الجمهورية الجزائرية الديموقراطية الشعبية
République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE SAAD DAHLEB DE BLIDA

FACULTE DE SCIENCES EXACTES

DEPARTEMENT D'INFORMATIQUE

Mémoire de projet de fin de cycle
en vu d'obtention du diplôme
d'ingénieur d'état en génie informatique

Option : Intelligence Artificielle

Thème

CONCEPTION ET RÉALISATION D'UN ENVIRONNEMENT D'ÉDITION DE DOCUMENTS MULTIMÉDIAS BASÉ SUR LE STANDARD SMIL

Proposé par :
Mme. A. ELMAOUHAB
Encadré par :
Mme. H. BOUALI

Jugé par :

M. Boukhlef (President)
Mme. Sellali (Membre)
M. Oueld aissa. (Membre)

Présenté par :

M. HAMDI Mohamed Nadjib

M. LAKHDARI Yacine

Année Universitaire 2003-2004

Remerciements

Tout d'abord nous remercions Dieu pour sa faveur.

Nous tenons à remercier toutes les parties qui nous ont aidé à réaliser ce modeste travail.
Nous remercions aussi notre promotrice Mme A.ELMAOUHEB.
Nous remercions surtout notre co-promotrice Mme H.BOUALI pour son aide, sa patience et sa disponibilité.
Nous remercions tout particulièrement, nos parents respectifs pour leur patience, leurs encouragements et leurs soutiens précieux tout au long de nos études.

Merci à tous ceux qui, de près ou de loin, ont fait que l'aboutissement de ce travail soit possible, sans oublier M.DAHMENE AICHOUCHE Rabah.

Dédicaces

Je dédie ce travail :

Tout d'abord à mes chère parents et mes chère frères Nadir et Nazim.
A mes grands parents et mes oncles.
A mon amis HAMDI Mohamed Nadjib et toute sa famille.
A tous mes amis.

Yacine.

Dédicaces

Je dédie ce travail :

A mes chère parents, mes frères Azzeddine et Oussama,
et à toute ma famille.
A mon ami LAKHDARI Yacine et à toute sa famille.
A tous mes amis sans exception, sans oublier
I.MAHDJOUBI et toute sa famille.

Nadjib.

Résumé

Les outils d'édition de documents multimédia existant actuellement se rapprochent plus d'outils de programmation (utilisation de langages de scripts) que de véritables outils d'édition. Dans le but de ne pas limiter les auteurs potentiels de documents multimédia aux seuls experts informaticiens, notre travail consiste donc à développer un prototype de système d'édition de documents multimédia basé sur le standard SMIL. Autrement dit ,réaliser un outil logiciel qui permet de composer différents médias (son, image, vidéo et texte) pour produire en finale une présentation multimédia ,cette présentation est sauvegardée sous forme d'un fichier standardisé (***.smi** ou ***.smil**) qui sera lu par la suite par des lecteurs spécialisés.

Notre système doit être facile d'utilisation, ne comportant pas d'étapes où l'on impose à l'utilisateur de faire de la programmation pure (saisir du code).

Mots clés

Document multimédia, SMIL, médias, dimension temporelle, vérification de cohérence, lien hypermédia, application(ou système) multimédia, édition de documents multimédias, la synchronisation .

Table des matières

Table des figures

Liste des abréviations et acronymes

SMIL	Synchronized Multimedia Integration language.
URI	Uniform Resource Identifiers.
WWW	Word Wide Web.
WYSIWYG	What You See Is What You Get.
W3C	Word Wide Web Consortium.
XML	eXtensible Markup Language.
SYMM	SYnchronized MultiMedia.

Introduction générale

Les percées technologiques récentes en matière de multimédia ont permis d'accroître les possibilités d'interaction entre l'homme et la machine. La manipulation digitale, de graphiques, de son et d'images animées sur des stations de travail ou des ordinateurs personnels a changé la nature d'un grand nombre d'applications. En particulier, les applications de traitement de documents électroniques, habituellement dédiées à la création et à la présentation de données textuelles et graphiques, trouvent dans le multimédia des possibilités nouvelles. L'information qu'elles manipulent est plus riche, puisqu'elles intègrent dans ces documents du son et des images animées. Ces nouveaux types de documents électroniques sont communément appelés *documents multimédia*.

Les documents ont jusqu'ici été abordés principalement sous l'angle de leur structure logique (organisation en chapitres, sections, paragraphes, etc.), de leur structure spatiale (présentation graphique et mise en page). Un nouveau type de structure est maintenant considéré : la structure temporelle qui décrit l'enchaînement des éléments dans le temps.

L'intégration de cette nouvelle dimension dans la structure globale d'un document, ainsi que l'introduction d'éléments de base qui ont eux-mêmes une dimension temporelle (vidéo, audio, interactions de l'utilisateur...), constituent l'objet de ce mémoire.

Les standards et les modèles employés pour représenter les documents classiques sont devenus inadaptés pour la représentation de tels documents. De ce fait, de nouveaux standards comme SMIL, émergent pour les compléter. Mais les standards ne suffisent pas, il faut des outils pour la création, la modification et la présentation de documents multimédias synchronisés, ces derniers sont très peu.

Le travail présenté dans ce mémoire a pour objectif de concevoir et d'implémenter un environnement d'édition de documents multimédias synchronisés, en considérant de façon prioritaire les besoins des auteurs, et tout en facilitant l'utilisation grâce aux outils graphiques de création de documents multimédias.

Plan du mémoire :

Ce mémoire est organisé en trois chapitres, dont le contenu sera détaillé ci-dessous :

Chapitre 1 :

Dans ce chapitre nous donnerons un aperçu sur quelques concepts du multimédia et sur la synchronisation, ensuite nous passerons à l'étude de quelques stratégies suivies par les concepteurs d'environnements d'édition de documents multimédias pour l'élaboration d'outils d'édition.

Chapitre 2 :

Ce chapitre sera consacré à l'étude du langage SMIL, cette étude sera organisée en trois parties: la première présentera le langage SMIL et ses avantages ainsi que la décomposition de l'architecture d'un document en structure spatiale (positionnement des différentes régions qu'occupe le média), temporelle (synchronisation temporelle des objets multi- médias). La deuxième partie décrira la syntaxe et la sémantique de ce langage, enfin la troisième partie présente les lecteurs actuels des fichiers SMIL ainsi qu'une critique de quelques éditeurs SMIL que nous avons pu manipuler.

Chapitre 3 :

Nous aborderons dans ce chapitre la démarche de développement de notre système, en commençant par la partie conception dans laquelle nous essayons de combler les manques observés dans les différents environnements auteurs décris dans le chapitre précédent (chapitre 2) ; ensuite nous aborderons la partie implémentation dans laquelle nous

présentons les interfaces qui composent notre éditeur, ainsi que les algorithmes utilisés pour l'implémentation du modèle de document SMIL, et ceux utilisés pour la vérification de la cohérence du document.

Enfin, dans la conclusion, nous résumerons l'apport essentiel de ce travail, et nous proposerons quelques perspectives qui pourront améliorer et enrichir notre prototype d'édition.

Chapitre 1

Le multimédia et l'édition de documents multimédias

1.1 Introduction

Au début des années quatre-vingt dix, l'apparition d'une nouvelle génération d'ordinateurs, et les avancées des techniques d'exploitation des différents médias, ont provoqué l'explosion du multimédia [1], qui se réfère communément à la capacité de l'ordinateur et des logiciels à manipuler et restituer des données autre que le texte et l'image, on désigne ici principalement le son, la vidéo et les présentations virtuelles, comme dans les jeux vidéo [1].

Cette évolution du multimédia a transformé la manière avec laquelle l'utilisateur communique avec sa machine, il peut à la fois travailler, naviguer et écouter de la musique sur la même machine, car le multimédia offre la possibilité de manipuler le son, l'image et le texte simultanément. On trouve beaucoup d'activités liées au multimédia tel que la vidéo- conférence, la formation, et les ateliers de création qu'ils soient artistiques ou techniques [2].

On peut distinguer deux grandes catégories d'application du multimédia, *la téléconférence* qui permet à plusieurs personnes se trouvant à des lieux distincts de participer à une réunion virtuelle, et le domaine des *présentations multimédias* ou *documents multimédias* [1].

1.2 Le multimédia

1.2.1 **Définition**

Le multimédia est « L'ensemble des techniques et des produits qui permettent l'utilisation simultanée est interactive de plusieurs modes de représentation de l'information (texte, son, images fixes ou animées) » [3].

Les progrès accomplies dans le domaine du matériel, des systèmes d'exploitation, et des techniques de codage et modélisation de données, ont permis au multimédia de se rapprocher vers son but ultime, qui est de pouvoir fournir un moyen d'interaction avec des données en tout genres (texte, son, vidéo ...etc.), qui soit simple et rapide en faisant abstraction de la technique informatique nécessaire à l'obtention du résultat [1].

Le multimédia fait appel à deux techniques voisines *l'hypertexte* et *l'hypermédia* :

- *l'hypertexte* :

  Il fut proposé en 1965 par Ted Nelson, s'appliquant à une bibliothèque de fichiers; la technique « ***Hypertexte*** » consiste à chaîner ensemble des fichiers par des relations, ces liens permettent ainsi à l'utilisateur de naviguer entre ces fichiers [4].

- *l'hypermédia* :

  La technique Hypermédia, similaire au mode hypertexte, s'applique aux documents composés de textes, de sons, d'images et de toutes combinaisons de ces derniers, il présente ces données de la même manière que la technique hypertexte [4].

1.3 Système multimédia (ou Application multimédia)

Cette nouvelle façon d'exploiter l'ordinateur a fait apparaître chez les utilisateurs un besoin de personnalisation de ce qu'ils manipulent, pour satisfaire à cette exigence un nouveau domaine de développement orienté vers les systèmes multimédias et d'édition de documents multimédias est apparu.

Le côté dynamique induit par, la nature temporelle de certaines données exploitées par l'utilisateur, et par les possibilités d'interactions auxquelles le lecteur a accès, ont conduit certains concepteurs à parler d'*applications multimédias*; on peut alors qualifier un système

(ou une application) de « multimédia », s'il (ou elle) supporte le traitement intégré de plusieurs médias dont au moins un est de nature temporisée [5].

Cette définition est basée sur une classification qui qualifie un système de multimédia, tout système se caractérisant par ces trois critères [5] :

- le nombre de médias manipulés dans l'application, comme l'audio, la vidéo, le texte.
- la nature temporelle des médias supportés (continus comme la vidéo, statique comme le texte).
- le niveau d'intégration de ces différents médias au sein de l'application (désigne la capacité d'une application à rendre homogène le traitement de différents médias de natures différentes).

Une fois combinés, ces critères permettent de dire si une application est « plus » multimédia qu'une autre [5].

1.4 Document multimédia et modèle de document

1.4.1 **Document multimédia**

Un document multimédia peut être défini comme un ensemble de médias de natures diverses (son, image, texte, vidéo), regroupés et organisés spatialement et temporellement pour constituer une unité sémantique qui est présentée à des personnes [6].

Le temps représenté dans ce type de document est le principal facteur à prendre en considération, car tout document multimédia est décrit par un *scénario temporel* qui montre l'enchaînement dans le temps des différents éléments le composant, et on distingue deux types de scénarios [7] :

- *les scénarios déterministes* :

  Ils décrivent les documents dont on connaît le moment de début et fin de tous les éléments les composant.

- *les scénarios indéterministes* :

  Ils décrivent les documents dont la durée de certains de leurs éléments est modifiable en cours de présentation, ou ceux dont on ne connaît la durée qu'au moment de la présentation.

1.4.2 **Pourquoi des documents multimédia (cas d'utilisations)**

De nombreux secteurs d'activité sont concernés par l'émergence des techniques du multimédia: les jeux, le tourisme, l'enseignement assisté par ordinateur (EAO), sont actuellement le cadre des plus nombreuses applications multimédias. L'EAO, par exemple, peut ainsi tirer parti des caractéristiques des différents médias pour réaliser des supports pédagogiques qui soient d'une part plus attractifs grâces aux images, aux animations et au son, et d'autre part plus interactifs et adaptables aux élèves grâce aux fonctions de navigation hypermédia [6].

D'autres domaines s'intéressent aux documents multimédias notamment la médecine; les données issues de l'imagerie médicale, comme les images par rayons X, par résonance magnétique ou par échographie, peuvent être exploitées sous forme numérique et intégrées à des données textuelles (par exemple les informations relatives au patient), et sonores (les commentaires du médecin), pour former de véritables documents multimédias médicaux qui peuvent être consultés à distance par des médecins ou par le patient lui-même [6].

L'apport du multimédia dans ce type d'application vient non seulement de la dynamique et de l'interactivité, mais aussi de l'augmentation de la qualité de l'information présentée. La variété des types de média utilisés apporte de nouvelles possibilités aux créateurs en leur permettant de jouer sur un plus grand nombre de modes de perception [6].

1.4.3 **Modèle de document multimédia**

Un modèle de documents multimédias, permet la représentation de toutes les relations qui peuvent exister entre les différents éléments d'un document multimédia, ces relations correspondent à l'organisation logique, la présentation spatiale, la synchronisation et l'interconnexion entre les différents éléments du document (l'hypermédia) [5], on présente ainsi les différentes dimensions d'un document multimédia.

1.4.3.1 **Dimension logique d'un document multimédia**

C'est l'architecture du document, elle représente la liaison sémantique entre différentes parties ou éléments du document, ces parties peuvent être regroupées par des relations logiques. Ce regroupement constitue de nouvelles entités logiques dans la structure interne du document [5]. Ces dernières peuvent elles aussi être regroupées pour constituer d'autres entités logiques plus complexes. (Voire Figure 1.1).

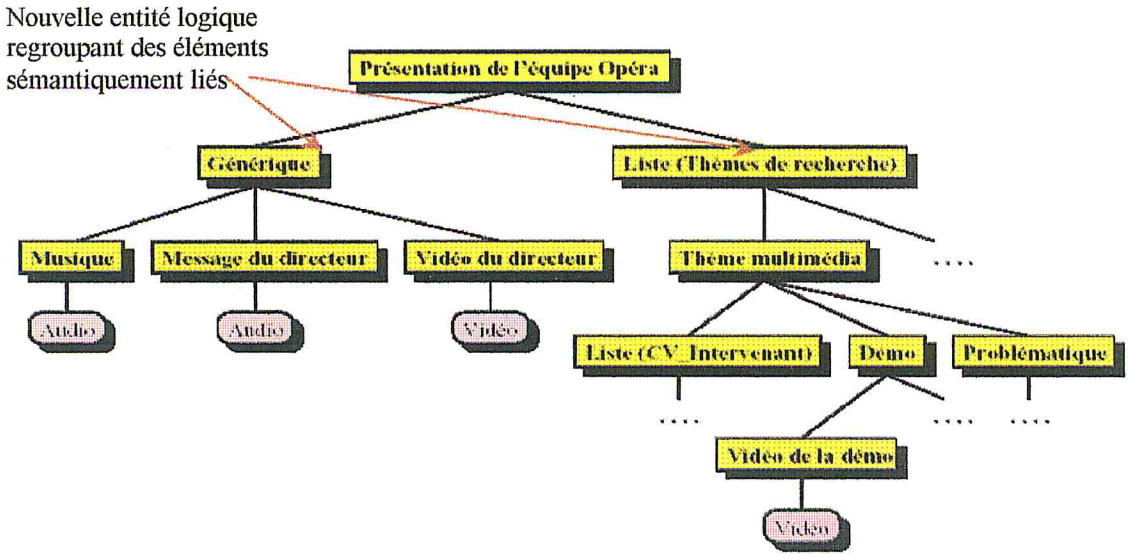


Figure 1.1 : *Exemple d'une structure logique d'une présentation multimédia (Projet Opéra).*

Cette construction décrit des relations d'inclusion entre les éléments logiques du document, on peut donc décrire le schéma de structure correspondant à l 'exemple comme suit (en Pascal) [5] :

```
Présentation = BEGIN
          Introduction = Générique;
          Corps = LIST OF (Thèmes_de_recherche);
          END;
Générique = BEGIN
          Musique = AUDIO;
          Message_du_directeur = Message;
          Vidéo_du_directeur = VIDÉO;
          END;
Message = LIST OF (CASE OF
          TEXT;
          AUDIO;
          END);
Thèmes_de_recherche = BEGIN
          Schéma = LIST OF (CV_intervenant);
          Problématique = ... ;
          END;
```

1.4.3.2 **Dimension spatiale d'un document multimédia**

Elle concerne l'allocation des ressources physiques aux éléments composant un document multimédia pendant le temps qui leurs est nécessaire. Par exemple l'allocation d'une certaine zone de la ressource *écran* à un élément vidéo, ou d'un *canal audio* (ressource) pour un élément audio pour toute la durée de sa présentation. Cette opération de mise en correspondance entre les différents éléments multimédia constituant un document, et les ressources qui leurs sont allouées durant une période temps donnée est appelée *formatage spatio-temporel* [5].

Le formatage spatio-temporel généralise la notion de *formatage géométrique ;* comme est le cas des éditeurs de documents statiques dont le formatage est restreint au positionnement géométrique d'un document, comme la définition de règles géométriques (taille de la page, des colonnes, ...etc.), et les caractéristiques géométriques des éléments du document ; ce qui n'est pas suffisant pour des documents multimédias, car ces documents présentent un aspect dynamique, où les espaces géométriques et les canaux sonores requis sont alloués, libérés, mixés au cours de la présentation d'où la nécessité d'un formatage dans le temps au même moment que celui dans l'espace [5].

1.4.3.3 **Dimension hypermédia d'un document multimédia**

Les liens dits « hypermédia » permettent de définir des relations entre différents documents multimédias ou parties du document, ils constituent ainsi un support de navigation la manière du Web; ces liens sont définis par une ancre de départ (désignant l'élément source du lien), et une ancre d'arrivée (désignant l'élément destination du lien); et ils nécessitent d'être activés explicitement par l'utilisateur au niveau de l'interface qui lui est présentée [5].

1.4.3.4 **Dimension temporelle d'un document multimédia**

C'est la caractéristique majeure d'un document multimédia, elle permet de définir les méthodes permettant le support de la structure temporelle du document, et rend compte de toutes les dépendances temporelles entre les composants de ce document. C'est ce qui permet la synchronisation des composants texte, image, son et vidéo [5].

Les éléments composant le document sont reliés temporellement de sorte à définir l'ordre de ces derniers lors de la présentation. La synchronisation temporelle représente essentiellement les dépendances temporelles des éléments du document [5].

On distingue trois types d'intervalles temporels qui représentent la synchronisation, ceci en fonction de leurs rôles (les intervalles) dans la présentation du document [5] :

- *les intervalles associés aux éléments logiques du document :*
  C'est l'intervalle abstrait délimité par l'instant de début du premier élément multimédia contenu dans un élément logique, et l'instant de fin du dernier élément multimédia contenu dans ce même élément logique.

- *les intervalles de style temporel :*
  Ils représentent des intervalles qui ne figurent pas dans la structure logique du document, mais qui sont ajoutés pour des besoins de contrôle du style d'une présentation multimédia. Par exemple, faire défiler progressivement le contenu de la fenêtre pour laisser place à la scène suivante.

- *les intervalles d'espacement et de rupture temporels :*
  Ce type d'intervalle est inséré entre les intervalles des éléments logiques du document et représente des intervalles d'espacement temporel, il correspond à un intervalle de durée fixe entre un élément et son successeur (un délai) ou encore de durée indéterminée correspondant à l'activation d'un bouton par l'utilisateur.

Ces deux derniers types d'intervalles n'ont pas de représentation dans la structure logique du document, néanmoins, ils sont importants pour affiner la présentation d'un document.

1.5 Les éléments multimédias composants un document multimédia

La production d'une présentation, ou d'un document multimédia, est réalisée en combinant différents éléments multimédias aux comportements temporels différents, on définit alors trois types d'éléments [7] :

- des *éléments intemporels*, comme les éléments textuels et graphiques.
- des *éléments déterministes*, comme les éléments audio et vidéo de durée connue.
- des *éléments indéterministes*, comme les programmes, et les flots de vidéo de durée inconnue (par exemple les vidéos représentant la réalité virtuelle qui sont calculées en cours de présentation).

L'intégration de ces éléments dans une présentation, (qui elle même peut être considérée par la suite comme un élément), suit deux schémas [5]:

- *l'intégration au niveau du codage :*

  On produit, un nouveau flux à partir de plusieurs flux multimédia qui sera stocké, transmis, et restitué comme un seul fichier. Ce nouvel élément a des unités de présentation et des contraintes de synchronisation similaires à celles des éléments à partir desquels il a été obtenu.

