente la vue principale (dite vue présentation) d'un document multimédia.

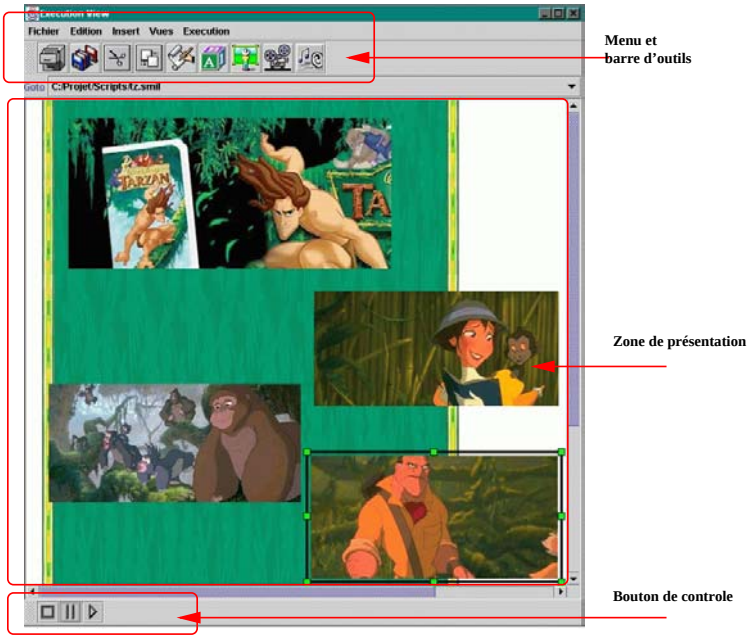


Figure 3 : Vue présentation dans Kaomi

Visualisation hiérarchique :
Cette vue présente l'arbre hiérarchique du document structuré.

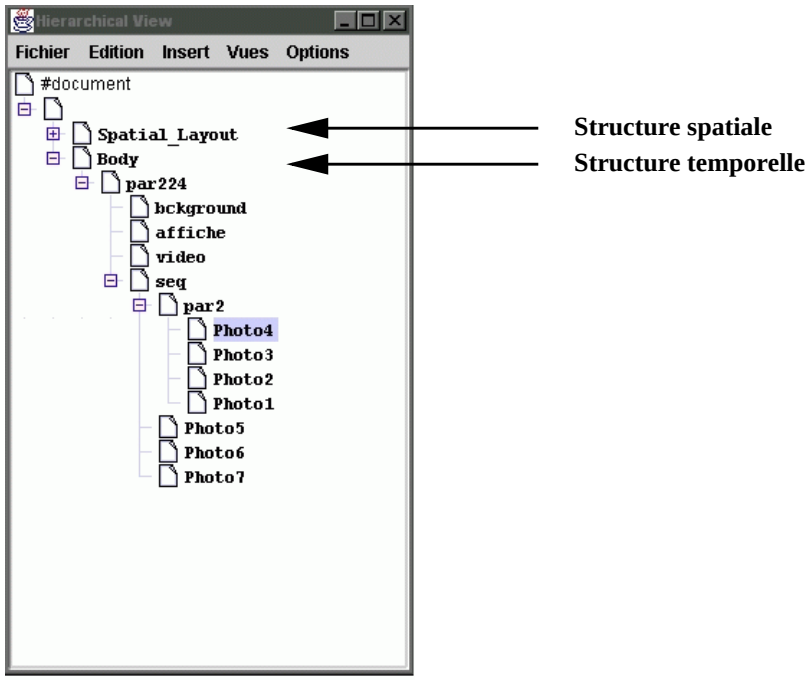


Figure 4 : Vue hiérarchique dans Kaomi

Visualisation du scénario temporel :
Cette vue est de type ligne temporelle (en anglais timeline) ; les éléments de présentation sont représentés par des rectangles dont la longueur est proportionnelle à leur durée. Les relations temporelles entre objets sont représentés par des traits horizontaux pour les délais fixes, et des ressorts pour les délais flexibles. Enfin les traits verticaux représentent les contraintes de simultanéité de deux instants. On peut voir un exemple de vue timeline dans la Figure 5.Cela permet à l'utilisateur d'avoir une vue globale « parlante » du scénario temporel. L'apport important de cette visualisation est le fait qu'elle soit dynamique : On peut déplacer les rectangles représentant les éléments dans le scénario temporel, dans les limites imposées par les contraintes.

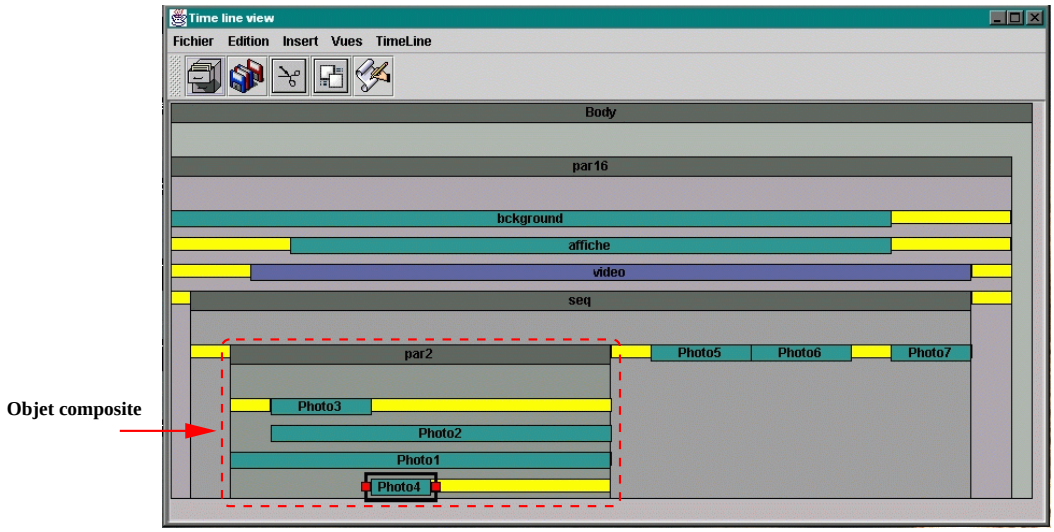


Figure 5 : La vue Timeline dans Kaomi

Fenêtre source :
Cette fenêtre est un éditeur du texte représentant le scénario. L'utilisateur familier avec le langage propriétaire de spécification (à base de contraintes) peut ainsi contrôler ou modifier la composition de son document dans sa structure brute. Ajoutons que le langage utilisé est uniquement déclaratif et ne nécessite donc pas de programmation.

3.2.2 Structure de Kaomi

3.2.2.1 Architecture générale et extensibilité de la boîte à outils

Le système se compose de trois parties principales : le système de gestion de fichiers, le système d'édition (système auteur) et le système de présentation (les vues).Le système de fichiers gère le chargement et la sauvegarde de documents sources, à savoir les transitions entre le document source en XML et la structure de données interne. Le système d'édition gère la manipulation du document : opérations d'ajout, de modification… Le système de présentation s'occupe d'ordonner les différents éléments dans le temps et de les présenter à l'utilisateur.

Kaomi est conçu pour être étendu simplement (Figure 6), et cela a plusieurs niveaux selon les services à utiliser :

- Les Vues : on peut intégrer dynamiquement des nouvelles vues grâce au gestionnaire de vues. De plus on peut étendre les fonctionnalités d'une vue existante.
- Structure de document : ajouter des attributs aux éléments est assez facile. On peut également modifier la structure de données ou ajouter des types d'éléments.
- Services d'édition : L'extension de ces services est plus complexe. Toutefois on peut ajouter des fonctionnalités d'édition.
- Services de fichiers : On peut s'appuyer sur des modèles de description différents, grâce au langage XML qui permet de décrire tout type de document. Il faut alors modifier le parser XML de Kaomi pour ce qui concerne la construction de la structure de données interne.

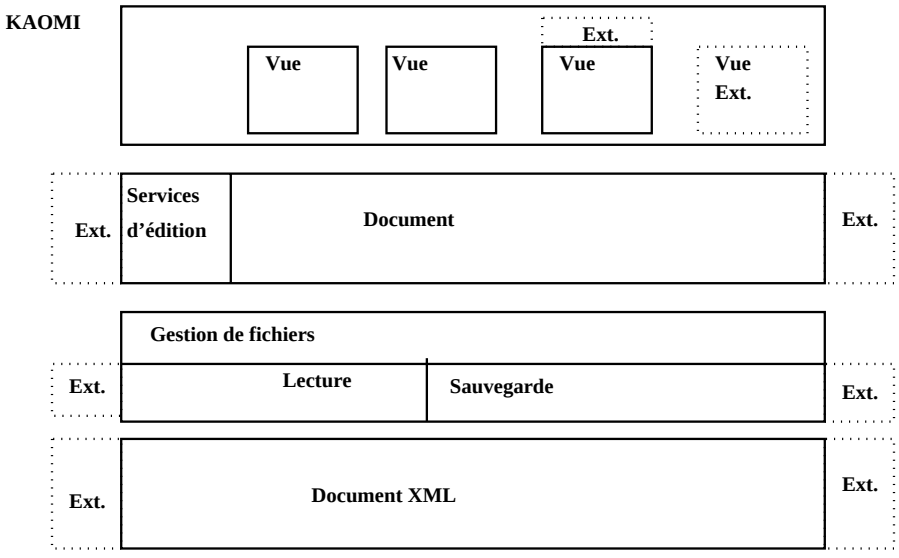


Figure 6 : Extensibilité de Kaomi

3.2.2.2 Représentation d'un document multimédia dans Kaomi

La structure de données de représentation d'un document dans Kaomi intègre les 4 dimensions du multimédia :

- dimension logique : les éléments sont décomposés en éléments basiques et composites, et tous les éléments sont typés. Pour les besoins du multimédia on a défini des types texte, image, audio, vidéo ou object, le type object permettant d'intégrer d'autres types de médias sous forme de plugins.
- dimension spatiale : chaque élément dispose d'attributs top, left pour définir sa position et width, height pour définir sa taille. Dans Madeus on peut utiliser des relations spatiales entre objets (A à gauche de B, A et B centrés verticalement…)
- dimension hypertexte : chaque élément peut être défini comme un lien, il pointe alors vers l'élément multimédia voulu.
- dimension temporelle : Kaomi utilise les opérateurs sur les intervalles d'Allen (voir 1ere partie) ; ceux ci sont étendus pour les besoins spécifiques aux documents multimédia : on introduit un paramètre pour les relations qui impliquent un délai, afin de pouvoir fixer ce délai, par exemple on peut dire « A est 10 secondes avant B ». De plus on ajoute aux relations d'Allen des relations temporelles causales parmin, parmax et parmaster qui permettent d'exprimer des contraintes du type « A provoque la terminaison de B ». on peut ainsi introduire la notion d'évenements dans le document multimédia, par exemple arrêter une vidéo lorsque l'utilisateur appuie sur un bouton.

  Les workflows quant à eux possèdent une dimension logique (découpage du workflow en sous-tâches) et une dimension temporelle (fondamentale).

3.2.2.3 Attributs des éléments

Pour modéliser les éléments basiques flexibles, chaque élément a un attribut durée à 2 valeurs : durée minimale, idéale, maximale.
Un élément dispose en outre d'attributs spatiaux : top et left pour la position, width et height pour la taille.

Les objets composites sont construits par une relation de composition ; Pour chaque composite on explicite alors les relations temporelles et spatiales entre ses fils.

3.2.3 La gestion temporelle

le gestionnaire temporel de Kaomi intervient à différentes phases :

- gestion des relations d'intervalles (Représentation interne des spécifications de l'auteur)
- traduction des relations (On passe de relations a base d'intervalles à des relations a base d'instants)
- maintien de la cohérence (vérification de la cohérence de chaque contrainte temporelle)
- formatage temporel (on attribue pour chaque élément son début et sa durée – une partie est faite dynamiquement)

Le formatage temporel statique se fait à partir de la construction d'un graphe de contraintes sur les instants des éléments, et d'algorithmes de parcours dans cet arbre [6].