- *l'intégration par référence :*

  Ce type d'intégration consiste à produire un nouvel élément en référençant des éléments déjà stockés de façon indépendante et externe, ces même éléments peuvent être référencés par plusieurs autres éléments sans avoir à les dupliquer.

Cette combinaison entre toutes les entités d'une présentation définit son scénario temporel; mais ces entités, comment sont elles disposées les unes par rapport aux autres dans le temps ? C'est le problème de « la synchronisation multimédia ».

1.6 La synchronisation multimédia

La synchronisation traite à la fois les problèmes liés aux données temporisées, à leurs combinaisons, et aussi ceux liés à leurs intégrations avec des informations non-temporisées comme le texte et les graphiques, cela étant représenté par des schémas de synchronisation [5].

1.6.1 **Schémas de synchronisation**

On considère trois schémas de synchronisation [5]:

- *synchronisation intra-élément :*

  Elle s'applique aux relations temporelles entre les informations composant un élément multimédia, comme est le cas de la vidéo où la synchronisation apparaît dans la succession des images à une vitesse de défilement de 25 images par seconde, soit une image affichée toutes les 40 millisecondes (voire Figure 1.2).

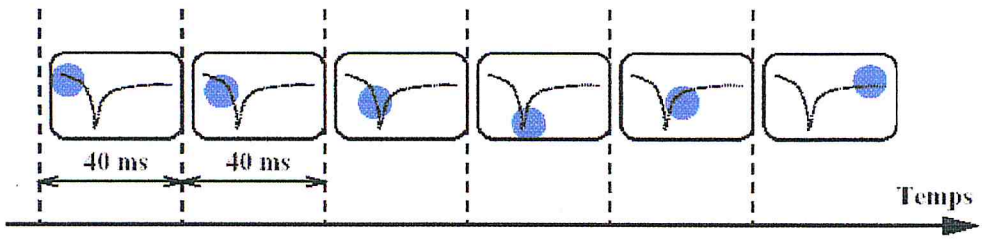


Figure 1.2 : *Représentation de succession d'image d'une séquence vidéo.*

- *synchronisation inter-élément :*

  Celle-ci s'applique aux enchaînements dans le temps de plusieurs éléments multimédia composant une présentation (voire Figure 1.3).

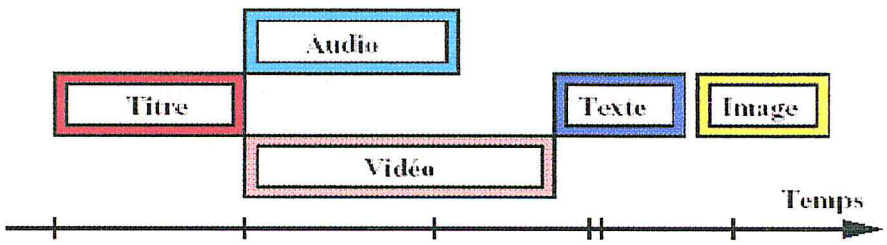


Figure 1.3 : *Enchaînement de différents éléments multimédia.*

- *la synchronisation des lèvres :*

  Elle impose un couplage temporel fort de la progression temporelle entre éléments multimédias; ce couplage est généralement représenté autant qu'un décalage admissible entre les éléments, comme est le cas dans une présentation simultanée d'un discours audio et la séquence vidéo qui lui est associée, pour permettre de maintenir le mouvement des lèvres en accord avec la voix. Une autre illustration de ce type de synchronisation est présentée dans la Figure 1.4.

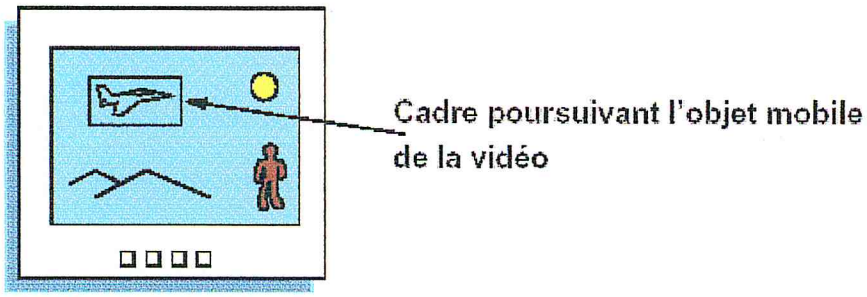


Figure 1.4 : *Exemple de la synchronisation des lèvres.*

1.6.2 **Application de la synchronisation**

L'application de la synchronisation correspond à deux types de contraintes temporelles liées aux différents éléments multimédias [5]:

- la contrainte tirée du monde réel, appelée *synchronisation naturelle*.
- la contrainte spécifiée par le concepteur de l'application, dont l'objectif est de produire des effets temporels, c'est la *synchronisation synthétique* ou *artificielle*.

1.6.2.1 **La synchronisation naturelle**

On trouve dans le traitement de la synchronisation naturelle, le schéma producteur–consommateur dont une configuration logicielle ou matérielle fait intervenir trois composants [5]:

- *une source* : elle permet de numériser l'information comme les caméras ou les microphones.
- *une connexion* : qui permet le transfert des données numérisées vers la destination.
- *une destination* : qui restitue le flux de données reçu au travers de la connexion.

On peut prendre comme exemple de la synchronisation naturelle la vidéo conférence (voir Figure 1.5), où les flux multimédias sont saisis, transférés, puis restitués sur le site destination qui doit respecter les contraintes temporelles prises en compte par la source.

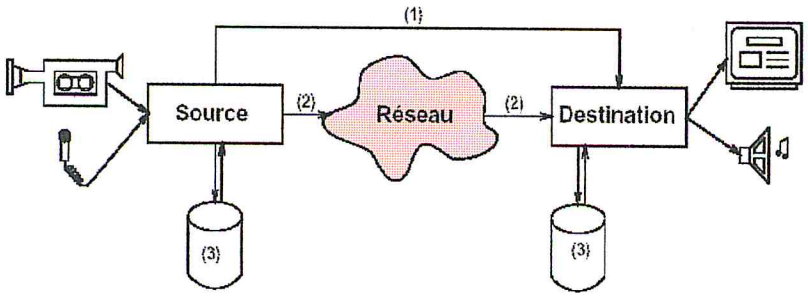


Figure 1.5. *Schéma représentatif d'un système vidéoconférence.*

Ici la difficulté consiste à considérer la synchronisation de bout en bout, car chaque contrainte sur une unité à transférer se traduit par des contraintes globales sur le temps de la vidéo conférence. La solution proposée pour remédier à ce problème, est de faire des adaptations dynamiques de la qualité du flux audio ou vidéo, en fonction de la disponibilité des ressources, On peut supposer par exemple, que le module destination peut décider de réduire la résolution graphique d'une vidéo, afin de diminuer le temps CPU nécessaire au traitement du flux, ou encore cette dernière tache peut être réaliser par la source en ayant des informations sur les ressources de la destination, afin d'envoyer des données moins importantes en taille [5].

1.6.2.2 **La synchronisation synthétique**

Dans la synchronisation synthétique, on retrouve une plus grande variété d'éléments multimédias, comme les animations, les interactions avec l'utilisateur, les images virtuelles …etc.

Ces éléments n'ayant pas de contraintes temporelles intrinsèques, l'auteur doit leurs affecter des valeurs de durée avant de les utiliser. Les contraintes naturelles liées à la vidéo et l'audio peuvent aussi être modifiées, on peut par exemple changer la vitesse d'une vidéo [5].

Elle porte aussi sur la disposition temporelle des éléments multimédia, comme les présenter en parallèle ou en séquentiel. Cette opération appelée *composition temporelle*, est en réalité très complexe, car elle se heurte à la nature temporelle très diverse des éléments multimédias, et à la difficulté de combiner des contraintes réelles et synthétiques [5].

1.7 L'édition et environnement d'édition de documents multimédias

Les documents multimédias intègrent une dimension temporelle en plus des dimensions : spatiale, logique et hypertexte. Cette dimension temporelle est définie par les caractéristiques temporelles des objets multimédias et par l'enchaînement temporel de ces objets. L'édition de documents comportant des objets multimédias (image, son, vidéo, ...) organisés dans la dimension temporelle, est une tâche complexe, car l'entité à construire est une entité dynamique dont il faut spécifier le *scénario temporel* [6]; de plus, le comportement d'un document multimédia peut varier d'une présentation à une autre, à cause des réactions aux interactions du lecteur [6]; de ce fait, la nature dynamique des objets manipulés ainsi que la variété des comportements qui en résulte empêche la fusion simple des phases d'édition et de présentation [6]; pour remédier à ces difficultés les concepteurs des systèmes informatiques ont développé des environnements de composition de documents multimédias, qui ont comme objectif de combiner au mieux un fort pouvoir d'expression (qui intègre les différentes dimensions d'un document, et les différents types d'objets multimédias le composant), avec un mode d'édition interactif et convivial; pour simplifier au mieux ce pouvoir [6].

1.7.1 **Besoins d'un environnement d'édition de documents multimédias**

Un environnement d'édition de documents multimédia doit respecter un certain nombre de critères qui représente les exigences d'un utilisateur, ceci pour aider l'auteur à maîtriser la complexité de la tâche d'édition [6]:

- l'environnement auteur doit être utilisable sans compétences en programmation informatique.

- l'environnement doit permettre à l'auteur de réduire la complexité de son document et de lui faciliter la réutilisation des parties déjà spécifiées. Ceci peut être permis,

par exemple, par l'utilisation d'une méthode de décomposition du document (décomposition hiérarchique et/ou logique).

- l'environnement auteur doit permettre de modifier facilement les documents en cours de conception.

- l'environnement auteur doit offrir des supports visuels permettant de percevoir le scénario en cours de spécification. Cette perception peut se faire de plusieurs façons, soit par une visualisation directe du document, soit par la visualisation du scénario temporel; par exemple avec un axe de temps et des objets représentant les éléments utilisés pour composer le document (voir Figure 1.6).
  Il faut cependant que le basculement vers le mode présentation soit facile.

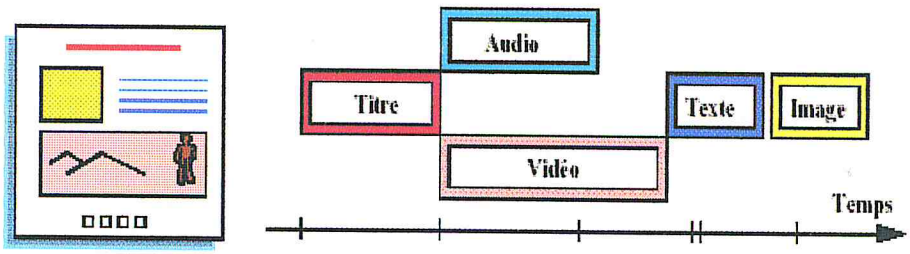


Figure 1.6. *Un document (à gauche) une représentation du scénario temporel (à droite).*

1.7.2 **Les fonctions d'un environnement d'édition de documents multimédias**

Pour qu'un environnement d'édition puisse satisfaire les besoins cités à la section précédente, il doit offrir à l'auteur une interface ergonomique, interactive et simple d'utilisation, de manière à ce que l'auteur puisse manipuler l'objet qu'il est entrain de créer tel qu'il sera aperçu par le lecteur, car la majorité de ces systèmes sont destinés à une clientèle majoritairement non informaticienne.

Pour cela, un environnement d'édition doit comporter principalement deux types de fonctions [5]:

- *des fonctions d'édition :*

  Ces fonctions permettent la création ou la modification d'un document par un auteur, en incluant différents éléments multimédias de base et spécifiant les relations entre ces éléments, ces relations peuvent être liées à leur organisation logique, leur disposition spatiale ou leur synchronisation temporelle.

- *des fonctions de présentation :*

  Ces fonctions permettent de restituer à l'utilisateur le contenu du document une fois achevé, en lui fournissant un ensemble de commandes permettant de l'explorer pour découvrir son contenu à travers l'espace et le temps.

1.7.3 **Besoins de l'édition de document multimédia**

L'édition de documents multimédia a pour but de créer des objets qu'un lecteur peut visualiser, l'auteur utilise en ce sens des éditeurs qui se basent sur le principe « offrir à l'auteur des fonctions qui vont lui permettre de créer un nouvel objet, de le modifier et enfin de le sauvegarder ». La construction d'un système d'édition de documents multimédias doit répondre aux exigences suivantes [5]:

- une représentation structurée du document *:*

  Les aspects liés à l'organisation logique du document doivent être séparés des aspects liés à sa présentation physique et sa synchronisation temporelle, ceux ci permettra de garantir l'indépendance de la représentation du document (son format) par rapport à une plate-forme spécifique.

- un processus d'édition incrémental :

  La création d'un document se fait généralement par des ajouts progressifs à son contenu structuré, l'effet de chaque opération effectuée par l'auteur doit pouvoir être immédiatement vu à la présentation.

- maintien de l'intégrité du document :

  Pour chaque opération d'édition effectuée par l'auteur, le système d'édition doit vérifier sa cohérence par rapport à l'état courant du document.

- intégration des différents aspects d'un document multimédia :

  La nécessité de prendre en compte les différents aspects d'un document multimédia (son organisation logique, sa synchronisation temporelle, sa présentation spatiale et les liens hypermédia).

- intégration de données multimédia hétérogènes :

  Il est primordial de rendre homogène la manipulation de données comme le texte, les graphiques, la vidéo, l'audio et les interactions avec l'utilisateur aux sein d'une même structure de document, cette intégration pose des problèmes liés principalement aux natures très diverses des éléments composant le document (temporisés, statiques, indéterministes).

- résolution automatique des contraintes temporelles :

  Afin d'offrir à l'utilisateur des opérations de synchronisation de haut niveau, le système d'édition doit lui fournir des fonctionnalités qui prennent en charge toutes les contraintes numériques induites par l'établissement de synchronisation du document.

1.7.4 **Catégories d'outils d'édition multimédia**

Il existe différents types d'éditeurs de documents multimédias, qui sont destinés à de différentes catégories d'auteurs plus ou moins spécialisés dans le domaine de l'édition, on peut distinguer quatre grandes catégories d'outils d'éditeurs [5]:

- les éditeurs timelines.
- les éditeurs graphiques.
- les éditeurs structurels.
- les éditeurs orientés programmation.

1.7.4.1 **Les éditeurs timelines**

Ce genre d'éditeurs est le plus répandu sur le marché, les outils de cette catégorie se basent sur une interface graphique qui présente un axe désignant la dimension temporelle d'un document multimédia, ces interfaces représentent les différentes présentations incluses dans un document par des axes parallèles dont chacun est réservé à une

présentation particulière, et un axe commun (commun à tous les autres axes) utilisé pour se référencer au temps globale [5].

L'édition de document dans ce type d'outils est réalisée en plaçant des icônes représentant les différents éléments (images, son, vidéos...etc.) composant une présentation sur l'un des axes (pistes). Ceci permet de définir de manière précise l'enchaînement des différents éléments, et le moment de leur apparition par rapport à l'instant du début du document.

Certains éditeurs permettent d'ajouter des effets sur ces éléments (ces effets sont appelés *propriétés*), comme est le cas de **Diractor** (outil d'édition de présentation multimédias développé par MACROMEDIA).

Voici ci-dessous (Figure 1.7), une scène réalisée avec Diractor, et (Figure 1.8) ci-après la vue timeline suivi de l'inspecteur des objets utilisés pour réaliser cette scène :

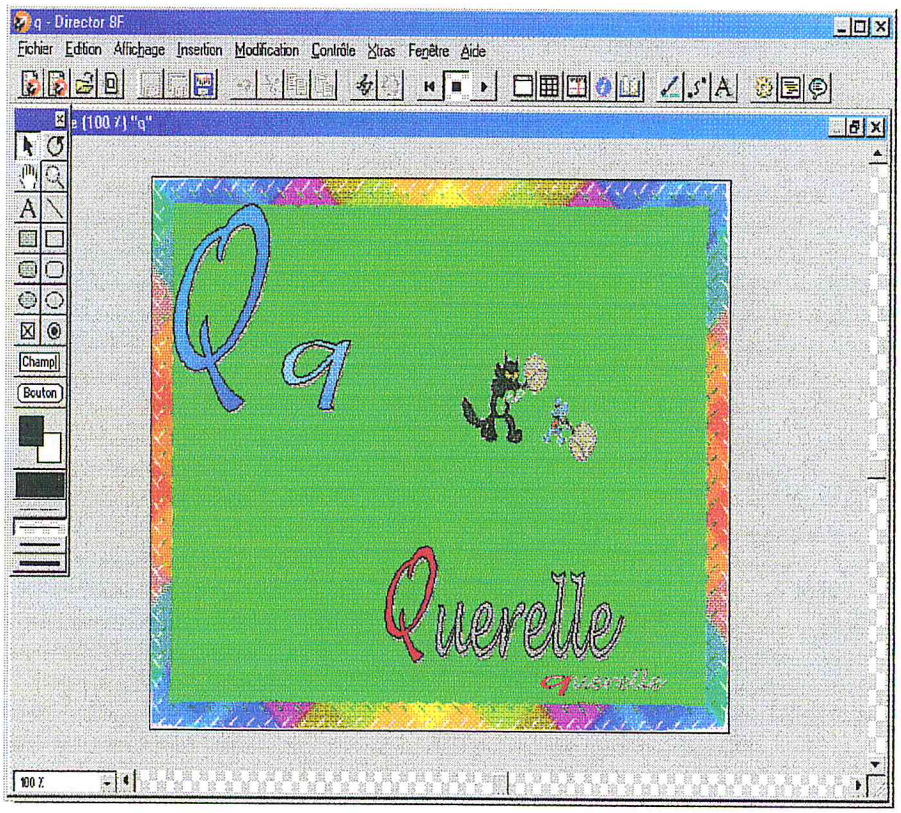


Figure 1.7 : *Interface de présentation de scène de Diractor8.*

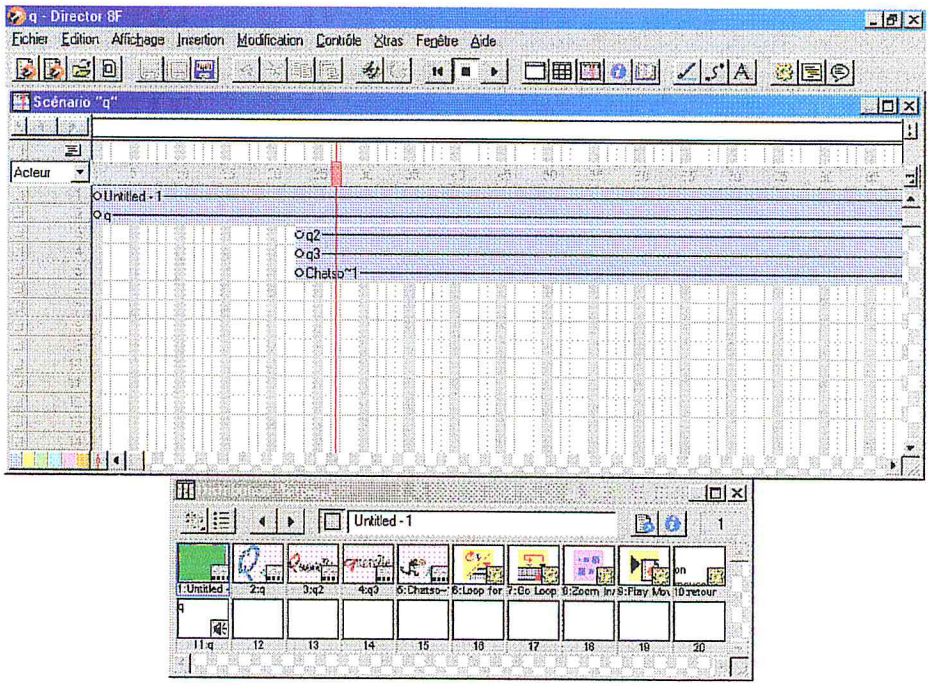


Figure 1.8 : *Vue timeline et inspecteur d'objet de Diractor8.*

Les éditeurs timelines sont bien adaptés aux présentations multimédias, grâce à leurs interfaces utilisateur, qui désignent bien et de façon intuitive le placement temporel de l'objet inséré. Mais, ces éditeurs comportent de multiples inconvénients liés à cette interface, on peut citer [5]:

- une modification du temps associé à un élément, que se soit dans la durée ou le moment d'occurrence, nécessite souvent le repositionnement à la main des autres éléments, car la dépendance temporelle entre les différents éléments n'est pas maintenue de manière relationnelle, tout y est en référence par rapport à l'axe commun, ce qui rend toute opération de changement fastidieuse, et qui est susceptible d'engendrer des erreurs.