3.3 Conclusion sur les possibilités d'intégration du modèle workflow dans Kaomi.

La sémantique des objets traités pour le workflow et le multimédia est différente (tâches et éléments de présentation), par conséquent à l'exécution les processus impliqués sont de nature différente (envoi de mails pour le workflow, lecture d'une vidéo pour le multimédia…). Cependant on peut assimiler un élément multimédia à une tâche (instants de début et de fin , durée), mis a part le problème de la contrainte de date sur les instants nécessaire pour les workflows. De plus, le modèle de représentation hiérarchique est parfaitement adapté au workflow, et le système d'intervalles temporels donnant le domaine de validité de l'élément est tout aussi valable pour les documents multimédias que pour les workflows. Il paraît donc intéressant d'étudier les avancées caractéristiques effectuées pour le formatage temporel de documents multimédia dans Kaomi, afin de pouvoir disposer d'un module de gestion temporelle efficace pour les environnements d'édition de workflow.
L'aspect gestion de ressources est par contre un problème qui n'est pas central pour l'édition de documents multimédia. Pour l'entreprise c'est un aspect déterminant dans l'optimisation de l'organisation, car il faut tenir compte des contraintes horaires des salariés, des personnes suremployées ou sous-employées, des congés.

4 Extension de Kaomi pour l'édition de workflow

Je présente dans cette partie comment j'ai utilisé la boîte à outils Kaomi pour l'édition de workflow, à savoir le choix d'un modèle de document décrit par une DTD, la structure de données interne adoptée, le remplissage de cette structure depuis le document source (parsing). Puis je présenterais les systèmes de contraintes et la façon dont on les utilise pour la gestion temporelle et la gestion de ressources.

4.1 Spécification d'un modèle de document workflow par une DTD

On définit une grammaire de description d'un workflow sous forme de DTD XML. En effet la boîte à outils Kaomi s'appuie sur des documents XML (SMIL et Madeus). On pourra ainsi étendre Kaomi aux documents workflow. Il existe un parser XML pour les documents Madeus et un pour les documents SMIL, il faudra donc implémenter un parser pour les documents de type workflow. Notre DTD doit permettre d'exprimer tous les aspects du workflow décrits dans le chapitre précédent ,tout en tenant compte des contraintes inhérentes à la boîte à outils comme le format de spécification des durées.
La notion supplémentaire importante dans les workflow est la gestion de ressources. L'auteur doit pouvoir a tout moment ajouter ou supprimer des ressources. Une liste des ressources disponibles et de leurs attributs (nom, temps d'utilisation consécutive,

cout a l'heure) sera donc déclarée dans un bloc en tête du document source. De plus on ajoutera un attribut « ressources » a chaque tâche basique, représentant les ressources utilisées (zéro, une ou plusieurs) et leur taux d'utilisation (de 1 a 100%).
Dans chaque tâche composite on pourra définir un ensemble de relations sur ses fils dans un bloc « relations ».

Grammaire simplifiée d'un document workflow :

Workflow = resourcelist (composite (relations)*)*
Resourcelist = (resource)*
Composite = (basique)* | (composite (relations)*)*

Relations = nom1 meets nom2 | nom1 before nom2 | etc… (les relations d'allen)

On donne la DTD complète d'un document workflow en annexe.

4.2 Représentation interne d'un workflow

4.2.1 Intégration dans Kaomi

Kaomi est une boîte à outils relativement ouverte, permettant l'extension à d'autres problématiques que le multimédia. Il était donc possible de modifier le parser sans complètement modifier la structure de Kaomi.

L'application est constituée de plusieurs bases (ou modules) : Kaomi la boîte a outils proprement dite, la base SMIL qui contient les classes utilisées lorsqu'on édite un document SMIL, et la base Madeus.
Mon travail introduisant un nouveau type de document, j'ai donc crée une base Workflow. Des que l'on édite un fichier de type *.workflow, l'application utilise les classes de ce module.

Les classes implémentées pour les besoins du workflow sont les suivantes :

- WorkflowSAXHandler : le parser XML pour les workflow (analyse grammaticale)
- WorkflowDocument : la structure de données de représentation interne d'un document workflow.
- WorkflowObject : représente une tâche basique

- ResourceUsedObjectList : le format d'attribut « ressources » pour les tâches basiques.

- WorkflowObjectAttributes.properties : configuration de la palette d'attributs pour éditer les objets du document workflow

- KaomiResourceChoice.java : l'interface de sélection de ressources dans la palette d'attributs.

Comment utiliser les services de Kaomi ?

On pourra utiliser les vues hiérarchiques et timeline.
Dans la vue timeline, La manipulation explicite des objets dans la limite des contraintes est très intéressante pour le workflow , car elle constitue une réelle aide à l'auteur pour la planification.
On utilise la vue de présentation pour observer le déroulement de l'exécution du workflow ; cela constitue une sorte de simulation d'exécution.
A terme on voudrait pouvoir lancer des processus , il faudrait implémenter un gestionnaire d'évenements (lancement de processus, evenements de fin notifiés a l'application).

4.2.2 Structure de données adoptée

Le parser XML de Kaomi construit la structure de données interne d'un document multimédia à partir de la description XML du document.
La structure de données interne de représentation d'un document multimédia (SMIL ou Madeus) est la suivante :

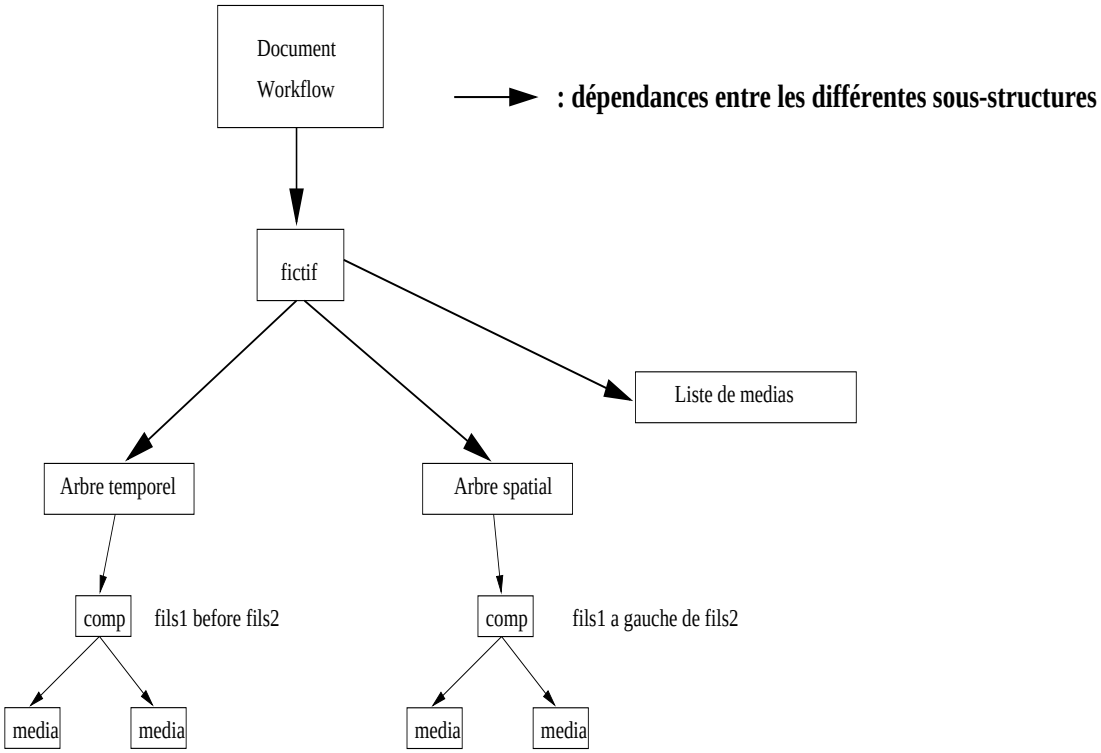


Figure 7 : Structure de données de représentation d'un document multimédia

Pour le document workflow on ajoute un pointeur vers une liste des ressources, et la liste des médias est modifiée en liste de tâches (on utilisait une classe MediaElement, on utilisera une classe WorkflowObject apte à décrire des tâches basiques)
On obtient donc la structure de données suivante :

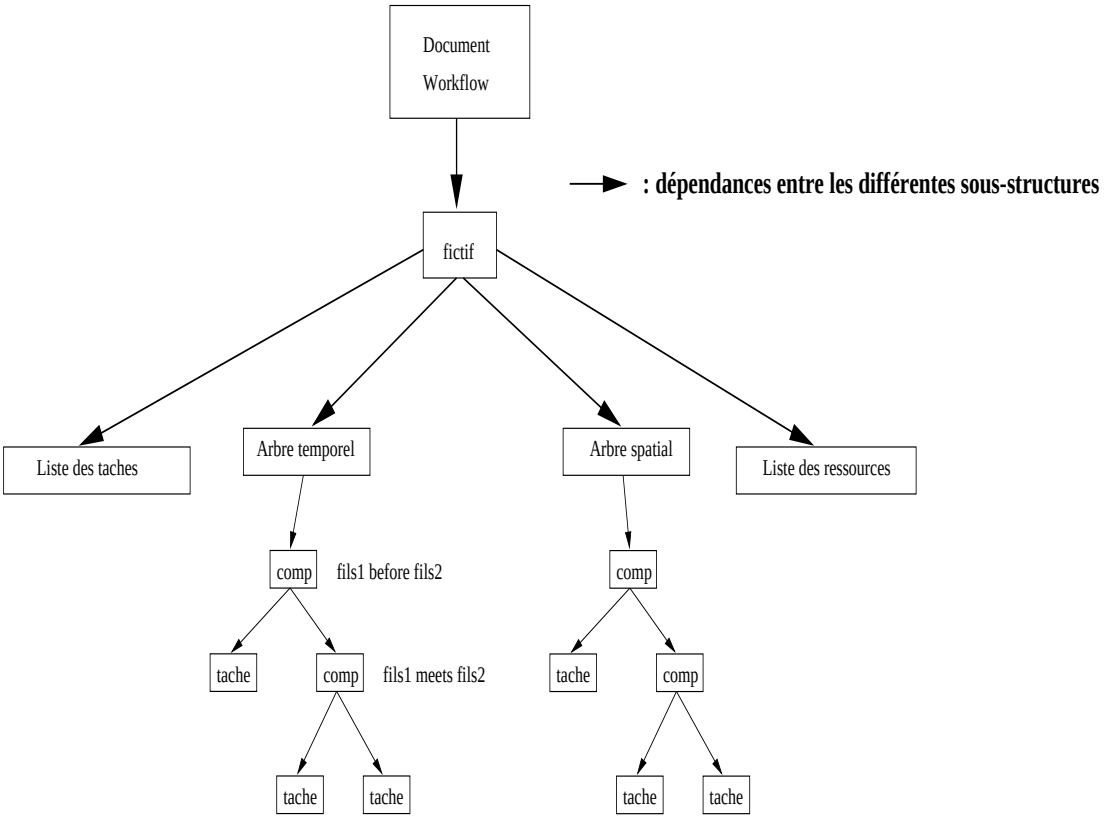


Figure 8 : Structure de données adoptée pour représenter un document workflow

4.2.3 Attributs des objets manipulés

Les ressources disposent des attributs suivants :
- Nom (identificateur de la ressource)
- Durée de travail consécutif possible
- Coût horaire

A terme on veut : contraintes absolues (par exemple congés). Contraintes générales : « ne travaille pas le dimanche ».

Les tâches basiques disposent des attributs suivants :
- ID : identificateur unique de la tâche ; si l'auteur ne le spécifie pas on le génere automatiquement
- Nom : le nom de la tache ; interet juste sémantique
- Durée : nécessaire pour le formatage temporel => format de la durée identique a madeus (min, max , pref)

- Ressources utilisées : aucune , une , ou plusieurs, avec des taux d'utilisation de 1 à 100%
  Ecrit sous forme de chaine de caractere : « nom :taux nom2:taux2 … » cette chaine est interprétée par le parser.

4.2.4 Structure temporelle

On construit l'arbre hiérarchique des tâches. Chaque tâche est de type TemporalElement à savoir qu'elle contient l'identificateur unique de la tache permettant de le lier avec la liste des tâches. Le TemporalElement contient également les informations de durée : durée minimale , maximale, et préférée. Ainsi le format de spécification des durées est identique a celui utilisé pour Madeus, et le gestionnaire temporel ne nécessitera pas de modifications particulières. Pour chaque relation entre les fils d'un composite, on appellera une méthode d'ajout de relation sur l'objet composite. On s'en tient là aussi aux primitives utilisées pour les documents Madeus. Ainsi Kaomi traduit les relations d'intervalles en relations d'instants, qu'il soumet au résolveur de contraintes.

4.2.5 Structure spatiale

On définit des fonctions pour générer les caractéristiques spatiales automatiquement ; l'auteur ne spécifie rien, la vue de présentation est juste a titre d'observation.
Dans un premier temps on représente les tâches comme des médias textes qui apparaissent et disparaissent lors de la présentation du document ; ceux ci sont placés sur des lignes différentes.