- l'impossibilité de représenter des éléments dont les durées sont inconnues, à cause de la datation explicite des éléments.

- l'organisation modulaire du document est très limitée car tout est basé sur le temps et les axes le représentant.

Plusieurs systèmes d'édition à base de timelines sont largement utilisés, dont **Director** (un produit commercial), **Integrator** (un prototype de recherche) et **MAEstro** (un produit commercial issu d'un travail de recherche) [5].

1.7.4.2 **Les éditeurs graphiques**

Dans cette approche, le document est représenté sous la forme du diagramme de son flot de contrôle qui décrit l'interaction entre les éléments constituant le document à l'exécution. Ce type d'éditeurs permet des descriptions puissantes de la synchronisation par rapport aux timelines. Les éditeurs graphiques fournissent un support plus adapté aux opérations d'éditions comme la suppression ou l'insertion d'éléments car la disposition de ces derniers est explicite (Figure 1.9) [5].

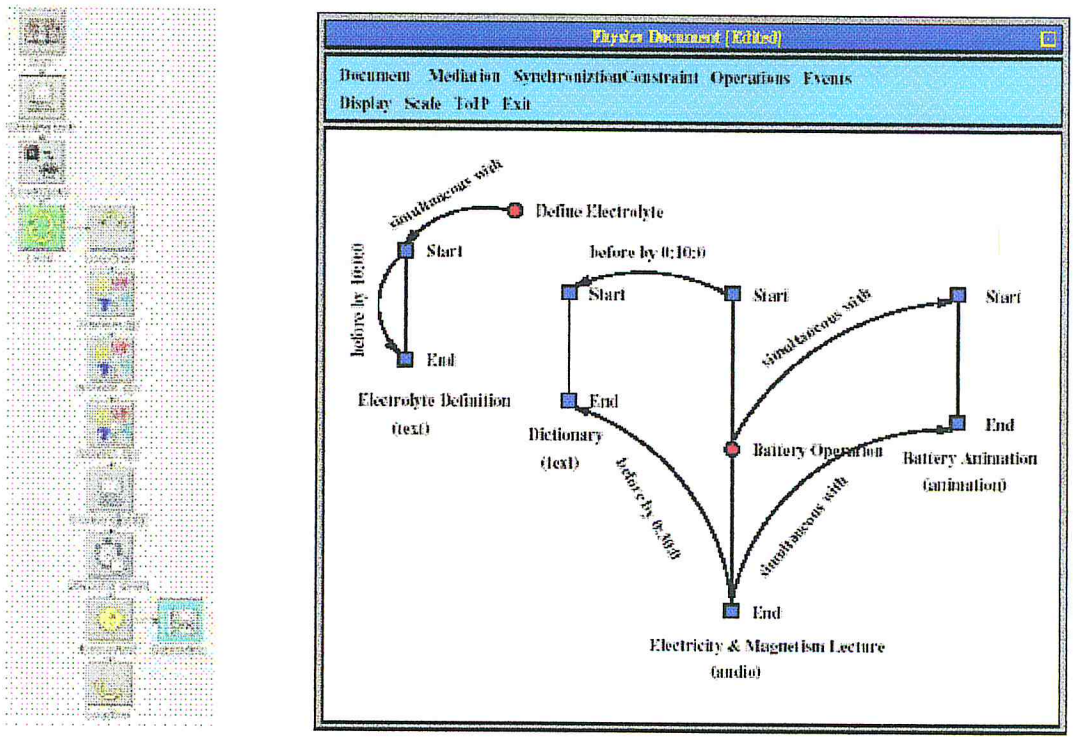


Figure 1.9 : *Interface d'Icon Author (à gauche) et de Firefly (à droite).*

L'inconvénient de ce type d'éditeurs est que la description temporelle d'un document devient rapidement complexe et illisible dés que sa taille augmente, car dans chaque élément intervient différents événements temporels, qui correspondent au début du document et sa fin et parfois ses événements internes, qui sont mises en relation avec d'autres éléments, de plus la représentation du document est très peu structurée ce qui limite la réutilisation de ses parties [5].

1.7.4.3 **Les éditeurs fondés sur la structure du document**

Les éditeurs graphiques et timelines appliquent le même principe pour la création de présentation multimédia, dans ce cas le document multimédia est défini en terme de primitive de synchronisation appliqué directement aux éléments constituants le document. L'édition fondée sur la structure exploite la structure logique du document pour permettre sa synchronisation temporelle, grâce à la possibilité d'organiser le contenu du document en modules isolés sur lesquels on peut appliquer des primitives globales de synchronisation, qui sont la mise en parallèle ou en séquence des éléments appartenant à une même entité logique [5].

CMIFed (*CWI Multimedia Interchange Format Editor*, issu d'une équipe du laboratoire CWI d'Amsterdam) est l'un des outils qui se base sur ce principe, il présente deux vues pour un document multimédia :

- une vue *Hiérarchique* qui présente le document sous forme de boites encapsulées représentant la hiérarchie (dans le temps) des éléments constituants le document (Figure 1.10).

- une vue C*anaux* qui permet d'avoir un aperçu de la dimension temporelle détaillée (Figure 1.11).

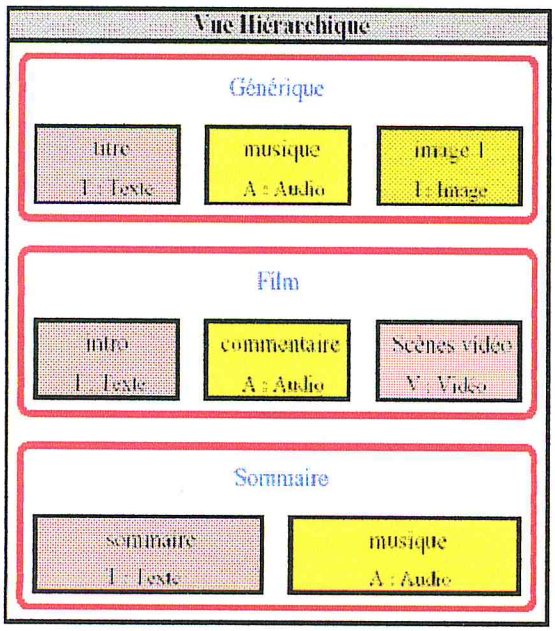


Figure 1.10 : *Vue hiérarchique.*

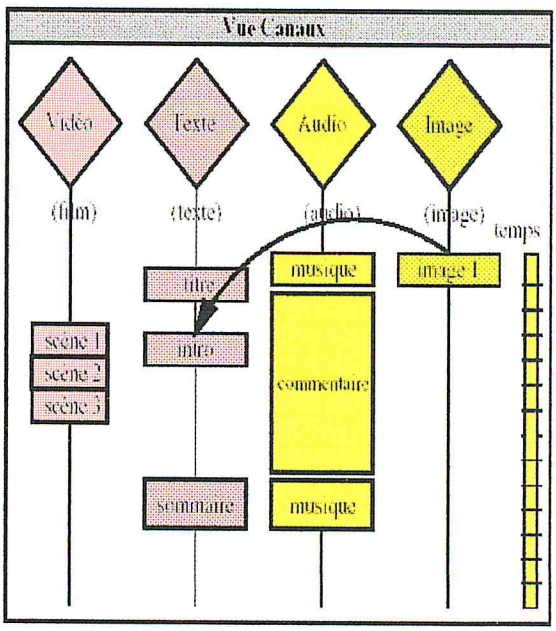


Figure 1.11: *Vue canaux.*

L'avantage de cette approche est la convivialité de son interface utilisateur, qui est plus claire que celle des éditeurs timelines ou graphiques, mais le grand point faible est le pouvoir expressif limité offert par ses primitives de synchronisation, pour résoudre ce problème des arcs de synchronisation sont ajoutés au document (voir Figure 11) [5].

1.7.4.4 **Les éditeurs orientés programmation**

Ce type d'éditeur est basé sur la programmation proprement dite, il offre à l'auteur toute la puissance d'un langage de programmation, qui rend possible la description d'enchaînements illimités que ce soit du point de vus temporel ou spatial, ce qui permet d'écrire des scénarios indéterministes, ce qui n'était pas possible avec les systèmes d'édition précédemment décris. Les problèmes posés par ces outils sont les suivants :

- ils demandent une bonne connaissance du langage utilisé, donc ils sont destinés à des personnes expérimentées en programmation.

- ils subtilisent la notion de document à celle de programme, ce qui implique la description du document par des procédures et algorithmes, ce qui rend toute modification très complexe.

Les concepteurs de langage de programmation ont essayé de remédier à ces problèmes en créant des outils orientés vers la programmation visuelle, dans le but de faciliter la tache à l'auteur en proposant une interface graphique qui présente des outils et contrôles plus simples et conviviaux qui définissent des procédures, c'est le cas par exemple de visuel C++, visuel Basic (Figure 1.12) et les produits builder de Borland [7].

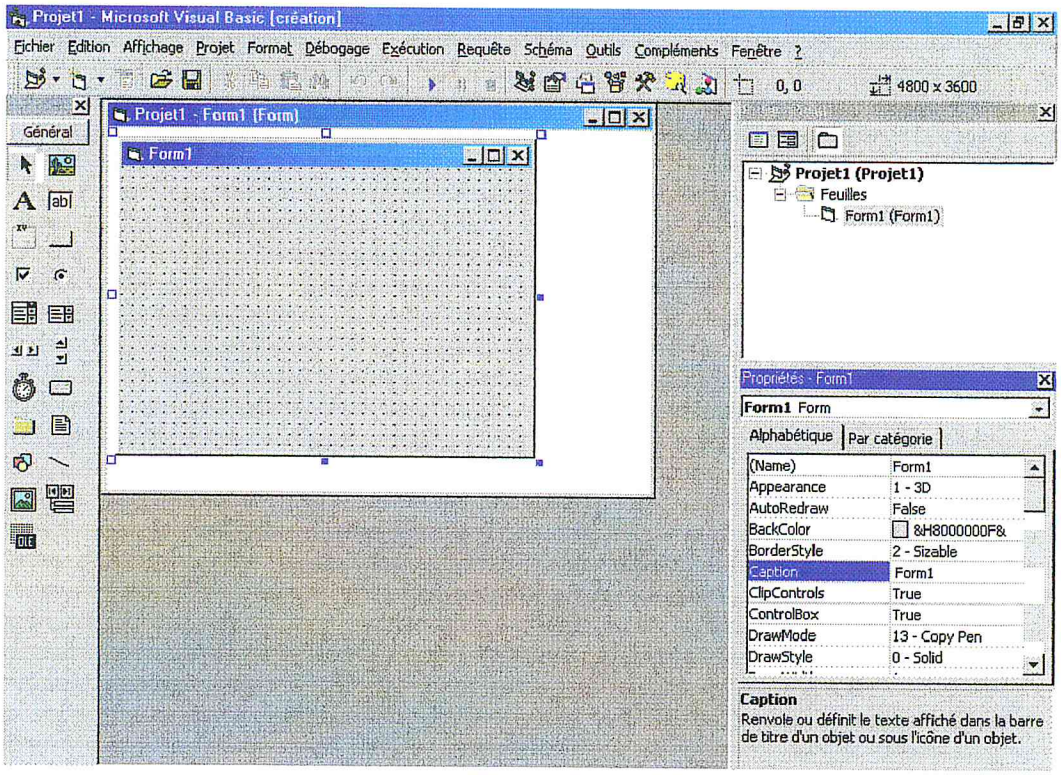


Figure 1.12 : *Interface de Visuel Basic6.*

Ces différents outils d'édition se base sur la technique **WYSIWYG** (what you see is what you get), où l'auteur manipule directement les éléments constituants son document sans passer par du code, il conçoit le document tel qu'il sera visualiser (même dans le cas des outils orientés programmation où la programmation visuelle s'oriente vers cette technique), l'intérêt de cette technique est que l'auteur a une vision préalable du résultat tout en construisant son document, cette technique a été choisie dans le but de satisfaire les besoins d'un environnements d'édition.

1.7.5 **Les différents niveaux d'un éditeur multimédia**

Pour la réalisation d'une application d'édition et de présentation de documents multimédias, on peut présenter une hiérarchie à cinq niveaux, qui correspondent à des traitements de la synchronisation; chaque niveau fournit un ensemble de mécanismes à travers des interfaces adaptées, qui permettent à un niveau donné de fournir des services en s'appuyant sur celle du niveau inférieur [5] (Figure 1.13) :

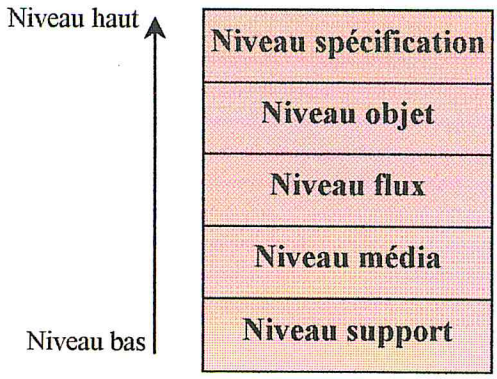


Figure 1.13 : *Niveaux d'un éditeur multimédia.*

- niveau spécification :

  Il s'agit dans ce niveau de décrire les différentes dépendances existantes entre les différents éléments multimédia.

- niveau objet :

  Ce niveau spécifie les traitements permettant la manipulation des différentes dépendances qui existent entre les différents éléments multimédia, en identifiant un ensemble d'opérations de synchronisation (démarrage, arrêt, etc.) qui s'appliquent à tous les médias, ces méthodes sont souvent d'écrites en programmation objet, au moyen de classes.

- niveau flux :

  Il offre des opérations qui s'appliquent sur un ou plusieurs flux de données multimédia, en particulier, toutes les opérations de contrôle et de gestion des contraintes de synchronisation de type intra-élément et synchronisation des lèvres. Les problèmes traités concernent l'allocation des ressources du système pour garantir les contraintes de qualité de service de la présentation.

- niveau média :

  Il concerne les opérations de traitement du média, par exemple, les transformations graphiques des images, le changement de volume de l'audio, les effets visuels appliqués aux séquences vidéo…etc.

- niveau support :

  Il concerne les opérations permettant l'accès au niveau physique du système, comme les opérations relatives à la lecture d'un fichier local contenant un élément multimédia, l'écriture et l'affichage sur écran, les primitives et protocoles d'accès au réseau ...etc.

1.8 Conclusion

Dans ce chapitre nous avons évoqué quelques concepts du multimédia, et certaines approches adoptées par les systèmes d'éditions de documents multimédias, ceux-ci afin de faciliter au mieux, à l'utilisateur, la synchronisation et l'intégration des divers médias constituants son document.

Pour satisfaire le besoin d'intégration et de synchronisation de médias hétérogènes, un choix de standard répondant à cette exigence s'avère nécessaire.

Avec l'évolution technologique, un nouveau langage d'intégration multimédia synchronisé est apparu, appelé SMIL; le prochain chapitre sera donc consacré à l'étude du standard SMIL (Synchronized Multimedia Integration Language).

Chapitre 2

Description du langage SMIL

2.1 Introduction

De nombreux langages ont été proposés pour la spécification des documents multimédias, et notamment pour l'expression des informations temporelles (placement des objets dans le temps, synchronisation entre les objets, etc.). (comme MHEG.

Plus récemment, un nouveau standard de documents multimédia, [13] appelé *SMIL (Synchronized Multimedia Integration Language*), a été proposé au sein du W3C (World Wide Web Consortium). Ce langage, qui est un standard du langage XML (*eXtensible Markup Language*), a été développé par le groupe de travail SYMM du W3C, il propose un format de documents qui s'appuie sur une structure d'arbre d'opérateurs temporels et sur l'utilisation d'un marquage descriptif défini par une DTD (Définition de Type de Document) XML.

Autrement dit, il permet d'écrire la structure de documents principalement textuels à l'aide d'un système de balises, permettant de marquer les éléments qui composent la structure et les relations entre ces éléments.

Actuellement, SMIL existe en deux versions: la première version est SMIL 1.0, la deuxième est SMIL 2.0 qui complète SMIL1.0 en ajoutant d'autres balises.

Dans ce chapitre nous allons décrire la première version du langage SMIL (SMIL1.0) en commençant par la définition du SMIL et la structure générale des documents SMIL,puis la syntaxe ainsi que la sémantique de ce langage.

2.2 Définition et avantages du langage SMIL

Langage d'intégration multimédia synchronisée: SMIL :

Adopté formellement par le W3C en juin 1998, SMIL prononcé (smaïl)est l'acronyme de *Synchronized Multimedia Integration Language*, c'est un langage standardisé, qui reste proche de l'esprit HTML sauf que SMIL permet d'introduire la notion de temps, de découpage en séquences, de transition...etc. "synchronisée" est un mot important qui indique bien que ces éléments (images, sons, vidéos, textes) peuvent être synchronisés les uns avec les autres, pour créer au final de véritables présentations multimédia.

A l'aide de SMIL, un auteur peut :

- décrire le comportement d'une présentation dans le temps ;
- décrire la disposition d'une présentation à l'écran ;
- associer des hyperliens à des objets média.

SMIL est un langage balisé (contient des balises), permettant de représenter et d'échanger des présentations multimédias, intégrant sons, images et textes présentés de façon synchrone. Le langage SMIL a de nombreux avantages, parmi lesquels nous citons [14] :

- la synchronisation est le point fort de SMIL : Les différents médias d'une présentation peuvent être synchronisés de façon discrète ou continue. Autrement dit, l'auteur (l'utilisateur) peut décider sur l'écran quand et où la présentation se déroulera.

- le langage SMIL permet le contrôle du cadre temporel et de la mise en page de présentation multimédia. De cette manière, l'utilisateur peut accorder la lecture de fichiers son et vidéo, de plus il peut ajouter des éléments textuels à un emplacement quelconque du film.

- le placement géométrique, le placement temporel, et la sémantique des inter-acteurs sont définis dans un fichier unique. Le téléchargement de ce fichier suffit à ''jouer'' un titre multimédia arbitrairement complexe.

- les médias sont référencés et pas inclus ce qui permet la disponibilité des éléments medias pour plusieurs présentations.

- SMIL permet de gérer des liens hypermédia en fonction du temps. Ce qui offre plusieurs choix dans une présentation, par exemple on peut afficher des sous-titres ou transmettre une traduction synchrone dans une autre langue.

- un fichier SMIL est lisible aussi bien par l'être humain que par la machine, ce qui facilite considérablement la modification d'une présentation interactive : un simple éditeur de texte suffit.

- parce que le formalisme est normalisé, plusieurs auteurs peuvent coopérer dans la production d'une présentation multimédia, sans pour autant s'obliger à utiliser la même plate-forme de production.

- SMIL ne fait aucune hypothèse sur les formats des composants (images textes, sons, vidéo), tous les composants sont utilisables.

- SMIL a une capacité à séparer les médias ce qui permet d'obtenir un contenu plus léger à charger.

- enfin, la production d'une présentation multimédia interactive avec SMIL, ne demande aucune compétence en programmation. C'est une différence majeure avec les méthodes de scripts fondées sur des langages de programmation tels que JAVA, JAVA SCRIPT, ...etc.

2.3 Structure d'un document SMIL

2.3.1 **Règles de construction**

Le SMIL étant une extension du langage XML, il est structuré comme le langage HTML avec des balises ouvrantes et fermantes :

- les commentaires sont toujours encadrés par < !-- --> ;
- un fichier SMIL commence par <smil> et termine par </smil> ;
- les balises et les attributs sont écrits en minuscule ;
- les extensions des noms des fichiers sont .smi ou .smil ;
- il n'y a pas de balises vides en SMIL.

2.3.2 **Le schéma SMIL**

Le début d'un document SMIL est marqué par la balise <smil>, tandis que la fin est marqué par </smil> ; entre ces deux balises on trouve l'en-tête (*head*) et le corps (*body*). Le schéma SMIL est présenté en générale comme suit :

```
<smil>
 <head>
  <meta>
    … informations générales sur le document.
  </meta>
  <layout>
    …définition de mise en page.
  </layout>
 </head>
 <body>
   …objets et relation temporelles.
   …liens
 </body>
</smil>
```

Figure 2.1: *L'architecture d'un document SMIL*

L'en-tête contient les informations concernant le scénario appelé *meta* tel que l'auteur, le copyright, ...etc. Entre les deux balises <layout>et </layout> on définit l'aspect spatial de la présentation, qui se traduit par la définition de la taille de la surface totale qu'occupera la fenêtre principale qui est appelée *root-layout* et des zones appelées *région* fixées en absolue par rapport à la fenêtre principale, qui contrôle la position, la taille et le rendu (qualité expressive de l'exécution) des médias.