4.2.6 Mise au point du parser XML pour les documents workflow

Il s'agit essentiellement de modifier le parser écrit pour les documents SMIL ou Madeus (au choix) ; on dispose ainsi de primitives de construction de la structure de données, aisément modifiables pour notre structure de données (proche de celle utilisée pour les documents multimédia). Concrètement chaque balise rencontré dans le document XML provoque l'appel de méthodes prenant en paramètre les attributs pour ce tag. Ainsi lorsqu'on rencontre une balise <basic> on va définir une nouvelle tâche : insertion dans la liste des tâches, insertion dans l'arbre spatial avec des attributs générés, et insertion dans l'arbre temporel avec les attributs temporels ; on pourra éventuellement ajouter des relations dans l'arbre temporel (lorsque le parser rencontrera un tag <relations>).

5 Gestion de ressources

5.1 Edition des ressources

La palette d'attributs est un outil de Kaomi permettant d'éditer facilement les attributs d'un élément. En cliquant sur un élément, on a accès à ses attributs classés par type : identification, informations spatiales, informations temporelles… On peut éditer simplement ces attributs via des boites d'édition.
Compte tenu des spécificités d'un document workflow, il a fallu adapter la palette d'attributs.
La partie gestion de ressources a notamment fait l'objet d'une présentation spécifique, conviviale pour l'auteur (Figure 9).
En effet le format de description textuelle des ressources utilisées par un élément est peu pratique. On a donc fait apparaître dans une liste l'ensemble des ressources disponibles, et on peut aisément ajouter, modifier, ou enlever des ressources pour une tache.

Figure 9 : Editions des attributs ressources

5.2 Utilisations des résolveurs de contraintes pour la gestion de ressources

Dans l'optique d'une aide à l'auteur dans la planification de tâches, l'application d'édition doit permettre d'assurer la cohérence temporelle du workflow mais également la cohérence en terme d'utilisation des ressources.

On peut vouloir spécifier des contraintes sur les ressources de différent type :

- Contraintes d'utilisation continue : un homme ne peut travailler pendant 25h
- Contraintes d'utilisation sur plusieurs taches : on ne peut dépasser un usage à 100%
- Contraintes de dates absolues : un homme ne travaille pas le dimanche

- Contraintes de cout : on veut pouvoir spécifier un coût maximum du travail pour une tache.

5.2.1 Présentation des systèmes de contraintes

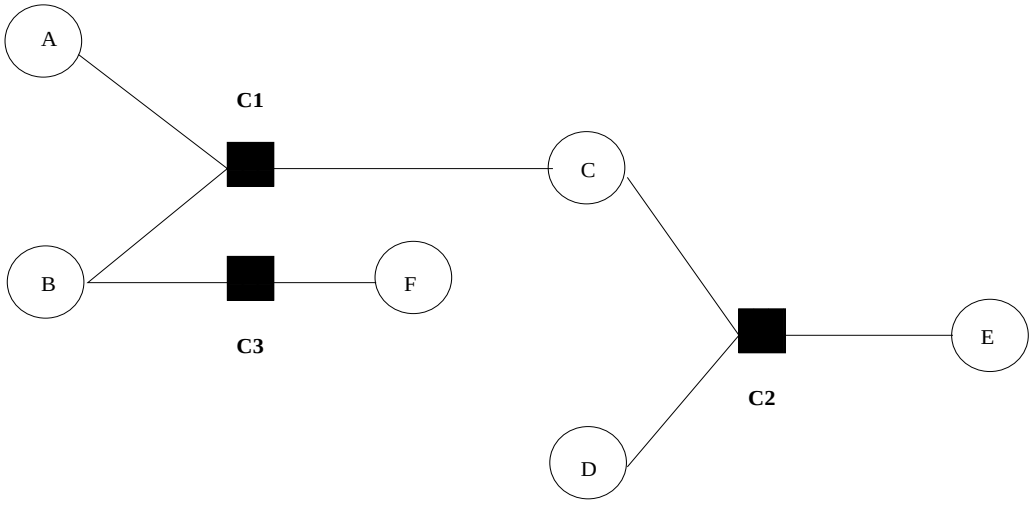
Un système de contraintes consiste en un ensemble de variables et en un ensemble de contraintes qui limite les combinaisons de valeurs pour ces variables.

Plus formellement, on définit un système de contraintes par un triplet {V,D,C} où :

- V = { V1,V2,…,Vn} est un ensemble fini de variables,
- D={D1,D2,…,Dn} représente l'ensemble des domaines de valeurs,
  $\forall$ i $\in$ [1,n], Di est le domaine des valeurs possibles pour la variable Vi,
- C= {C1,C2,…,Cn} est un ensemble de contraintes entre les variables, c'est à dire un sous ensemble de D1x…xDn. Chaque contrainte peut être définie en intention par une équation ou en extension par l'ensemble des t-uples qui la satisfont.

Une solution pour un tel système est une instanciation des variables de V qui satisfait toutes les contraintes de C.

Un système de contraintes peut être représenté par un graphe non orienté appelé graphe de contraintes.



C1 : A+B=C
C2 : C+D=E
C3 : F=B

Figure 10 : Système de contraintes (variables A,B,C,D,E,F ; contraintes C1,C2,C3)

Dans ce graphique, les cercles représentent les variables et les carrés les contraintes. Les arêtes symbolisent les liens variables-contraintes c'est à dire la présence d'une variable dans une contrainte.

Il existe deux problèmes de résolution différents pour un système de contraintes :

- **La satisfaction de contraintes** : il s'agit d'assurer la cohérence d'un système de contraintes. La résolution d'un tel problème consiste à trouver une ou toutes les combinaisons de variables qui permettent de satisfaire l'ensemble des contraintes.

- **Le maintien de solution** : Dans un problème de maintien de solution, l'objectif est de rétablir la cohérence d'un système de contraintes à partir d'une solution courante et d'une perturbation de cette solution. Cette perturbation peut être provoquée soit par la modification de la valeur d'une variable, soit par l'ajout ou le retrait d'une contrainte.
  Dans un système auteur, on rencontre souvent ce problème lors d'un changement de variable qui doit être répercuté à l'écran, il faut que ce que voit l'auteur à l'écran soit pertinent par rapport à l'espace des solutions.