Le corps contient les informations nécessaires sur la localisation et la synchronisation temporelle des objets multimédias participant au déroulement du scénario et les liens nécessaires.

2.4 La syntaxe et la sémantique du langage SMIL

Dans cette partie on va donner plus de précision, concernant l'aspect syntaxique et l'aspect sémantique des éléments cités précédemment (élément smil, l'en-tête et le corps) d'un document SMIL, en citant les éléments appelés *fils* qui sont introduits à l'aide de balises et qui peuvent recevoir en option des attributs.

Donc ,on va donner, dans ce qui suit, une explication sur les balises dites de « structure » les plus rencontrées et les plus importantes du langage SMIL :[13]

2.4.1 **L'élément smil**

Un document SMIL commence par <smil>et fini par </smil>.

Attribut de l'élément :

L'élément smil admet l'attribut suivant : [8]

- Id : «identifiant », cet attribut identifie un élément de manière unique dans un document ;sa valeur est un identifiant XML.

Contenu de l'élément :

L'élément smil peut contenir les éléments fils suivants :

- head : l'en-tête .
- body : le corps.

Exemple d'utilisation de l'élément ***smil***:

```
<smil id =”exemple” >
  <head> […] </head>
  <body> […] </body>
  </smil>
```

2.4.2 L'en-tête du document SMIL (l'élément head)

Il contient des informations qui n'ont pas de rapports avec le comportement temporel de la présentation. Il est limité par les deux balises <head> et </head>.

Attribut de l'élément :

L'élément head admet l'attribut suivant :

- id : identifiant.

Contenu de l'élément :

L'élément head peut contenir les fils suivants :[12]

- layout : mise en forme.
- meta : propriétés de la présentation.
- switch : choix à l'utilisateur.

L'élément head peut contenir plusieurs « meta » et l'un ou l'autre entre un élément « layout » et un élément « switch ».

Exemple d'utilisation de l'élément ***head***:

```
<head id =”code”>
  < meta name =”code” content =”code”/ >
  <layout> […] </layout>
  <switch> […] </switch>
</head>
```

2.4.2.1 **Description de l'élément *layout* :**

L'élément *layout* donne les propriétés d'une présentation, il détermine la manière dont les éléments, dans le corps du document, se positionnent sur une surface de rendu abstraite (soit visuelle, soit acoustique).Autrement dit, l'élément *layout* indique le positionnement géométrique des éléments du corps.

Quand un document n'a pas d'élément layout, le positionnement des éléments de corps dépend de l'implémentation.

Attribut de l'élément :

L'élément layout admet les attributs suivants :

- *id* : identifiant.
- *type* : cet attribut spécifie le langage de mise en forme utilisé dans l'élément layout.

  Pour obtenir la mise en page par défaut pour tous les éléments multimédia, une mise en page de base vide peut être déclarée de la façon suivante :"text/smil-basic-layout".

Contenu de l'élément :

Si l'attribut type de l'élément layout a la valeur "text/smil-basic-layout", il peut contenir en plus les éléments suivants :

- region
- root-layout

Si l'attribut type de l'élément layout a une autre valeur, alors l'élément contient des données textuelles.

2.4.2.1.1 **Description de l'élément *region***

L'élément *region* contrôle la position, la taille et l'échelle des éléments des objets médias.

Attributs de l'élément :

L'élément *region* peut avoir les attributs suivants :

- *background-color* : cet attribut est utilisé pour définir la couleur de l'arrière-plan il est initialisé par défaut à noir.

  Si l'attribut *background-color* est absent alors l'arrière-plan sera transparent.

- *height* et *width* : c'est la hauteur et la largeur du rectangle alloué à la région. Ces valeurs s'expriment en pixels ou en pourcentage.

- *top* et *left* : fixe en absolue, respectivement la coordonnée verticale et horizontale du coin supérieur gauche de « *region* » .
  Ces valeurs s'expriment en pixel ou en pourcentage.

- *title* : c'est un ensemble de caractères qui permettent de donner une description pour *region*.

- *right* et *bottom* : fixe en absolue, respectivement la coordonnée horizontale et verticale du coin inférieur droit de « *region* » .
  Ces valeurs s'expriment en pixel ou en pourcentage.

- *Z-index* : c'est un entier qui donne la profondeur d'affichage. Si par exemple deux éléments A et B ont la même profondeur alors:
  1. Si B débute après A alors B est placé au-dessus de A
  2. Si B et A débute en même temps et si B est après A dans l'ordre lexical alors B est placé au-dessus de A.

- *id* : c'est un ensemble de caractères qui fixent l'identifiant de *region* qui doit être unique.

- *Region-name* : c'est un ensemble de caractères qui assignent un nom à *region*.

- *fit* : Cet attribut spécifie le comportement adopté si la hauteur et la largeur intrinsèques d'un objet média diffèrent des valeurs spécifiées pour les attributs height et width de l'élément region. Il peut prendre comme valeur les attributs suivants :

  1. *fill* : étire l'objet sur la surface de *region* de manière à ce que le contenu vienne juste toucher tous les bords de la boîte. La valeur par défaut de *fill* est "hidden".

2. *hidden* : cette valeur indique que le remplissage commence par le coin supérieur gauche. Si la taille de l'objet est supérieure à celle de *region* alors juste la partie visible sera affichée, sinon le reste sera remplit avec la couleur de fond.
3. *scroll* : un mécanisme de défilement devrait apparaître quand le contenu rendu de l'élément excède ses limites.
4. *slice* : ajuste l'objet média visuel tout en préservant ses proportions d'aspect de manière à ce que sa hauteur, ou sa largeur, soit égale à la valeur spécifiée par l'attribut height, ou width, des parties du contenu pouvant être rognées (découpées). Selon la situation donnée, une partie de l'objet média soit horizontale, soit verticale, sera affichée. Un débordement en largeur est rogné à partir de la droite de l'objet média, un débordement en hauteur est rogné à partir du bas de l'objet média.
5. *meet* : ajuste l'objet média visuel tout en préservant ses proportions d'aspect de manière à ce que sa hauteur, ou sa largeur, soit égale à la valeur spécifiée par l'attribut height, ou width, aucune partie du contenu n'étant rognée .Le coin supérieur gauche de l'objet se fixe sur celui de la boîte et l'espace vide à droite ou en-dessous est rempli par la couleur d'arrière-plan.

Contenu de l'élément :

L'élément *region* est un élément vide.

2.4.2.1.2 **Description de l'élément *root-layout***

Détermine les valeurs de la fenêtre dans laquelle la présentation SMIL est rendue (positionnement des éléments *region*).

Si un document contient plus d'un élément root-layout alors ce document ne devrait pas être affiché car il s'agit d'une erreur.

Attribut de l'élément :

L'élément root-layout peut avoir les attributs suivants :

- *background-color*: ensemble de caractères qui définit la couleur de fond de la fenêtre principale, dans laquelle sont positionnés les éléments *region.*

- *height* : fixe la hauteur de la fenêtre principale. (unité en pixel).
- *id* : ensemble de caractères fixant l'identifiant de root-layout. Il doit être unique.
- *title* : ensemble de caractères donnant un titre pour la fenêtre principale.
- *width* : fixe la largeur de la fenêtre principale. (unité en pixel).

Contenu de l'élément :

L'élément *root-layout* est un élément vide.

Exemple d'utilisation de l'élément ***layout*** :

```
<layout>
  <root-layout background-color="white" height="300" width="400" />
  <region id="video" background-color="gray" top="10"
          left="10" height="200" width="380" z-index="0"/>
</layout>
```

Explication de l'exemple :

Cet exemple a comme résultat l'affichage d'une fenêtre blanche de 300/400pixel, dans laquelle une autre fenêtre grise avec une dimension de 200/380pixel apparaît :

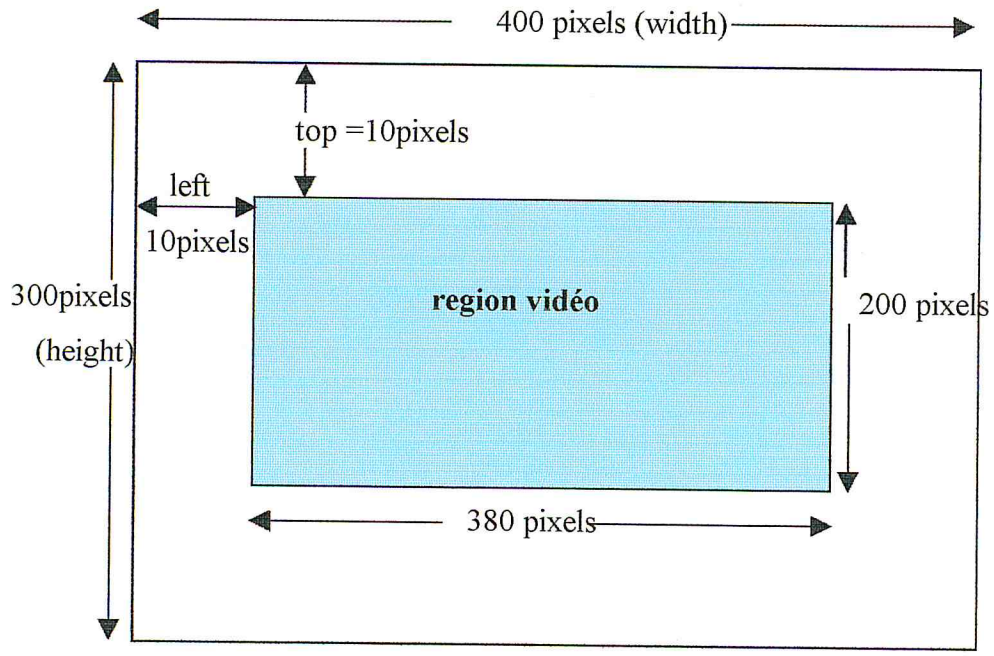


Figure 2.2: *Résultat de l'exemple proposé*

2.4.2.2 **Description de l'élément meta**

Cet élément sert à donner des informations à propos de la présentation SMIL, telles que l'auteur, le titre ou un résumé …etc.

Attribut de l'élément :

L'élément *meta* peut contenir les attributs suivants :

- *content* : ensemble de caractères qui indiquent la valeur de la propriété définie dans l'élément *meta*.

- *name* : ensemble de caractères qui identifient le nom de la propriété.
- *id* : ensemble de caractères qui fixent l'identifiant de *meta*.

Contenu de l'élément :

L'élément *meta* est un élément vide.

Exemple d'utilisation de l'élément ***meta*** :

```
<meta name="author" content="James Turner"/>
<meta name="title" content="Cours sur SMIL"/>
<meta name="copyright" content="(c)2002 JamesTurner"/>
```

2.4.2.3 **Description de l'élément switch**

Cet élément permet de sélectionner des composants et les règles de placement en fonction de certaines préférences de l'utilisateur. Le premier élément multimédia acceptable (c'est à dire que le type de média est reconnu et tous les attributs de test (définis dans la section 4.4) sur l'élément ont pris la valeur "true" lors de son évaluation) est joué.

Si aucun élément n'est acceptable alors aucun fils de l'élément *switch* n'est joué. [8]

Il faut toujours placer les éléments de meilleure qualité en premier, car au plus un fils de l'élément *switch* est joué.

Attributs de l'élément :

L'élément *switch* peut contenir les attributs suivants :

- *id* : ensemble de caractères qui fixent l'identifiant de *switch* .
- *title* : ensemble de caractères qui donnent un titre pour *switch*.

Contenu de l'élément :
Si l'élément *switch* est utilisé dans l'en-tête, il peut contenir l'enfant suivant : [8]

- *layout* : Plusieurs éléments layout peuvent survenir dans l'élément *switch*.

Nous verrons, dans les sections qui viennent (section 2.4.3.3), le contenu de l'élément *switch* quand il est utilisé dans le corps (le *body*).

Exemple d'utilisation de l'élément ***switch***:

```
<switch>
 <meta name =“titre” content =”Entête d’un document SMIL” language =“fr”/>
 < meta name =“title” content =”head of SMIL document” language =“en”/>
</switch>
```

Nous allons terminer cette partie, qui est celle de la syntaxe et la sémantique de l'en-tête, par l'exemple récapitulatif suivant :

```
<smil>
 <head>
  <meta name="author" content="James Turner"/>
  <meta name="title" content="Cours sur SMIL"/>
  <meta name="copyright" content="(c)2002 James Turner"/>
  <layout>
   <root-layout id="principale" height="300" width="400" background-color="white" />
   <region id="r1" title="titre" height="36" width="220" background-color="blue"
           left="98" top="10" z-index="1" fit="hidden" />
   <region id="r2" title="diapositives" height="226" width="308"
           background-color="grey" left="48" top="60" z-index="2" fit="fill" />
   <region id="r3" title="legende" height="43" width="266"
           background-color="blue" left="71" top="235" z-index="3" fit="fill" />
  </layout>
 </head>
 <body>
  .....
 </body>
</smil>
```

Explication de l'exemple :

Ce code SMIL permet d'obtenir une première fenêtre principale d'un fond blanc avec une dimension de 300 pixels de longueur sur 400 pixels de largeur, dans laquelle trois autres régions apparaissent *r1*, *r2*, *r3* :

- *r1* a un fond bleu et une dimension de 36 pixels de longueur sur 220 pixels de largeur.

- *r2* a un fond gris et une dimension de 226 pixels de longueur sur 308 pixels de largeur.

- *r3* a un fond bleu et une dimension de 43 pixels de longueur sur 266 pixels de largeur.

Nous remarquons dans cet exemple que la valeur de l'attribut *z-index* de la région *r3* est supérieure à celle de la région *r2*, qui est à son tour plus grande que celle de la région *r1*, donc si une intersection entre ces régions aura lieu, alors la région *r3* s'affichera au-dessus de la région *r2*, et cette dernière sera affichée au-dessus de la région *r1*, ce qui donne la Figure 2.3 :

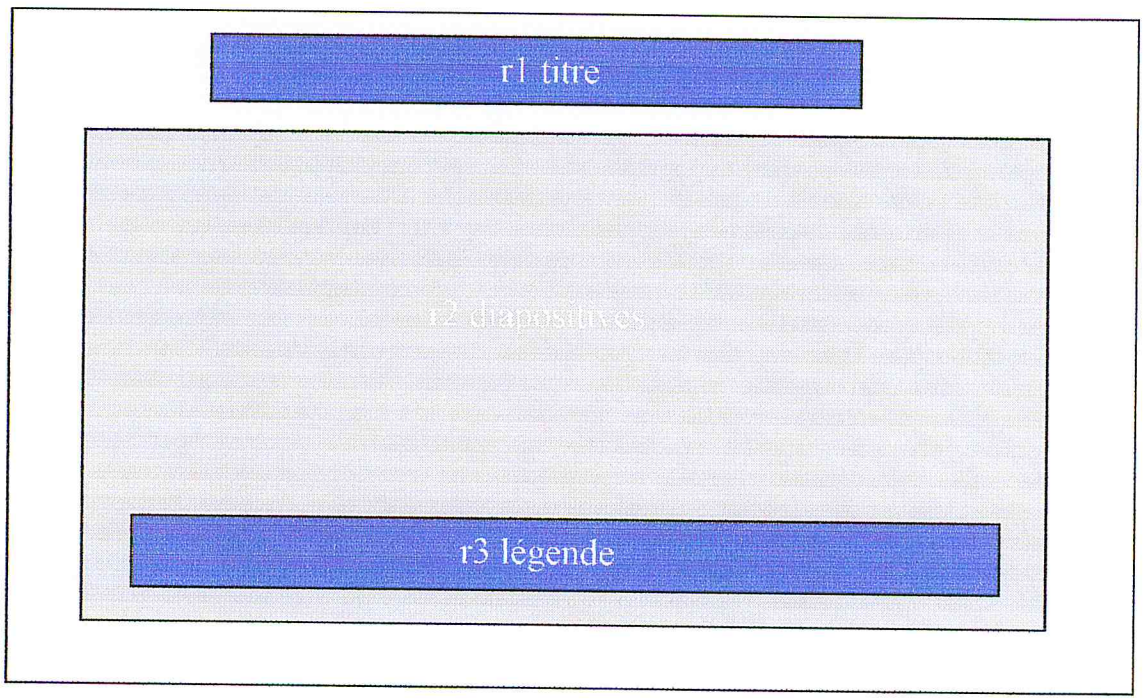


Figure 2.3 : *Résultat de l'exemple*

2.4.3 **Le corps d'un document SMIL (l'élément body)**

Le corps d'un document SMIL est limité par les deux balises <body> et </body> ; il sert à spécifier *l'ordonnancement temporel* de l'ensemble des composants, sonores, visuels ou mixtes, et le comportement des liens du document. Par exemple on peut spécifier le début et la fin d'une vidéo, le moment et la durée de l'apparition d'une image.

Attributs de l'élément :

L'élément body admet l'attribut suivant :

- *id* : identifiant.

Contenu de l'élément :

L'élément body à pour fils :

- *Les éléments de synchronisation* :
  - *par* : permet le déroulement (ou l'affichage) de plusieurs médias en parallèle (au même temps)
  - *seq* : permet le déroulement (ou l'affichage) séquentiel de plusieurs médias.
- *Les éléments des objets média* :
  - *ref* : une référence sur un objet média.
  - *animation* : un vecteur d'animation graphique.
  - *vidéo* : un clip vidéo au format *mpeg* ou autre.
  - *audio* : une séquence audio au format *mp3, wave* ou autres.
  - *img* : une image au format *jpeg, jif* ou *png*.
  - *text* : une référence textuelle.
  - *textstream* : un texte à charger.
- *Les éléments d'hyperlien*
  - *a, anchor* : définissent un lien de navigation.
- *switch* : il spécifie un groupe d'éléments alternatifs. Un élément est choisi et exécuté selon l'évaluation de l'attribut de test.

2.4.3.1 **Les éléments de synchronisation**

2.4.3.1.1 **Description de l'élément *par***

L'élément *par* permet la lecture parallèle d'un ensemble d'éléments média. Autrement dit, *par* définit des groupes de composants présentés simultanément. L'ordre textuel d'apparition des enfants dans un élément *par* n'a pas de signification pour le minutage de leur présentation [9].

Attributs de l'élément :

L'élément *par* peut avoir les attributs suivants :

- *id*: identifiant.
- *abstract* : chaîne de caractères qui donne une courte description du contenu de l'élément.

- *author* : chaîne de caractères qui donnent le nom de l'auteur du contenu de l'élément multimédia.

- *copyright* : chaîne de caractère qui donne la notice des droits d'auteur.

- *dur* : cet attribut spécifie la durée explicite d'un élément, il prend une valeur *horloge* ou la valeur *indefinite* (pour une durée indéfinie).
  La *durée implicite* d'un élément de synchronisation correspond à la différence entre la fin implicite et le début implicite.
  Dans le langage SMIL un élément peut avoir une *durée explicite* en ajoutant l'attribut *dur* : dur = " *valeur de durée explicite*".

- *begin* : Cet attribut spécifie l'instant du début explicite d'un élément de synchronisation.
  Chaque élément dans le langage SMIL a un *début implicite* ; comme il peut avoir un début explicite si on ajoute l'attribut begin : begin ="*valeur de début explicite*".
  L'attribut *begin* peut avoir les valeurs suivantes :
  - *valeur de délai* : une valeur d'horloge mesurant l'instant de présentation.

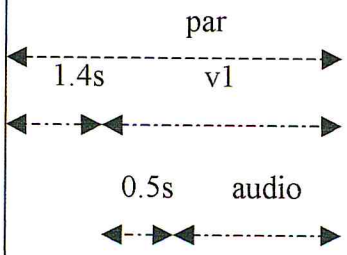
Exemples :

```
<par>
  <audio id="a" begin="6s" src="audio" />
 </par>
```

Figure 2.4 : *Utilisation de* ***la valeur de délai*** *dans un élément* ***par***

- *valeur d'événement* : le début de l'élément dépend d'un événement donné qui surviendra.

```
<par>
 <video id="v1" src="cnn.mpv" region="V-main"begin="1.4s" />
 <audio src="cnn.wav" region="music" begin="id (v1) (0.5s)" />
</par>
```



L'audio est retardée de 0,5 seconde par rapport au début de l'élément vidéo "v1"

Figure 2.5 : *Utilisation de* ***la valeur d'événement*** *dans un élément* ***par***

- *end* : cet attribut spécifie la fin d'un élément de synchronisation.
  Il admet les mêmes types de valeur que l'attribut begin.
  Chaque élément de synchronisation a une fin implicite, comme il peut avoir une fin explicite si l'attribut *end* figure dans le code SMIL : end = "*valeur de fin explicite*".

- *endsync* : cet attribut contrôle le comportement temporel implicite de l'élément *par*, il admet les valeurs suivantes : first ,last ,id-ref.

- Si *endsync* est égale la valeur ***first*** : la durée implicite de l'élément *par* est le minimum entre les durées voulues de ses enfants.
- Si *endsync* est absent ou égale à la valeur ***last*** : la durée implicite de l'élément *par* correspond au maximum entre les durées de ses enfants
- Si *endsync* est égale à la valeur ***id-ref*** : la durée implicite de l'élément *par* est celle de l'enfant appelé par cette valeur d'attribut *id-ref*. Cette dernière a la syntaxe suivante : id-ref ::= "id(" valeur-id ")" où "valeur-id" doit être un identifiant XML légal.

La valeur par défaut de l'attribut *endsync* est *last*.

- *region* : spécifie l'identifiant de la surface d'affichage attribuée à l'objet. Cet attribut a comme valeur l'identifiant de région.

- *repeat* : permet de répéter le déroulement d'un objet .La valeur de l'attribut peut être un entier ou la chaîne ***indefinite***. La valeur par défaut est "1".

- *Attributs de test* : ce sont des attributs qui prennent des valeurs booléennes (vrai, faux) et qui sont : (expliquer dans la section 2.4.4)
  - *system-bitrate*: contrôle le taux de transmission.
  - *system-captions*:ignoré si l'utilisateur ne veut pas de sous titre.
  - *system-language* : adaptation de la langue.
  - *system-overdub-or-caption* : spécifie la forme de légende ou de bulle pour afficher les commentaires.
  - *system-required*:reconnaissance ou non d'une extension.
  - *system-screen-size*:spécifie la taille minimale de l'écran.
  - *system-screen-depth* : spécifie la résolution minimale de l'écran.
- title : titre.

Contenu de l'élément :

L'élément *par* peut contenir les fils suivants :

- *a* : un élément d'hyperlien, il définit un lien de navigation.

- *animation* : un vecteur d'animation graphique
- *audio* : une séquence audio au format *mp3, wave* ou autre.
- *img* : une image au format *jpeg, jif* ou *png*.
- *par*.
- *ref* : une référence sur un objet média.
- *seq* : (défini dans la section 2.4.3.1.2.)
- *switch* : (défini dans la section 2.4.3.3.)
- *text* : une référence textuelle.
- *textstream* : un texte à charger.
- *video* : un clip vidéo au format *mpeg* ou autre.

Tous ces éléments peuvent se présenter plusieurs fois comme un fils direct de l'élément *par*.

Exemple :

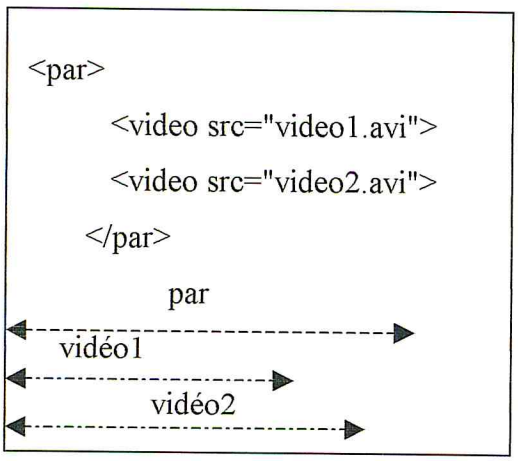


Figure 2.6 : *Utilisation de l'élément* ***par***

2.4.3.1.2 **Description de l'élément *seq***

L'élément *seq* regroupe les éléments qui se dérouleront de façon séquentielle .Les enfants sont joués les uns après les autres dans l'ordre où ils apparaissent dans le fichier (fichier SMIL).

Attributs de l'élément :

Les attributs de l'élément *seq* sont les mêmes que ceux de l'élément *par*.

Contenu de l'élément :

L'élément *seq* peut contenir les mêmes éléments qui apparaissent dans le contenu de l'élément *par*.

Exemple :

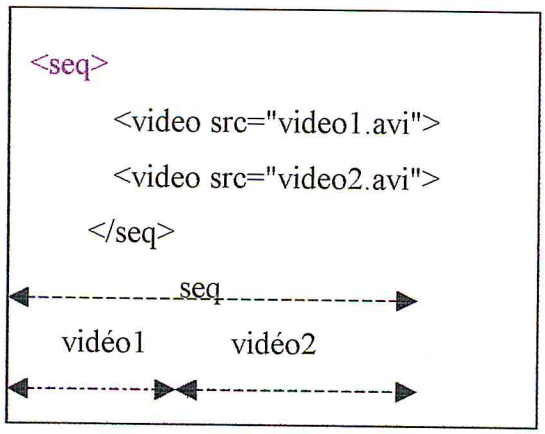


Figure 2.7 : *Utilisation de l'élément* ***seq***

Les deux éléments *par* et *seq* peuvent être imbriqués. Autrement dit il est possible d'imbriquer les deux modes de lecture (séquentielle et parallèle).

Exemple :

L'exemple ci-dessous se déroule en deux phases (l'une après l'autre) :

- *1ère phase* : Deux vidéos (video1.avi et video2.avi) seront lues en même temps
- 2ème phase : La vidéo (video3.avi) sera jouée

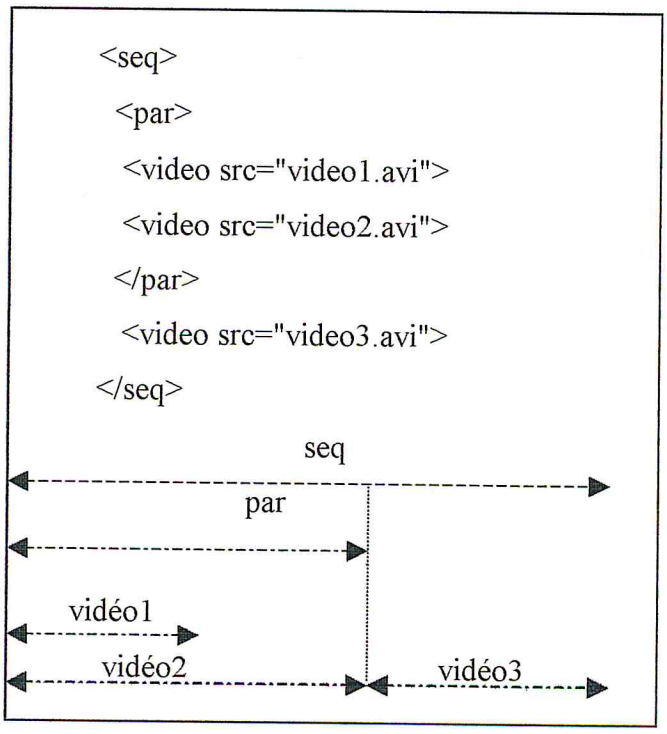


Figure 2.8 : *Exemple d'imbrication des éléments* ***par*** *et* ***seq***.

Notons qu'il existe plusieurs façons d'imbrication de ces deux éléments.

2.4.3.1.3 **Description des éléments des objets média**

Les élément des objets média sont les suivants : *ref, animation, audio, img, video, text* et *textstream*. Ces éléments permettent l'inclusion des objets média dans un document SMIL en les référençant (en utilisant un URI (Uniform Resource Identifiers)).

On distingue deux classes d'objet média :

- Ceux avec une durée intrinsèque désignés par *Médias continus*, par exemple l'audio et la vidéo.
- Ceux sans une durée intrinsèque désignés par *Médias discrets*, par exemple le texte et l'image statique.

Attributs des éléments :

Les éléments des objets média peuvent avoir les attributs suivants :

- *alt* : une chaîne de caractères qui forment un texte de remplacement pour les médias qui ne peuvent pas être visualisés par agent utilisateur.
- *id* : identifiant interne unique.
- *title* : indique le titre du média.
- *src* : donne l'emplacement de l'objet média .La valeur de l'attribut src est l'URI de l'objet média.
- *type* : indique le type MIME de l'objet média appelé par l'attribut src.
- *fill* : utilisé pour la persistance d'un objet multimédia sur l'écran, il peut prendre les deux valeurs suivantes :
  - *freeze* : n'a de sens que pour les composants vidéo ou animation : dans ce cas, lorsque la durée prévue de présentation est atteinte, le composant est figé sur son dernier frame (garder la dernière image sur l'écran après la terminaison de l'objet multimédia).
  - *remove* : le composant disparaît lorsque sa durée prévue de présentation est atteinte.
- *longdesc* : spécifie un lien (URI) vers la description détaillée d'un objet média, qui devrait compléter la description courte offerte par l'attribut alt.
- *clip-begin* et *clip-end* : ce sont des délais utilisés pour le découpage temporel des médias continus. Ils indiquent un décalage depuis l'instant de début ou de fin d'un projet.

Et les éléments définis dans la section (2.4.3.1.1) :

- *abstract.*
- *author*
- *Repeat*
- *begin.*
- *copyright.*
- *dur.*
- *end.*
- *region .*
- *les attributs de test.*

Contenu des éléments :

Les éléments des objets média peuvent contenir l'élément suivant :

- *anchor* : L'élément *anchor* autorise la spécification de sous-parties spatiales ou temporelles d'un élément multimédia. (détaillé dans la section 2.4.3.2.2).

2.4.3.2 **Les éléments d'hyperlien**

Il existe deux éléments d'hyperlien dans le langage SMIL :

2.4.3.2.1 **Description de l'élément *a***

L'élément *a* permet de créer un lien vers une source spécifiée dans son attribut *href.*

Attribut de l'élément :

L'élément *a* peut avoir les attributs suivants :

- *id* : identifiant.
- *title* : titre pour le lien.
- *href* : contient l'adresse URI du lien.
- *show* : permet de spécifier la gestion de l'état de la présentation quand un lien est suivi, cet attribut peut prendre les valeurs suivantes :
  - *replace* : la présentation courante sera interrompue et remplacée par la ressource de destination.
  - new : la présentation de la ressource de destination commence dans un nouveau contexte sans affecter la présentation source.

- pause : la présentation courante sera figée, elle est reprise dés la fin de l'affichage de la ressource de destination.

Contenu de l'élément :

L'élément a peut contenir les enfants suivants :

- *par*.
- *seq* .
- *les éléments des objets média.*
- *switch* : (défini dans la section 2.4.3.3).

Exemple1:

Une nouvelle présentation en plus de celle en cours de lecture, est lancée par le lien.

```
<a href="http://www.cwi.nl/somewhereelse.smi" show="new">
 <video src="file://D:/graph.imf" region= "fenêtre"/>
</a>
```

Exemple2:

La présentation en cours de lecture est mise en pause et la nouvelle est lancée par le lien.

```
<a href="http://www.cwi.nl/somewhereelse.smi" show="pause">
  <video src="file://D:/graph.imf" region= "fenêtre"/>
 </a>
```

2.4.3.2.2 **Description de l'élément *anchor***

L'élément *anchor* permet d'associer un lien avec un sous-ensemble spatial ou temporel de l'objet média. Autrement dit, il utilise une partie ou un extrait d'un objet multimédia comme source dans un lien.

Attribut de l'élément :

L'élément *anchor* admet les mêmes attributs de *a* plus les suivants :

- *begin*.
- *end*.

C'est deux éléments sont définis dans la section (2.4.3.1.1).

- *coords* : cet attribut exprimé en pixels ou en pourcentage, définit le cadre de la surface visible du média qui représente la surface active pour le lien. L'attribut *coords* contient quatre valeurs : les deux premiers (left-x, top-y) représentent les coordonnées X et Y du coin supérieur gauche et les deux autres (right-x, bottom-y) celles du coin inférieur droit.

Voici un schéma explicatif pour mieux comprendre la signification de l'attribut *coords* :

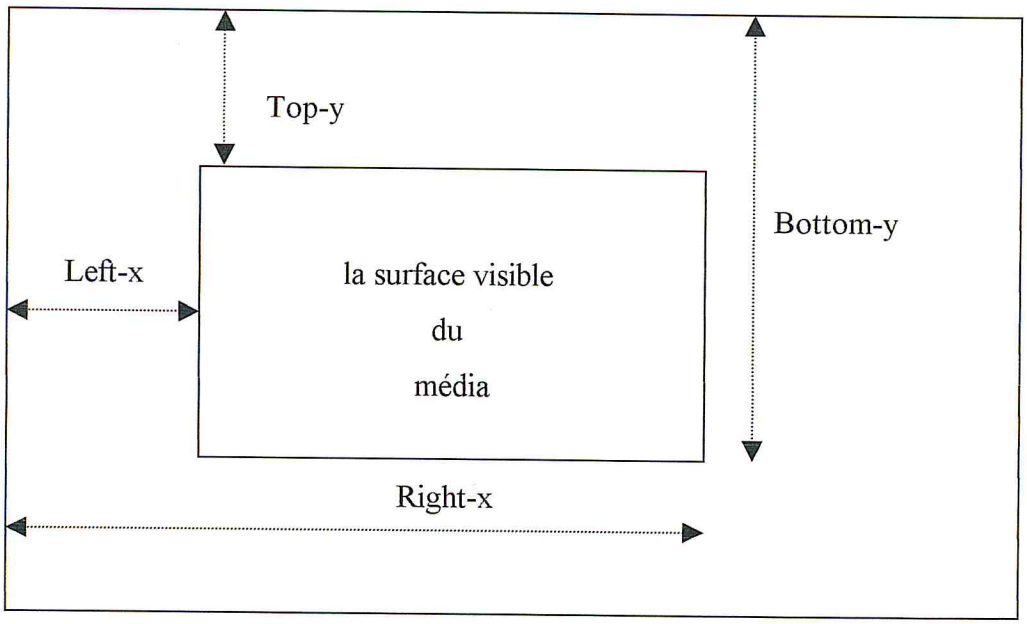


Figure 2.9: *Sémantique de l'attribut* ***coords***.

Exemple1 :

Association de liens aux sous-ensembles spatiaux

Dans l'exemple suivant, la place sur l'écran prise par un clip vidéo est partagée en deux sections. Un lien différent est associé à chacune d'elles :

```
<video src="http://www.w3.org/ChouetteTruc">
 <anchor href="http://www.w3.org/AudioVideo" coords="0%,0%,50%,50%"/>
 <anchor href="http://www.w3.org/Style" coords="50%,50%,100%,100%"/>
</video>
```

Exemple2 :

Association de liens aux sous-ensembles temporels

Dans l'exemple suivant, la durée d'un clip vidéo est partagée en deux sous-intervalles. Un lien différent est associé à chacun d'eux :

```
<video src="http://www.w3.org/ChouetteTruc">
 <anchor href="http://www.w3.org/AudioVideo" begin="0s" end="5s"/>
 <anchor href="http://www.w3.org/Style"      begin="5s" end="10s"/>
</video>
```

2.4.3.3 **L'élément switch**

Il a le même rôle que celui d'un switch placé dans l'en-tête (voir la section 2.4.2.3) avec les mêmes attributs, mais il contient des éléments différents.

Contenu de l'élément :

L'élément *switch* dans le *body* peut contenir les éléments suivants :

- *les éléments de synchronisation* (***par, seq, les éléments des objets média***).
- *Les éléments d'hyperlien* (***a***).
- *switch*.

2.4.3.4 **Expression des durées sur l'horloge SMIL**

Dans le langage SMIL, il existe deux syntaxes qui permettent d'exprimer des durées d'horloge. La première utilise des abréviations pour les heures, minutes, secondes et millisecondes :

1.5h = 1 heures 30 minutes.

2.75min = 2 minutes 45 secondes.

10.55s = 10 secondes 550 millisecondes.

100ms = 100 millisecondes.

La seconde syntaxe, appelée « temps de représentation normal » (normal play time), est de la forme :

" npt = hh :mm :ss .xyz"

dans laquelle **hh** est le nombre d'heures, **mm** le nombre de minutes, **ss** les secondes, **x** des dixièmes de seconde, **y** des centièmes de seconde et **z** des millièmes de seconde. L'indication des heures est optionnelle :

"npt = 00 :04 :12" = "npt = 04 :12" = 4 minutes 12 secondes.

"npt = 00 :05.13" = 5 secondes et 130 millisecondes.

Nous pouvons donner la valeur d'un attribut de placement temporel en utilisant l'une ou l'autre de ces syntaxes. Les deux éléments suivants sont parfaitement équivalents :

<audio src="g.mp3 " begin="0.175 min" />

<audio src="g.mp3 " begin="npt =00 :10.5 " /> .

2.4.4 Les attributs de test

C'est un ensemble d'attributs qui ont des valeurs booléennes (true, false), qui peuvent accompagner tout élément de synchronisation et qui évaluent les capacités et les paramétrages des systèmes.

Ils permettent d'effectuer un test sur un ensemble d'objet média placé souvent dans un bloc *switch*. Ces attributs sont les suivants : [13]

- *system-bitrate*: Cet attribut spécifie, en bits par seconde, la bande passante de la connexion réseau de l'utilisateur, il est évalué à la valeur *true* si cette bande est supérieure ou égale à la valeur spécifiée.

- *system-captions* : Cet attribut permet à l'auteur de fournir des sous-titres. il peut prendre les deux valeurs suivantes :
  - **On:** si l'auteur désire avoir un sous-titre et dans ce cas l'attribut *system-caption* s'évalué à *true*.
  - **Off :** si l'auteur ne veut pas de sous-titre et dans ce cas l'attribut *system-caption* s'évalué à *false*.

- *system-language*: Cet attribut permet de tester la langue configurée par l'utilisateur, par exemple 'fr' pour spécifier la langue française.

- *system-overdub-or-caption*: Cet attribut spécifie la forme de légende ou de bulle pour afficher les commentaires. il peut prendre les deux valeurs *overdub* ou *caption*.

- *system-required*: Cet attribut spécifie le nom d'une extension. Il s'évalué à "true" si l'extension est reconnue par l'implémentation, sinon s'évalue à "false". L'attribut prend comme valeur le nom de l'extension.

- *system-screen-size*: Cet attribut permet de tester si l'écran est capable d'afficher la présentation avec sa taille spécifiée par l'auteur.
  Les valeurs de l'attribut *system-screen-size* ont la syntaxe suivante :
  Valeur-taille-écran ::= hauteur-écran"X"*largeur-écran* (en pixels).

- *system-screen-depth*: Cet attribut spécifie la profondeur, en bits, de la palette de couleurs de l'écran nécessaire pour l'affichage d'une vidéo ou une image.
  Sa valeur doit être supérieur à 0, il peut prendre les valeurs : 1, 4, 8, 16, 24 ou 32.

Exemple:

```
<par>
 <video src = "video.avi /">
 <audio src =="music.wav">
 <switch>
 <text src=="français.rt" system-language=="fr">
 <text src=="anglais.rt" system-language=="en">
 <text src=="allemand.rt" system-language=="de">
 </switch>
</par>
```

Dans cet exemple, les trois médias (vidéo, son et texte) seront exécutés en parallèle grâce à l'élément *par*. Le texte sera lu en fonction de la configuration de l'utilisateur. Si aucune langue ne correspond à cette configuration, alors le texte ne sera pas affiché.

2.5 La DTD SMIL

Un document SMIL 1.0 peut contenir en option une déclaration de type de document qui nomme la définition de type de document (DTD) qui est rédigée dans une syntaxe formelle qui explique avec précision quels éléments et entités peuvent apparaître dans le document et quels sont les contenu des éléments et des attributs.