Le programmeur spécifie le système de contraintes par une approche déclarative qui consiste à ajouter des contraintes (ex : A = B +C, B = 2*A – C etc)
Celui-ci ne s'intéresse pas à la manière dont le système de contraintes va produire un ensemble de solutions : c'est le résolveur de contraintes qui effectue cette tache. Il assure le maintien de l'ensemble des contraintes et donne une solution au programmeur.

Pour les documents multimédias, Kaomi utilise différents résolveurs de contraintes pour

- le formatage temporel du document
- l'édition dans la vue temporelle
- la vérification de cohérence.

5.2.2 Algorithmes de résolution de contraintes

Kaomi fournissant les services nécessaires à la gestion temporelle des tâches, on peut obtenir un formatage temporel cohérent selon les spécifications de l'auteur : durées des tâches et relations temporelles entre tâches.
En définissant un système de contraintes pour les ressources, on pourra utiliser des résolveurs de contraintes pour que la planification des tâches tienne compte des contraintes liées à l'utilisation de ressources.
Pour ce faire, on étudie les différents algorithmes de résolution de contraintes car ils constituent une solution possible a notre problème. On s'intéressera à travers l'étude des différents résolveurs à une utilisation optimale des résolveurs dans le cadre de la gestion de ressources.

Algorithmes basés sur la propagation locale :

On distingue deux phases :
la phase de planification qui consiste à orienter le graphe de contraintes de manière à obtenir un graphe de solutions
la phase d'exécution au cours de laquelle on applique le plan (les méthodes qui sont définies)

La phase de planification présente l'avantage de n'être exécutée que lorsque l'on perturbe le système c'est à dire lorsqu'on introduit une nouvelle variable ou une nouvelle contrainte, et la phase d'exécution présente celui d'être très peu coûteuse et donc très rapide. Le problème majeur de cette méthode est que elle n'aboutit pas forcément à une solution lorsqu'il y a des cycles dans le graphe.

Dans l'exemple précédent, si on introduit une contrainte d'égalité C4 : E = D… , le graphe de contraintes présente alors un cycle.

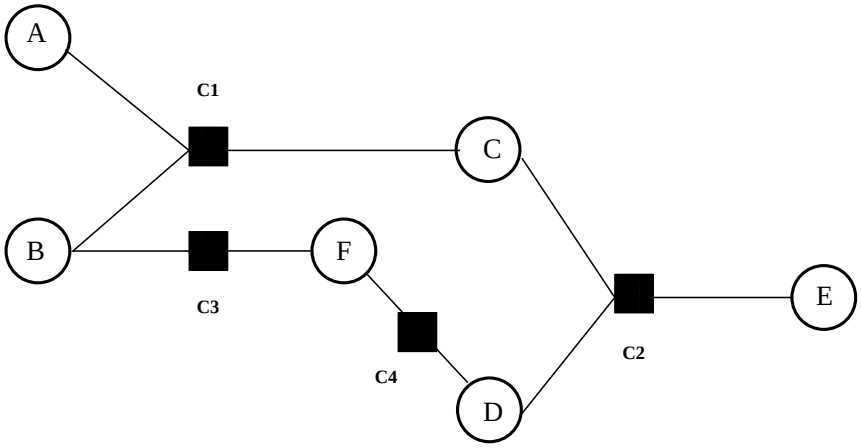


Figure 11 : L'ajout d'une contrainte fait apparaître un cycle dans le graphe de contraintes.

Lors de la phase d'orientation du graphe de contraintes, le résolveur a propagation locale aboutit à une incohérence qu'il ne sait pas traiter. Il ne peut donc résoudre le système de contraintes.

Algorithmes basés sur le retour arrière (backtrack) :

Le résolveur tient compte de toutes les contraintes.
L'algorithme démarre en fixant un ordre sur les variables et sur les valeurs des domaines de chaque variable. Ensuite, il affecte des valeurs aux variables, l'une après l'autre, et teste chaque fois si l'instanciation partielle obtenue est localement cohérente. En cas d'incohérence (contrainte non satisfaite), l'algorithme effectue un retour arrière sur la dernière affectation. Elle est remplacée par une nouvelle affectation sur la même variable avec la valeur suivante. S'il n'y a plus de valeur dans le domaine, il faut effectuer un retour-arriere sur la variable précédente. Ainsi, ce genre d'algorithme parcourt de façon exhaustive l'espace des solutions jusqu'à en obtenir une cohérente.
L'inconvénient qui se présente est la complexité de l'algorithme : en effet celui ci se révèle inutilisable si l'on souhaite un temps de réponse rapide pour des problèmes de taille importante.

Algorithmes basés sur le simplex

L'objectif d'un algorithme à base de simplexe est de maximiser une fonction d'optimisation (f) tout en satisfaisant l'ensemble des contraintes.
L'algorithme se décompose en deux phases :

- la phase de prétraitement : Soient deux tableaux Cc, et Cn. Cc contiendra les variables et les (équations) contraintes, et Cn celles qui sont non contraintes. Cette première étape a été améliorée pour rendre la deuxième phase plus efficace. Initialement toutes les équations sont placées dans le tableau Cc. Un filtrage base sur la méthode d'élimination de Gauss-Jordan a été rajouté. Cette phase a pour but de supprimer toutes les variables non contraintes du tableau Cc pour les placer dans Cn. Pour réaliser cela, on cherche dans Cc une équation contenant une variable v non contrainte. On retire cette équation de Cc. On résoud l'équation pour v, en générant une nouvelle équation e de la forme : e1 = v (ou e1 est une expression). On ajoute e1 a Cn. On substitue alors toutes les occurrences de v dans Cc, Cn et f.
- Lors de la seconde phase (phase de propagation), on résoud Cc et on obtient une évaluation de toutes les variables par propagation.

Les travaux menés par Alan Borning *ref* pour mettre au point le résolveur Cassowary ont aussi porté sur l'aspect incrémental des opérations d'ajout et de retrait de contraintes. En effet dans un contexte d'édition on ne veut pas résoudre le système de contraintes depuis le début à chaque opération, et on gagne beaucoup en performances à adopter des algorithmes incrémentaux.

Cet algorithme permet d'examiner tout l'espace de solutions (s'il y a une solution, le résolveur la trouve) en un temps bien moindre que les algorithmes de retour arrière.