Autrement dit, la DTD autorise à placer des contraintes sur la forme qu'un document SMIL peut prendre ;elle peut être déclarée formellement dans le corps du document SMIL ou dans un fichier à part.[15]

2.6 Les éditeurs SMIL

Les recherches dans le domaine de l'édition des documents multimédias sous le format SMIL sont récentes, c'est la raison pour laquelle il existe peu d'outils d'édition qui génèrent ce type de documents. Parmi ces éditeurs nous citons ce qui suit :

2.6.1 **SMIL Wizard**

L'objectif de cet éditeur, qui est basé sur des maquettes prédéfinies, est de générer très facilement des documents multimédias en respectant le standard SMIL 1.0 [13].(voir Figure 2.10)

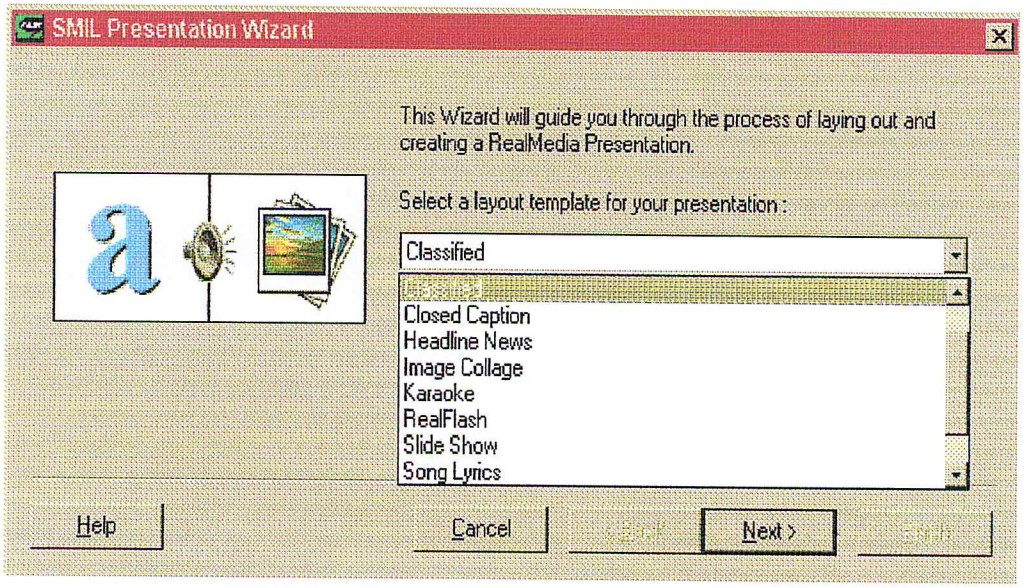


Figure 2.10 : *SMIL Wizard choix d'un modèle de scénario*

Le principe de cet outil est de proposer des maquettes de documents à l'utilisateur, par exemple, après avoir choisie une présentation de type «*Image Collage*» (voir Figure 2.10), qui consiste à afficher quatre images simultanément avec un commentaire audio, l'utilisateur n'a plus qu'à indiquer au système les médias à mettre en place dans le scénario, ensuite l'éditeur génère un document au format SMIL.

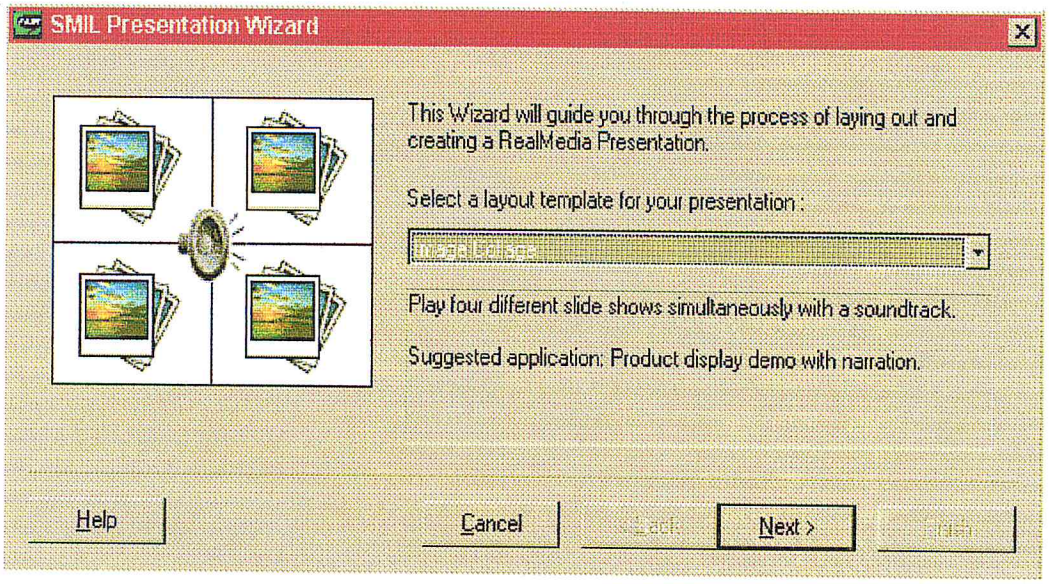


Figure 2.11 : SMIL Wizard affectation des différents médias

L'utilisateur peut visualiser le fichier source de son document et exécute celui-ci. L'avantage de cet outil est sa simplicité et l'impossibilité d'écrire des scénarios temporels incohérents.

SMIL Wizard est destiné aux utilisateurs novices, son usage est facile et bien qu'il est possible de sauvegarder des documents au format SMIL 1.0, l'expressivité des document reste très pauvre. En effet, la synchronisation du scénario temporel est liée aux maquettes disponibles, et n'exploite pas toutes les possibilités offertes par SMIL.En plus, l'auteur ne peut pas modifier son document ni de point de vue spatial ni de point de vue temporel, il est obligé d'utiliser les schémas d'exécution proposés, avec le risque de ne pas en trouver des satisfaisants.

2.6.2 GRiNS

L'objectif de cet outil, qui est actuellement commercialisé par la société Oratrix [11], est l'édition de documents SMIL; son principe est d'associer à un groupe d'éléments un opérateur (mise en parallèle ou en séquentielle)qui va synchroniser, d'une façon implicite, tous les éléments du groupe.

Au niveau de l'interface de GRiNS, dans la vue structurelle, les éléments d'un groupe parallèle sont affichés sur des lignes différentes (une ligne pour chacun), alors que dans le cas d'un groupe séquentiel, chaque élément est affiché dans une colonne, un tel affichage

permet de donner, en plus de l'information hiérarchique, une information temporelle qui permet une certaine compréhension de l'enchaînement des objets dans le temps.

Pour effectuer la synchronisation des objets, GRiNS propose une vue *timeline* dans laquelle les opérations d'ajout, de suppression ou de modification sont possibles.

Malheureusement, la manipulation de cette vue n'est pas directe, elle passe par des boites de dialogues pour modifier les valeurs des objets.

De plus, elle ne reprend pas la notion de structure, il est donc difficile pour l'auteur de faire un lien entre les deux vues (temporelle et structurelle). Notons qu'il a été remarqué que l'interface de la vue temporelle ne donne pas toujours une représentation pertinente de la taille des objets.

2.6.3 SMIL Composer

SMIL Composer de Sausage [16] est un outil qui s'appuie sur la notion de structure de document, il est destiné à faire l'édition de documents SMIL; sa vue structurelle permet l'ajout de groupes parallèles ou séquentiels, puis des médias à l'intérieur, cette vue ne rend aucune dimension temporelle, elle s'apparente plus à une vue textuelle que graphique (voir Figure 2.13).

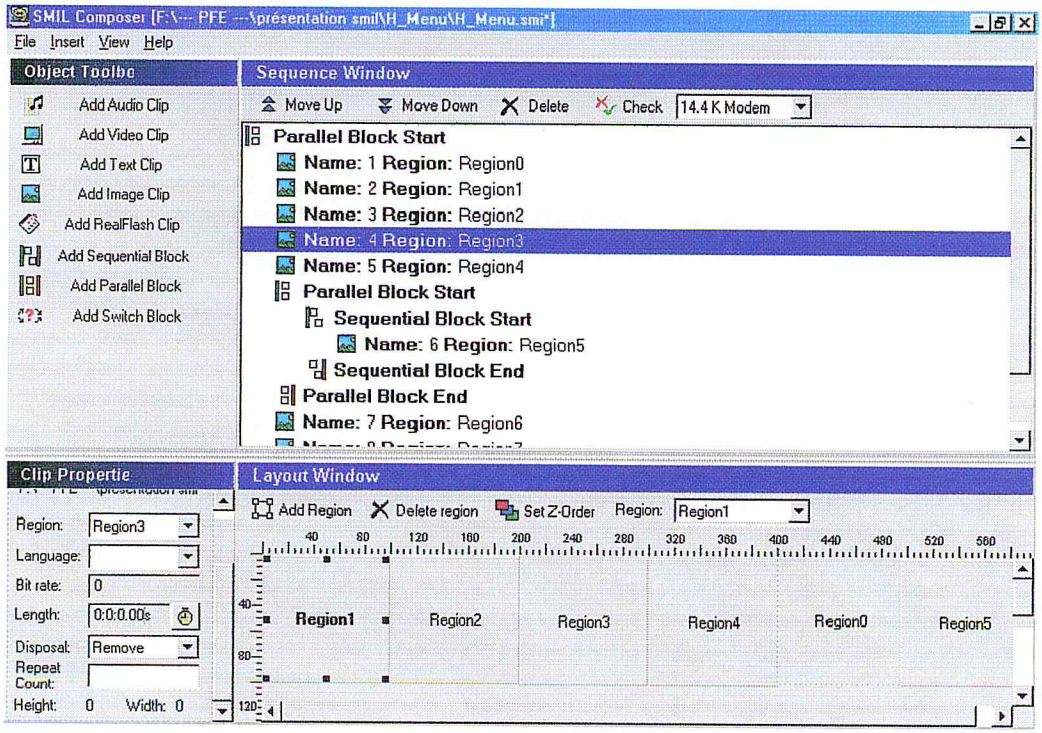


Figure 2.13 : *Vue structurelle et vue spatial de SMIL Composer.*

Aucune autre vue ne vient aider l'auteur dans la perception temporelle, donc il est difficile d'avoir une idée du déroulement temporel du scénario. De plus cet éditeur limite le pouvoir d'expression du langage SMIL, en n'autorisant pas de description de synchronisation fine entre les objets.

SMIL Composer a une interface graphique simple et relativement pratique à l'utilisateur, il propose une synchronisation basée sur la structure des scénarios.

Notons que dans les éditeurs SMIL cités auparavant, la lecture du fichier résultat est externe.

2.7 Les lecteurs (les players)

SMIL n'est pas lisible directement par les navigateurs, il est nécessaire d'avoir un module externe pour lire les fichiers correctement.

A titre d'exemple, pour le navigateur *Internet Explorer de Microsoft*, à partir de la version *5.5* qu'il peut lire ces fichiers.

Il existe plusieurs lecteurs SMIL appelés encore **players** SMIL, même s'ils sont encore rares.

Voici la liste des principaux lecteurs actuellement utiles pour afficher du SMIL:

- *RealPlayer*, de RealNetworks
  URL: http://www.real.com/products/player/index.html
- *RealOne Player*, de RealNetworks.
  URL: http://www.realnetworks.com/solutions/leadership/realone.html
- *Apple QuickTime 5*
  URL: http://www.apple.com/quicktime/download/
- *GRiNS* de Oratrix qui est un lecteur et éditeur SMIL
  URL : http://www.oratix.com/GriNs/

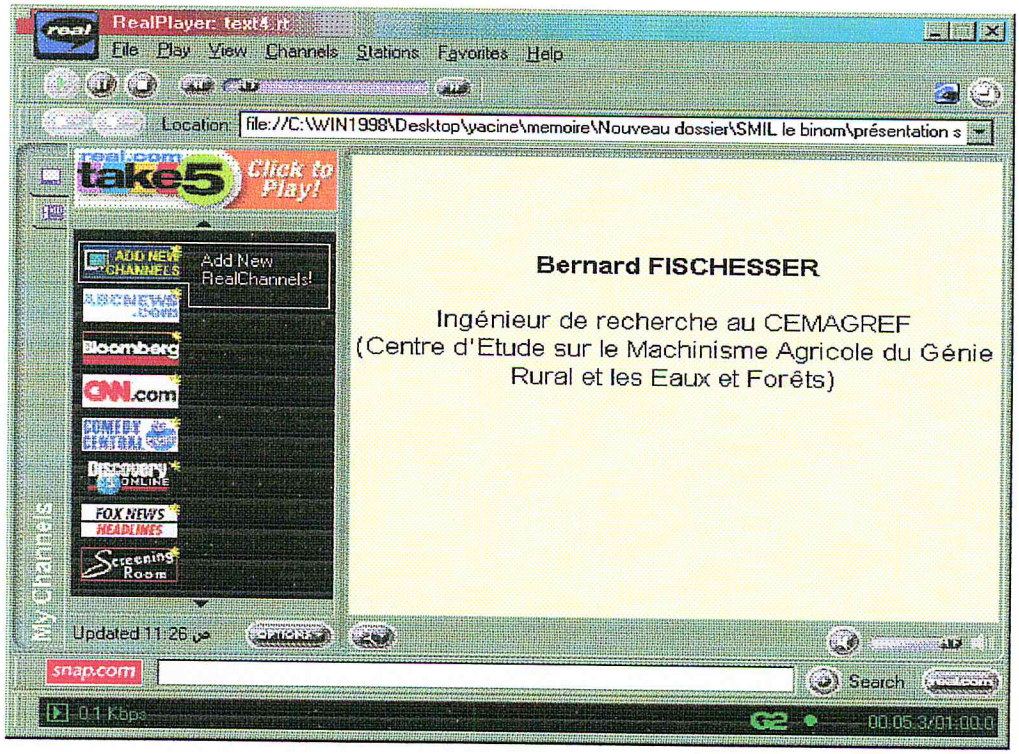


Figure 2.13: *Lecture d'un document SMIL par le lecteur* ***Real Player***.

2.8 Exemples récapitulatifs

Exemple 1

Pour mieux comprendre les informations données dans ce chapitre voici un code SMIL complet avec des commentaires mentionnés en **gras**.

```
<smil>
 <head>
Propriétés du nouveau document SMIL
  <meta name="author" content="allhtml"/>
  <meta name="title" content="exemple de code"/>
  <meta name="copyright" content="© 2001"/>
Mise en page
  <layout>
Définition de la couleur d'arrière-plan, et de la taille de la présentation
  <root-layout background-color="#0000FF" height="160" width="200"/>
Création de zones où seront affichés les éléments. Ici, on a une première zone de la
taille de la présentation, les autres régions sont disposées comme suivant :
-1 : zone_image1
-2 : zone_video2
-3 : zone_image2
-4 : zone_video2
  <region id="zone_accueil" left="0" top="0" height=" 160" width="200"/>
  <region id="zone_image1" left="0" top="0" height=" 40" width="50"/>
  <region id="zone_video1" left="50" top="0" height=" 40" width="50"/>
  <region id="zone_image2" left="50" top="40" height=" 40" width="50"/>
  <region id="zone_video2" left="0" top="40" height=" 40" width="50"/>
  </layout>
 </head>
 <body>
Nouvelle séquence : L'image d'accueil s'affichera dans un premier temps puis dans un
deuxième temps une lecture parallèle se mettra en place
 <seq>
<img id="accueil" src="debut.rp" region="zone_accueil"/>
```

Nouvelle lecture parallèle à l'intérieur de la séquence : Les images, les videos et la voix-off s'exécuteront en même temps

```
<par>
```

L'image1 et 2 auront des liens sur des nouvelles présentations cible1 et cible2

```
<a src="cible1.smi" show="new"/>
<img id="image1" src="bleu.rp" region="zone_image1"/>
</a>
<a src="cible2.smi" show="new"/>
<img id="image2" src="rouge.rp" region="zone_image2"/>
</a>
```

La vidéo1 sera stoppée au bout de 15 secondes, mais restera affichée

```
<video id="video1" src="jaune.rm" region="zone_video1" clip-end="15s"/>
```

La vidéo2 commencera au bout de 5 secondes

```
<video id="image1" src="vert.rm" region="zone_video2" clip-begin="5s"/>
```

La voix-off varie selon la langue du système de l'utilisateur

```
<switch>
<audio src="sonfrancais.rm" system-language="fr" repeat="2"/>
<audio src="sonanglais.rm" system-language="en" repeat="2"/>
<audio src="sonallemand.rm" system-language="de" repeat="2"/>
```

Si le système utilise une autre langue, la voix-off par défaut sera *sonanglais.rm*

```
<audio src="sonanglais.rm" repeat="2"/>
```

Fermeture (dans l'ordre) de toutes les balises

```
</switch>
</par>
</seq>
</body>
</smil>
```

Exemple 2

```
<smil>
 <head>
  <layout>
  <root-layout background-color="white" height="600" width="600"/>
  <region id="r1" left="150" top="0" height="30" width="300" z-index="1" />
  <region id="r2" left="0" top="0" height="600" width="600"
                        z-index="0" fit="fill"/>
  </layout>
 </head>
 <body>
  <par>
    <text src="\Titre.rt" region="r1" dur="30"/>
    <img src="\Firewks3.jpg" region ="r2" dur="30"/>
  </par>
 </body>
</smil>
```

Le résultat de cet exemple (exemple 2) est représenté dans la figure suivante (Figure 2.11)

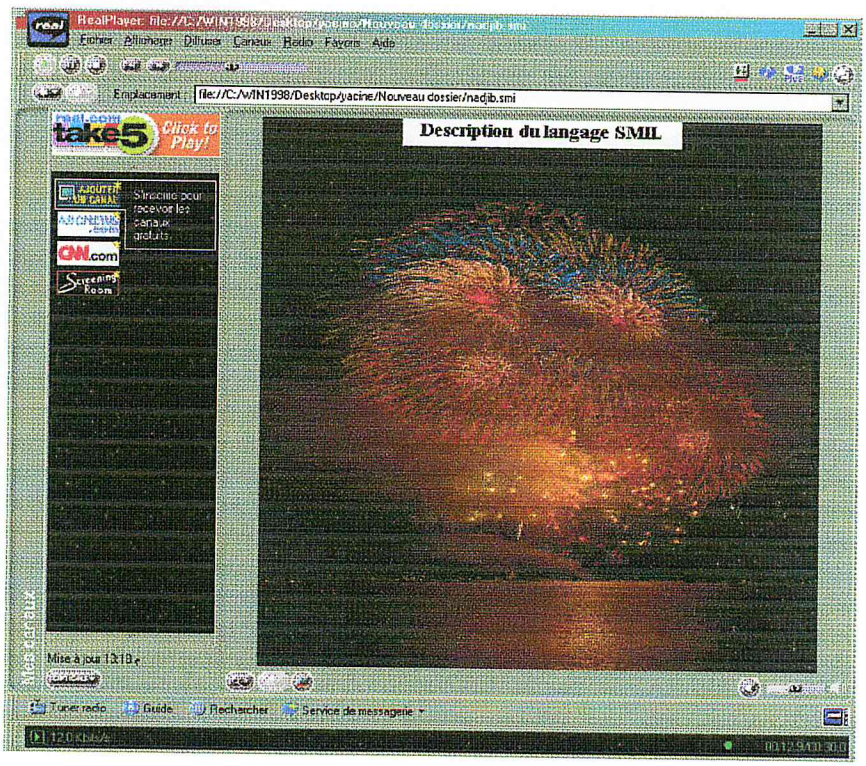


Figure 2.14: *Résultat de l'exemple 2*

2.9 Conclusion

Nous avons présenté dans ce chapitre la description du langage SMIL dans sa première version qui se résume en la composition de son document en aspect spatial et temporel.

Concernant l'aspect spatial, qui est défini dans l'en-tête, nous pouvons dire qu'il englobe toutes les déclarations de l'emplacement des *region* ainsi que leurs dimensions dans une fenêtre principale *root-layout.*

Alors que l'aspect temporel, qui est défini dans le corps, sert à définire la synchronisation entre les médias en utilisant les éléments ***par*** et ***seq*** et à l'affiner grâce aux éléments ***begin***, ***end***, ***dur*** et ***endsync***. Ces derniers permettent la spécification des comportements plus complexes.

Dans le document SMIL, il existe aussi l'élément ***switch*** qui permet à l'auteur de fixer des choix de présentation de son document.

Enfin, la description de l'essentiel des éléments du langage SMIL, nous a amené à remarquer quelques limitations [10] comme :

- la probabilité d'avoir des cycles en introduisant les attributs temporels événementiels.
- La référence des objets inexistants.

De cela peut résulter des documents incohérents.

Néanmoins, SMIL est un langage de synchronisation des objets médias qui peut servir à la création des scénarios très complexes, et contrairement à d'autres langages, il ne demande aucune connaissance particulière en programmation, il offre aussi l'avantage d'être indépendant du système d'exploitation des machines sur lesquelles il est utilisé. C'est pour ces raisons que nous avons adopté ce format pour la génération des documents multimédias dans notre éditeur.