Conclusion sur les résolveurs

Dans Kaomi on dispose de deux résolveurs de contraintes : DeltaBlue (propagatiuon locale) et Cassowary (basé sur le simplex).
DeltaBlue est très rapide mais ne se révèle utilisable que dans des contextes bien précis. On pourra l'utiliser lors de la modification de variables, mais pas lorsqu'on en introduira des nouvelles.
Cassowary quant à lui est un résolveur qui peut être utilisé dans tous les cas de figure, mais la manipulation de systèmes de contraintes complexes posera des problèmes de délai dans le maintien de solution.
On peut toutefois optimiser le temps de calcul en effectuant des prétraitements sur le système de contraintes. En effet, grâce à l'algorithme de filtrage PC2, on réduit les domaines de validité des variables avant d'appliquer le résolveur.

Afin de gérer les ressources dans le formatage temporel, il faut pour cela traduire chaque contrainte liée aux ressources en relation, à savoir poser des variables et des contraintes sur ces variables et ensuite fournir ces données au résolveur.
On va utiliser le résolveur Cassowary pour éviter les problèmes liés à la perturbation du système.

5.2.3 Implémentation

On pose donc plusieurs types de contraintes sur la ressource :

- Contraintes d'utilisation continue : une tache ne doit pas utiliser une ressource plus d'un certain temps. Toutes les variables de durée d'une tache sont donc contraintes par ce temps maximal.

- Contraintes de taux d'utilisation : pour chaque ressource on pose une variable égale à la somme des taux d'utilisation de cette ressource a tout moment. Cette variable ne doit jamais être supérieure a 100 (seconde contrainte).
- Contraintes de coût : on pose une variable coût pour chaque tâche, égale a la somme des produits (coût horaire de la ressource * durée effective de la tâche) pour chaque ressource. On pose ensuite une variable coût total qui peut être contrainte via l'interface d'édition selon les besoins de l'auteur.

Cet ensemble de variables et de contraintes forme donc un système de contraintes. Afin de privilégier certaines modifications plutôt que d'autres, on donne des poids aux contraintes. On veut pouvoir modifier les paramètres de durée, mais ne pas modifier les coûts horaires ou les temps d'utilisation maximale. On utilisera donc des temps d'utilisation maximale. On utilisera donc des *stay constraints* (interdisant la modification de la variable) sur les temps maximaux et les coûts horaires pour s'assurer que ces variables ne sont pas modifiées. Ensuite, on établira les poids des contraintes : les contraintes sur le taux d'utilisation inférieur à 100% sont requises (poids *required*), les contraintes sur les durées d'utilisation sont fortes (poids *strong*), et enfin les contraintes sur les coûts des tâches sont faibles (poids *weak*), car estimées moins importantes que celles sur la durée d'utilisation. Le paramètre *durée effective* de chaque tâche n'est pas contraint par une *stay constraint*, c'est donc celui qui sera le plus apte à être modifié.
En intégrant ce système de contraintes à celui existant pour la gestion temporelle, et en soumettant ce système au résolveur de contraintes, on dispose alors d'un formatage temporel du workflow qui tient compte des spécifications temporelles (durées min, max, pref, et relations) mais aussi de l'allocation des ressources (taux d'utilisation, temps maximal, coût).

6 Conclusion

Après avoir étudié le formalisme de description d'un workflow, et la structure de la boîte à outils Kaomi, j'ai donc implémenté un éditeur de workflow basé sur cette boîte à outils, et tirant partie des différents services fournis comme les différentes vues, l'édition des attributs ou le formatage temporel du document.
L'éditeur, à terme, devrait disposer d'un module de gestion des évènements pour l'exécution du workflow proprement dite et proposer des statistiques basées sur les résultats d'exécution afin d'aider l'auteur à adapter sa planification.
Lors de ce stage à l'INRIA, je me suis familiarisé avec l'environnement Java : héritage, classes abstraites, conventions de programmations… J'ai du apprendre à appréhender un « gros » code comme celui de Kaomi grâce aux documentations automatiques (javadoc), ou en s'adressant aux différents membres de l'équipe selon les parties de l'application qui posaient problème.

Bibliographie :

[1] ALLEN J.F, “Maintaining Knowledge about Temporal Intervals”, *Communications of the ACM*, vol 26, num.11, pp 832-843, November 1983.

[2] BRAY T. et SPERBERG C.M., “Extensive Markup Language (XML)”, (WD-xml-961114), *World Wide Web Consortium*, novembre 1996.

[3] BUCHANAN C., ZELLWEGER P.T., « Specifying Temporal Behavior in Hypermedia Documents », *Proceedings of the ACM, Conference on Hypertext*, pp 262-271, December 1992.

[4] DAVULCU H., KIFER M., RAMAKRISHNAN C.R., RAMAKRISHNAN I.V, « Logic based Modeling and Analysis of Workflows », *.PODS 1998*: 25-33.

[5] EDER J., PANAGOS E., RABINOVICH M., « Time Constraints in Workflow systems », *CAiSE'99, Springer Verlag, LNCS 1626,* Heidelberg, Germany, June 1999, pp. 286-300.

[6] FARGIER H., JOURDAN M., LAYAIDA N., VIDAL T., « Using Temporal Constraints Networks to manage temporal scenario of multimedia documents *», ECAI 98 Workshop on Spatial and Temporal Reasoning*, Brighton (UK), August 1998.

[7] HOFSTEDE A.H.M., ORLOWSKA M.E., RAJAPAKSE J., « Verification Problems in conceptual workflow specifications », *Data & Knowledge Engineering 362* (1998) 239-256.

[8] HyTime Information Technology, « Hypermedia / Time-based Structured Language (HyTime) », *ISO/IEC 10743*, novembre 1992.

[9] JOURDAN M., LAYAIDA N., SABRY-ISMAIL L., « Madeus, an Authoring Environment for Multimedia Documents », *ACM Multimedia’98*, September 1998

[10] M. Jourdan, C. Roisin, L. Tardif, ``A Scalable Toolkit for Designing Multimedia Authoring Environments'', *numéro spécial `Multimedia Authoring and Presentation: Strategies, Tools, and Experiences' de Multimedia Tools and Applications Journal, Kluwer Academic Publishers,* 1999.

[11] M. Jourdan, C. Roisin, L. Tardif, L. Villard, ``Authoring SMIL documents by direct manipulations during presentation'', *World Wide Web, Balzer Science Publishers*, vol. 2, num. 4, décembre 1999.