Chapitre 3

Démarche de développement de l'outil

3.1 Introduction

Dans le chapitre précédent, nous avons parlé de l'édition multimédia et du standard SMIL qui répond parfaitement à notre besoin d'édition synchronisée, ainsi qu'une étude de quelques outils d'édition de documents multimédias synchronisés avec SMIL, comme *SMIL Composer*, *SMIL Wizard*...etc.

Dans ce présent chapitre, nous présentons les étapes les plus importantes dans la conception et la réalisation de notre système, en commençant par la présentation des différents problèmes qui causaient le développement de l'outil, ainsi que les objectifs qu'il faut réaliser. Nous passons, par la suite, à la présentation de la démarche de développement de l'outil depuis l'analyse jusqu'à l'implémentation.

3.2 Problématique et objectif

3.2.1 **Problématique**

Notre travail consiste à résoudre certains problèmes qui se résument dans ce qui suit :

- la création d'une présentation multimédia en utilisant un langage de programmation est une tache très difficile à manipuler, c'est ce qui fait la nécessité de créer un environnement d'édition des documents multimédias.

- il existe des environnements d'édition qui sont flous, c'est à dire que l'utilisateur de cet outil ne peut pas comprendre facilement ses fonctionnalités (manque d'ergonomie).

- un document multimédia est caractérisé par un aspect temporel, qui reste le plus grand problème à résoudre, ce qui fait que la synchronisation de plusieurs médias dans un document multimédia est une tâche très compliquée, c'est à nous de la rendre moins difficile en utilisant le langage d'intégration multimédia synchronisé (SMIL), décrit dans le chapitre 2, qui nous permet de définir le scénario temporel d'une manière très simple.

3.2.2 **Objectif**

L'objectif principal de notre travail est la création d'un environnement d'édition des documents multimédias synchronisés, qui soit fiable et convivial, en utilisant le langage de synchronisation SMIL. D'une manière plus détaillée, nous pouvons dire que nos buts sont :

- la réalisation d'une interface ergonomique et interactive.

- la traduction des actions de l'utilisateur effectuées au niveau de l'interface, en code SMIL.

3.3 Démarche de développement

Pour développer un logiciel, il est nécessaire de passer par plusieurs étapes, pour cela, des méthodes de développement ont été définies, ces méthodes permettent de mieux organiser, d'avoir une meilleure compréhension, de réduire la complexité des applications et permettent une plus grande facilité dans l'interprétation des concepts logiciels.

Pour la modélisation de notre travail, nous utilisons la notation UML qui représente un langage de modélisation et non pas une méthode objet [19], ça veut dire que l'UML ne décrit pas une démarche de développement de logiciel. Le processus de développement que nous utilisons est le modèle «en cascade», ce modèle est décri par Royce en 1970, qui a été largement employer depuis, pour la description générale des activités liées aux logiciel [19].

Le modèle «en cascade» présente un cycle de vie d'un logiciel par une suite de phases (analyse, conception, implémentation, test et maintenance) (Figure 3.1) qui s'enchaînent dans un déroulement linéaire depuis l'analyse des besoins jusqu'à la maintenance [19].

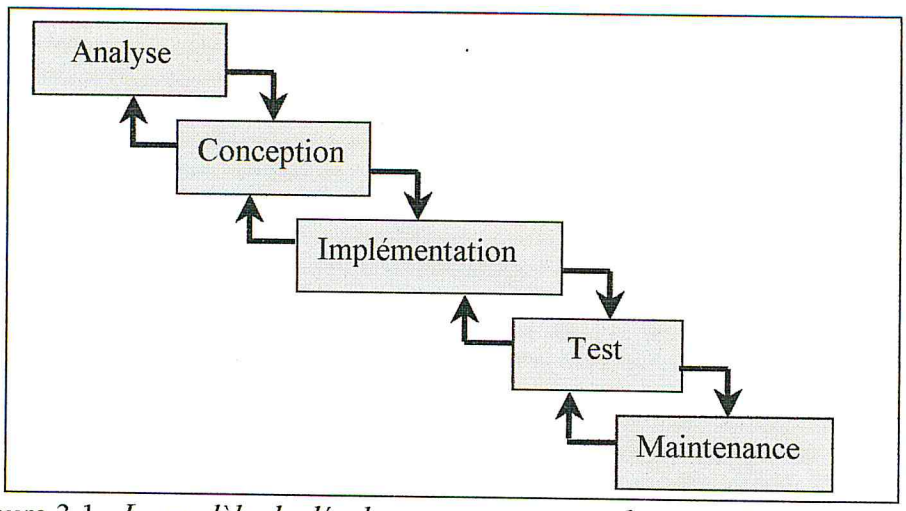


Figure 3.1 : *Le modèle de développement en cascade ouvre des points de visibilité sur le processus de développement*

3.4 Analyse

3.4.1 **Spécification des besoins**

La spécification des besoins est une étape essentielle au début du processus de développement. Son but est d'éviter de développer un logiciel non adéquat [22].

La finalité de cette étape est la description générale des fonctionnalités du système. Par la réponse à ces questions : "quelles sont les fonctions du système ?", "quels sont les utilisateurs du système?", "et qu'attendent-ils du système ?". Cette étape étudie le comportement du système exprimé sous la forme des cas d'utilisation, le contexte du système, les acteurs et les scénarios.

Notre système doit répondre aux exigences suivantes :

- *la simplicité d'utilisation* : quelque soit l'auteur, une prise en main rapide de l'outil et la possibilité de créer rapidement des scénarios, sont les premières attentes.

- *la puissance d'expression* : notre système doit permettre la spécification des schémas de synchronisation complexe.

- *la structuration* : l'auteur à envie de découper le document en entités plus petites (groupement) pour avoir une meilleure lisibilité, et aussi pour pouvoir récupérer une partie d'un scénario, afin de l'inclure dans un autre document.

- *la simplicité de modification* : pouvoir ajouter et retirer des éléments du document, modifier le placement spatial, l'enchaînement temporel et la structure logique du document, inter changer des médias, sans avoir à agir sur tous les éléments.

- *la cohérence* : l'auteur veut éviter les erreurs de spécification qu'il peut introduire. Une incohérence peut provenir d'une référence sur un objet inexistant, de deux conditions contradictoires, ...etc.

- *l'aide au diagnostic* : en cours d'édition, l'auteur a besoin d'une explication lorsqu'il commet une erreur. Par exemple, s'il modifie la valeur de la durée d'un objet et que celle-ci ne respecte pas la syntaxe, il aimerait que l'outil lui propose la syntaxe exacte.

- *la perception* : en cours d'édition, l'auteur a besoin d'avoir un aperçu sur l'exécution de son document. Par exemple, la disposition temporelle des éléments et le placement spatial des objets...etc.

- *l'adaptabilité* : l'auteur veut que son document prenne en compte des paramètres spécifiques en cour de restitution au lecteur. C'est le cas par exemple pour les commentaires audio, le choix du fichier doit se faire en fonction de la langue du lecteur, pour la résolution, s'adapter au format de l'écran.

- *la diffusion* : l'auteur veut générer un document qui ne soit pas dépendant d'un type de machine, d'un système d'exploitation, d'un outil d'édition ou d'un logiciel de visualisation.

Le choix du langage SMIL comme format de description du document couvre une grande partie de ces exigences, car il permet à l'auteur, outre de pouvoir spécifier des comportements complexes d'un point de vue temporel, de bénéficier des fonctions d'adaptabilité [18] offertes par l'élément ***switch*** du langage SMIL.

De plus, les systèmes de présentation et d'édition supportant ce format sont de plus en plus nombreux, ce qui est un avantage pour la diffusion des documents générés avec ce langage.

3.4.1.1 **Les cas d'utilisation**

La spécification des cas d'utilisation détermine le « quoi faire », c'est à dire les besoins de l'utilisateur. L'expérience montre que la technique des cas d'utilisation (use cases) se prête bien à la détermination des besoins d'utilisateurs [19].

Les cas d'utilisation décrivent sous la forme d'actions et de réactions le comportement du système du point de vue de l'utilisateur.

L'étude des cas d'utilisation débute par la détermination des acteurs (catégories d'utilisateurs) du système.

3.4.1.1.1 **Les acteurs**

Les acteurs sont, par définition, des éléments qui interagissent avec le système; dans notre cas, il y a un seul utilisateur principal qui est l'auteur des documents multimédias, il peut être un enseignant, un médecin, un étudiant, un journaliste…etc.

3.4.1.1.2 **Les cas d'utilisation principaux**

Nous allons présenter notre système en trois cas principaux, qui sont les suivants :

1) création d'une présentation.
2) modification d'une présentation.
3) visualisation d'une présentation.

De là, nous obtiendrons le schéma suivant :

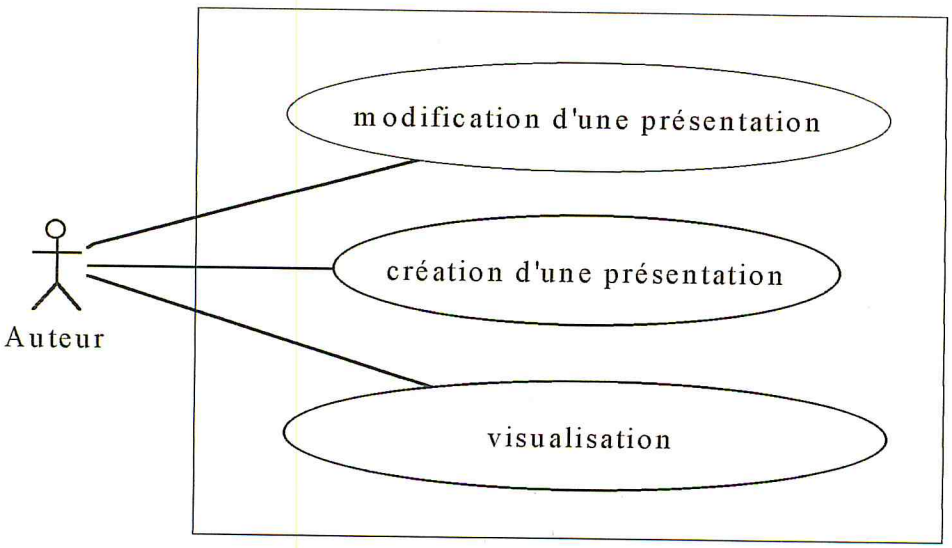


Figure 3.2: *Diagramme des cas d'utilisation pour les cas d'utilisation principaux*

Nous expliquerons, par la suite, chaque cas d'utilisation en donnant les diagrammes des sous cas d'utilisation qui lui correspondent, afin de pouvoir comprendre le diagramme général (récapitulatif de tous les diagrammes particuliers).

1) Création d'une présentation :

L'auteur se charge de la création d'un document multimédia (présentation multimédia) en collectant les médias (texte, son, image, vidéo), en spécifiant leurs emplacements spatiaux et en les synchronisant, comme il peut ajouter des liens.

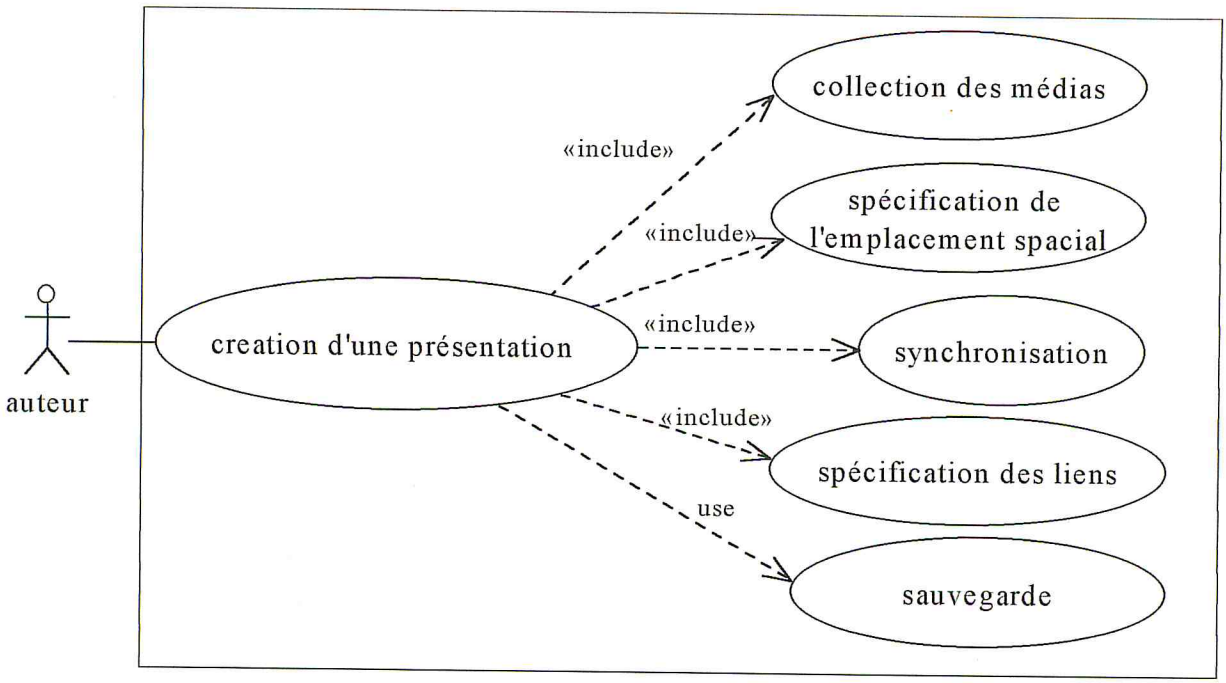


Figure 3.3: *Diagramme des cas d'utilisation pour le cas d'utilisation « création d'une présentation »*

Cas d'utilisation « Collection des médias » :

Dans ce cas, l'utilisateur peut effectuer deux opérations :

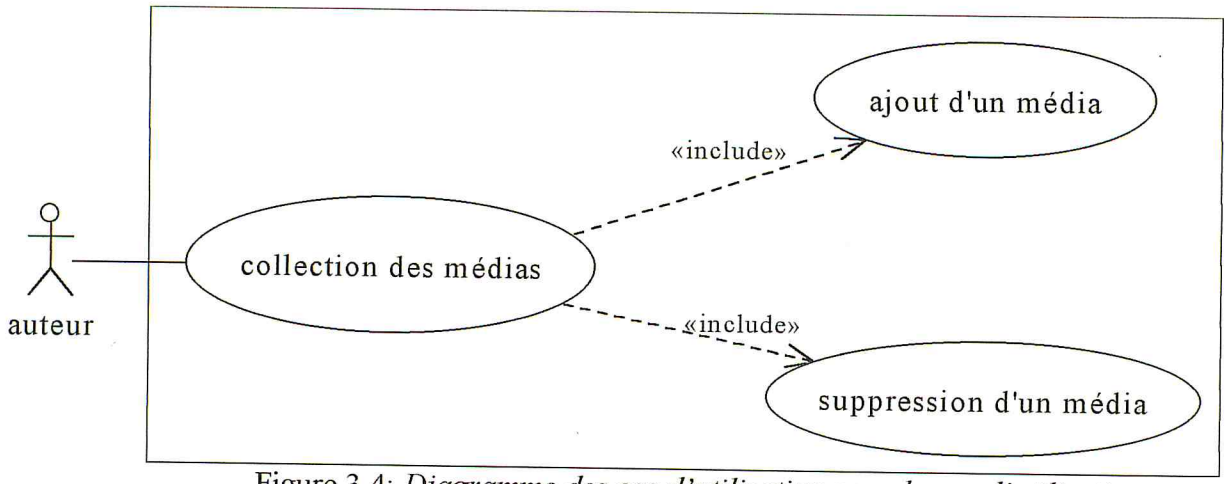


Figure 3.4: *Diagramme des cas d'utilisation pour le cas d'utilisation « collection des médias »*

- *Cas d'utilisation « ajout d'un média »* : Quant l'utilisateur ajoute un média, il est obligé de lui affecter une région (une zone d'apparition).

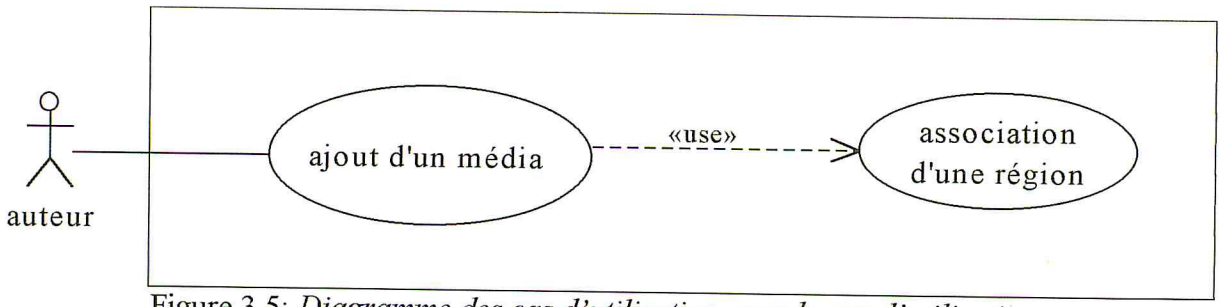


Figure 3.5: *Diagramme des cas d'utilisation pour le cas d'utilisation « ajout d'un média »*

- *Cas d'utilisation « suppression d'un média »* : Pour la suppression d'un média, l'auteur sélectionne le média puis il effectue l'opération de suppression.

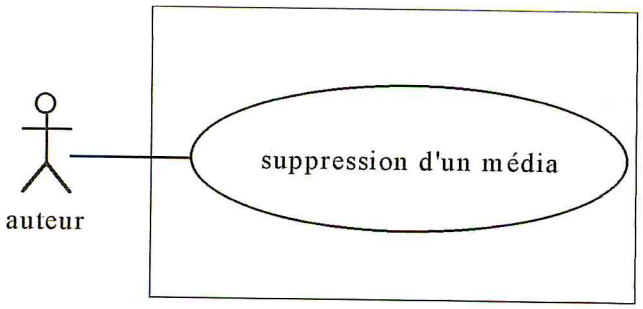


Figure 3.6: *Diagramme des cas d'utilisation pour le cas d'utilisation « suppression d'un média »*

Cas d'utilisation « spécification de l'emplacement spatial » :

La spécification de l'emplacement spatial consiste en la création des zones appelées *régions*, qui seront par la suite affectées à des médias.

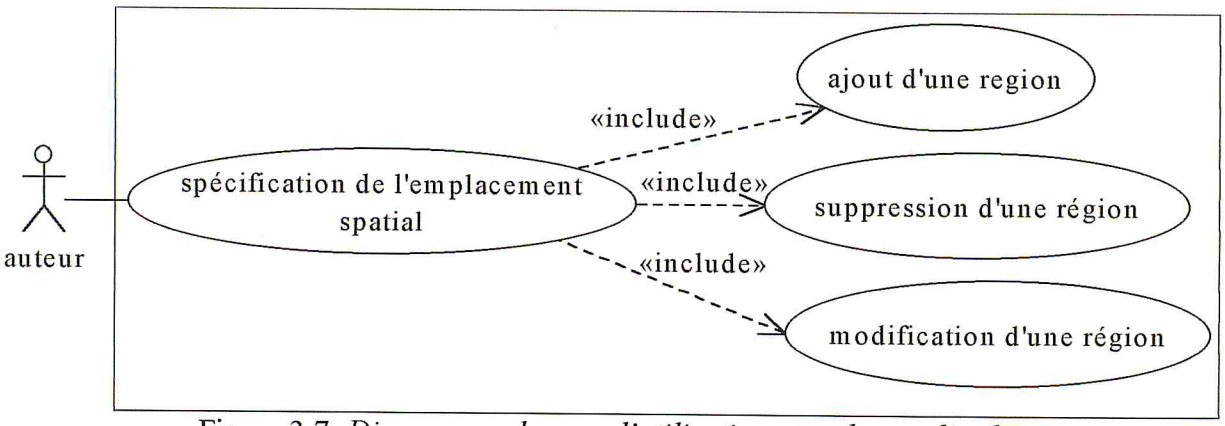


Figure 3.7: *Diagramme des cas d'utilisation pour le cas d'utilisation « spécification de l'emplacement spatial »*

- *Cas d'utilisation « ajout d'une région »* : Pour chaque présentation multimédia un ensemble de régions est défini, pour préciser l'endroit de l'exécution (déroulement) de chaque média (donc l'utilisateur peut ajouter/supprimer une région).