[12] KAFEZA E., KARLAPALEM K., « Temporally Constrained Workflows *», Lecture Notes in Computer Science* No. 1749, 1999.

[13] LAYAIDA N., SABRY-ISMAIL L., ROISIN C., « Dealing with uncertain durations in synchronized multimedia presentations », *Multimedia Tools and Applications Journal, Kluwer Academic Publishers,* vol. , num. , pp. , 2000.

[14] MACKWORTH A.K., FREUDER E.C., « The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems », *Artificial Intelligence*, 25 (1), pp. 65-74, 1985.

[15] QUINT V., VATTON I., “An introduction to Amaya”, *World Wide WebJournal*, 2(2), pp. 39-46, Printemps 1997

[16]. QUINT V., FURUTA R., ANDRE J., ``Systems for the Manipulation of Structured Documents", *Structured Documents,* ed., pp. 39-74, Cambridge University Press, 1989.

[17] Workflow Module, en ligne sur http://cne.gmu.edu/modules/workflow/workflow-intro.html.

[18] YOO J., LEE D., « X-MAS : Mobile Agent Platform for Workflow Systems with Time Constraints », *ISADS'99(International Symposium on Autonomous Decentralized Systems)*

Annexe 1 : DTD du modèle de documents workflow

```
< !-- Element de Base -->
< !ELEMENT workflow (resourcelist,(composite)+)>

< !ELEMENT resourcelist  ((resource)+)>

< !ELEMENT resource ANY>
< !ATTLIST resource
            nom CDATA #IMPLIED
            temps CDATA #IMPLIED
            cout CDATA #IMPLIED
>

<!ELEMENT basic ANY>
<!ATTLIST basic
          id CDATA #IMPLIED
          nom CDATA #IMPLIED
          duree CDATA #REQUIRED
          ressources CDATA #IMPLIED
>

<!ELEMENT composite ((composite|basic)+, relations ?) >
<!ATTLIST composite
           id CDATA #IMPLIED
 >

<!ELEMENT relations (Temporal)>

<!ENTITY % operands "
            Interval1 CDATA #REQUIRED
            Interval2 CDATA #REQUIRED">

<!ENTITY % operands_delay "
            Interval1 CDATA #REQUIRED
            Interval2 CDATA #REQUIRED
            Delay CDATA #IMPLIED
  Delay2 CDATA #IMPLIED" >

<!ELEMENT Temporal (Equals | Meets | Met_by | Finishes | Finished_by | Starts |
            Started_by | Kills | Killed_by | Parmin | Lipsync | Lipsync_by | Before |
            After | During | Contains | Overlaps | Overlaps_by | Begins | Ends)+>

<!ELEMENT Equals EMPTY>
<!ATTLIST Equals %operands;>

<!ELEMENT Meets EMPTY>
<!ATTLIST Meets %operands;>

<!ELEMENT Met_by EMPTY>
<!ATTLIST Met_by %operands;>

<!ELEMENT Finishes EMPTY>
<!ATTLIST Finishes %operands;>

<!ELEMENT Finished_by EMPTY>
<!ATTLIST Finished_by %operands;>

<!ELEMENT Starts EMPTY>
```

```
<!ATTLIST Starts %operands;>

<!ELEMENT Started_by EMPTY>
<!ATTLIST Started_by %operands;>

<!ELEMENT Kills EMPTY>
<!ATTLIST Kills %operands;>

<!ELEMENT Killed_by EMPTY>
<!ATTLIST Killed_by %operands;>

<!ELEMENT Parmin EMPTY>
<!ATTLIST Parmin %operands;>

<!ELEMENT Lipsync EMPTY>
<!ATTLIST Lipsync %operands;>

<!ELEMENT Lipsync_by EMPTY>
<!ATTLIST Lipsync_by %operands;>

<!ELEMENT Before EMPTY>
<!ATTLIST Before %operands_delay;>

<!ELEMENT After EMPTY>
<!ATTLIST After          %operands_delay;>

<!ELEMENT During EMPTY>
<!ATTLIST During %operands_delay;>

<!ELEMENT Contains EMPTY>
<!ATTLIST Contains %operands_delay;>

<!ELEMENT Overlaps EMPTY>
<!ATTLIST Overlaps %operands_delay;>

<!ELEMENT Overlaps_by EMPTY>
<!ATTLIST Overlaps_by %operands_delay;>

<!ELEMENT Begins EMPTY>
<!ATTLIST begins %operands_delay;>

<!ELEMENT Ends EMPTY>
<!ATTLIST Ends %operands_delay;>
```

ANNEXE 2 : Exemple de document workflow et résultats produits

Exemple de fichier

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Generate by Madeus Authoring Tool-->

<!-- DOCTYPE workflow SYSTEM "/home/Workflow.dtd" -->
<workflow>
<ressources>
                    <ressource nom="secretaire" temps="9h"/>
  <ressource nom="directeur" temps="5h"/>
</ressources>
 <composite id="Workflow" duree="min:4s max:250s pref:304s">
   <composite id="comp1" duree="min:4s max:250s pref:154s">
    <composite id="TOTO"  duree="min:4s max:250s pref:154s">

      <basic id="basic1" nom="Faire la vaisselle" duree="min:4s max:250s pref:62s" ressources="secretaire:0.5 directeur:0.9" />
      <basic id="basic1bis" nom="Faire la vaisselle et frotter" duree="min:4s max:250s pref:61s" />
      <composite id="comp 3" duree="pref:73s">
        <basic id="basic2" nom="Ranger Vetements" debut="10s" fin="5s" duree="min:4s max:250s pref:30s" />
                                                                                <basic id="basic2bis" nom="Ranger CDs" duree="min:4s max:250s pref:34s" />
                                                                                <relations>
                <meets Interval1="basic2" Interval2="basic2bis"/>
                                                                                </relations>
                                                                                </composite>
                                                                                <basic id="basic3" nom="Faire la tambouille" duree="min:4s max:250s pref:11s" />
                                                                                <relations>
                <meets Interval1="basic1" Interval2="basic1bis"/>
                                                                                </relations>
    </composite>
    <basic id="basic 4" nom="Laver les carreaux" duree="min:4s max:250s pref:52s" />
    <basic id="basic 5" nom="Laver le sol" duree="min:4s max:250s pref:121s" />
  </composite>
 </composite>
</workflow>
```

Vue timeline de ce document