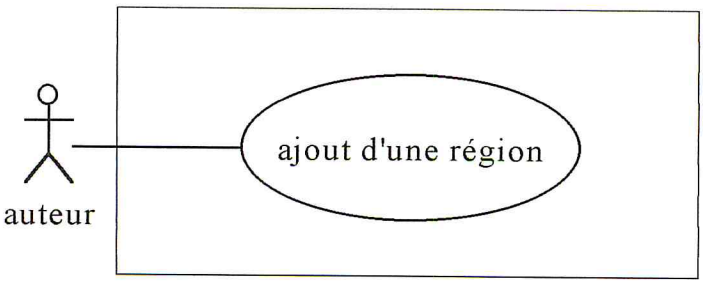


Figure 3.8: *Diagramme des cas d'utilisation pour le cas d'utilisation « ajout d'une région »*

- *Cas d'utilisation « suppression d'une région »* :

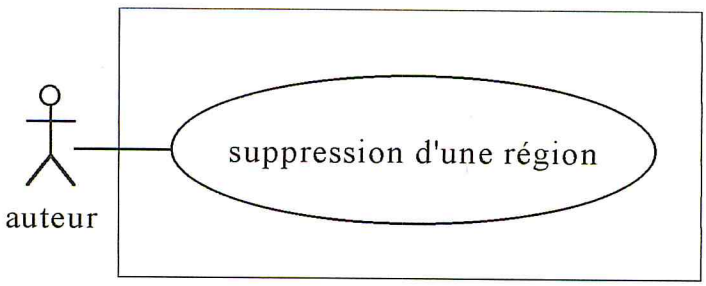


Figure 3.9: *Diagramme des cas d'utilisation pour le cas d'utilisation « suppression d'une région »*

- *Cas d'utilisation « modification d'une région »* : Pour qu'une région soit modifiée, l'auteur accède aux propriétés de cette dernière (longueur, largeur, top, left,...etc.) puis il effectue les modifications voulues.

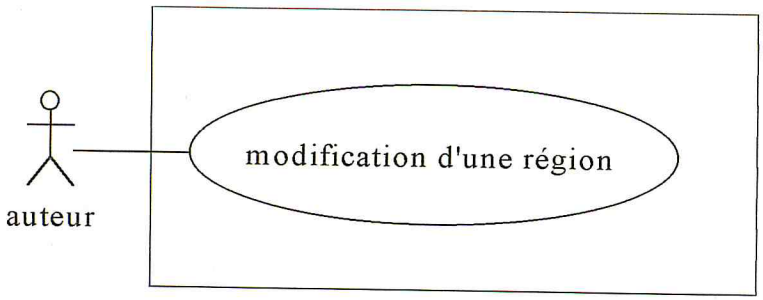


Figure 3.10: *Diagramme des cas d'utilisation pour le cas d'utilisation « modification d'une région »*

Cas d'utilisation « synchronisation » :

La synchronisation est l'étape la plus importante dans la création d'un document multimédia, dans laquelle l'auteur définit le scénario temporel de sa présentation, pour cela, il utilise les éléments de synchronisation (les blocs **par** et les blocs **seq**) qui doivent être ajoutés.

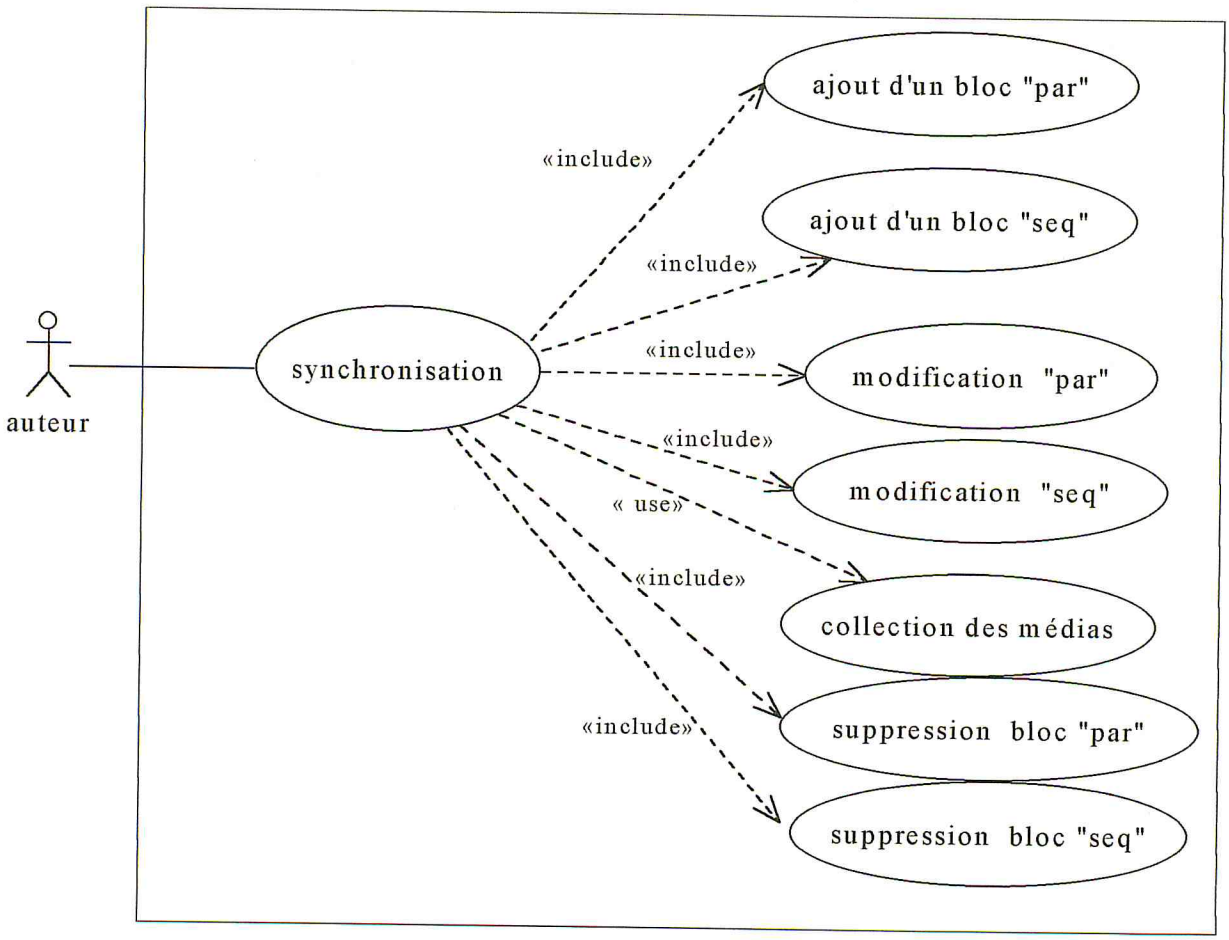


Figure 3.11: *Diagramme des cas d'utilisation pour le cas d'utilisation « synchronisation »*

- *Cas d'utilisation « modification d'un bloc par »* : La modification d'un bloc parallèle consiste en la modification de ses propriétés ou de son contenu (les médias et les blocs «seq» qu'il contient).

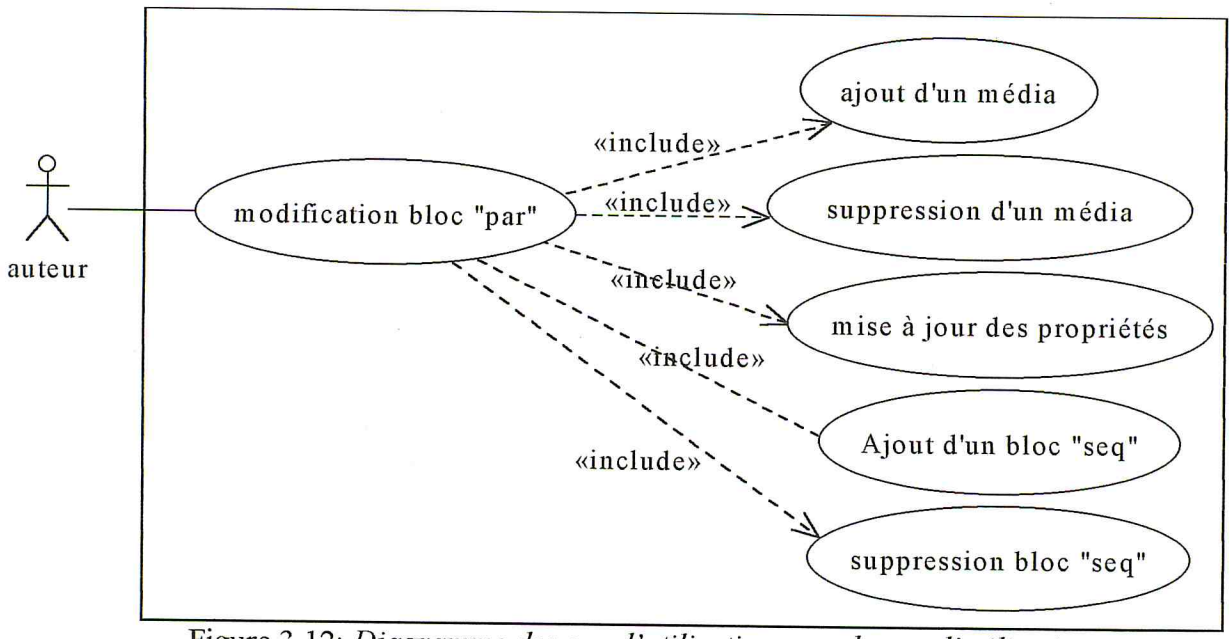


Figure 3.12: *Diagramme des cas d'utilisation pour le cas d'utilisation « modification d'un bloc "par" »*

- *Cas d'utilisation « modification d'un bloc seq »* : Ce cas, est similaire à celui de la modification d'un bloc **par**.

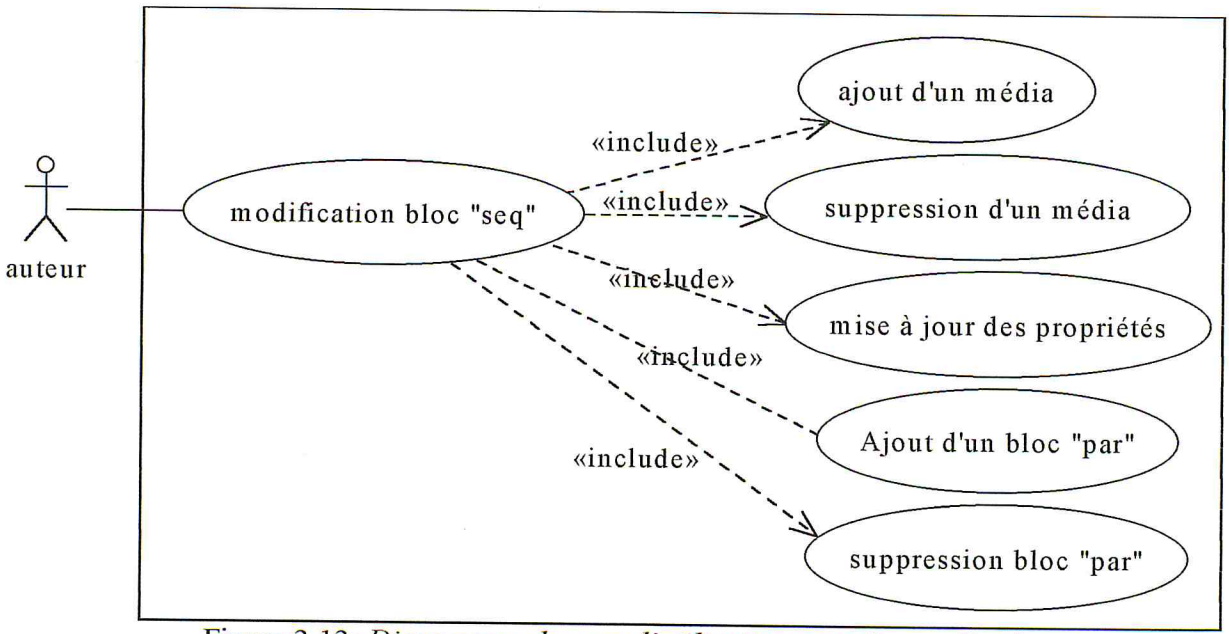


Figure 3.13: *Diagramme des cas d'utilisation pour le cas d'utilisation « modification d'un bloc "seq" »*

Cas d'utilisation « spécification d'un lien » :

L'auteur peut créer des liens en spécifiant le média source et l'objet de destination qui peut être un autre média, un autre document multimédia…etc. Un lien peut être supprimé ou modifié, comme il peut y avoir plusieurs liens à la fois.

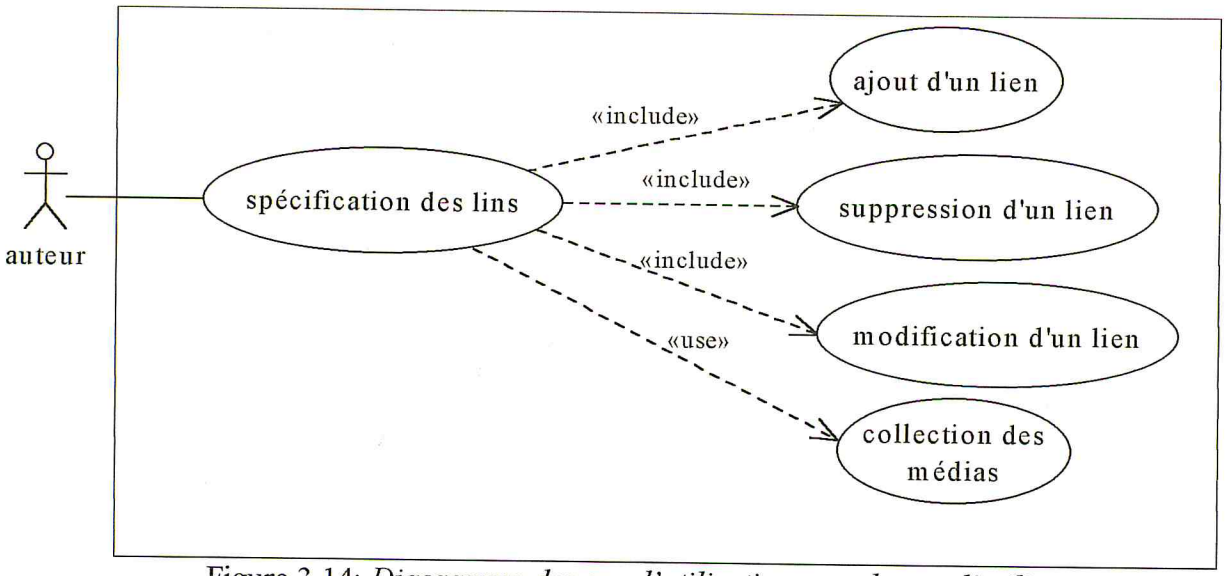


Figure 3.14: *Diagramme des cas d'utilisation pour le cas d'utilisation « création d'un lien »*

- *Cas d'utilisation « modification d'un lien »* : La modification d'un lien consiste en la modification de ses propriétés, comme par exemple la zone d'activation du lien, le moment d'activation …etc.

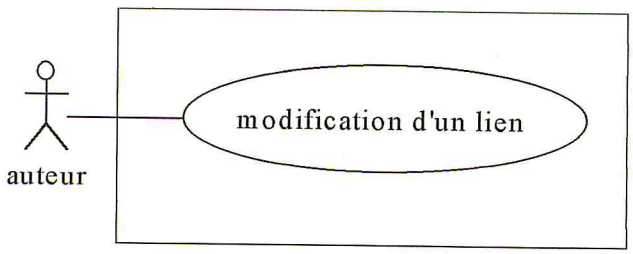


Figure 3.15: *Diagramme des cas d'utilisation pour le cas d'utilisation « modification d'un lien »*

Cas d'utilisation « sauvegarde » :

Quant l'auteur termine la phase de la création de son document multimédia, il sera obliger de le sauvegarder (stocker) pour garder sa trace.

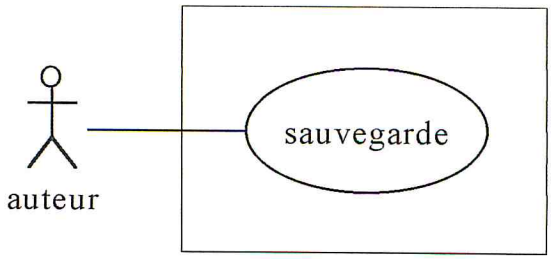


Figure 3.16: *Diagramme des cas d'utilisation pour le cas d'utilisation « sauvegarde ».*

2) Modification d'une présentation :

Ce cas d'utilisation est similaire à celui de la création, sauf l'utilisateur utilise un document multimédia existant. De là, nous pouvons déduire qu'il existe une relation de généralisation entre les deux cas cités (création et modification d'une présentation).

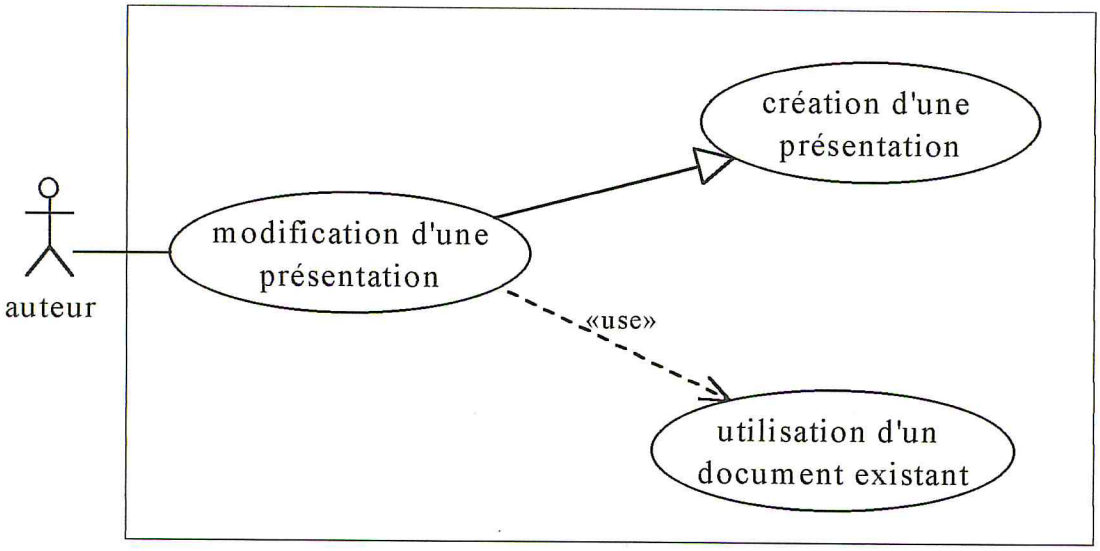


Figure 3.17: *Diagramme des cas d'utilisation pour le cas d'utilisation « modification d'une présentation »*

3) Cas d'utilisation « visualisation » :

L'auteur peut visualiser son document en appelant un lecteur depuis le système (éditeur).

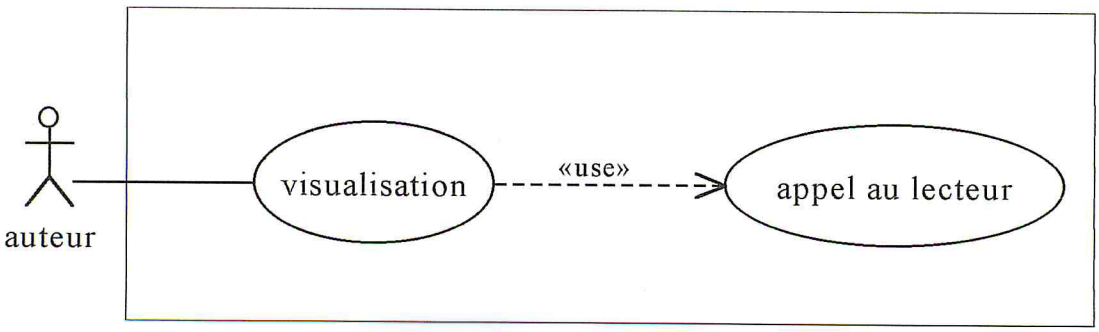


Figure 3.18:*Diagramme des cas d'utilisation pour le cas d'utilisation « visualisation »*

3.4.1.1.3 **Diagramme des cas d'utilisation**

Les différentes fonctionnalités offertes par notre outil forment ainsi un ensemble de *cas d'utilisation* (*"Use Case"*), exprimés dans les sections précédentes, afin de les formaliser, UML propose au travers des diagrammes des cas d'utilisation de répertorier et de structurer les fonctionnalités que le futur système offrira à ses utilisateurs.

La figure suivante illustre ce diagramme. (Figure 3.19)

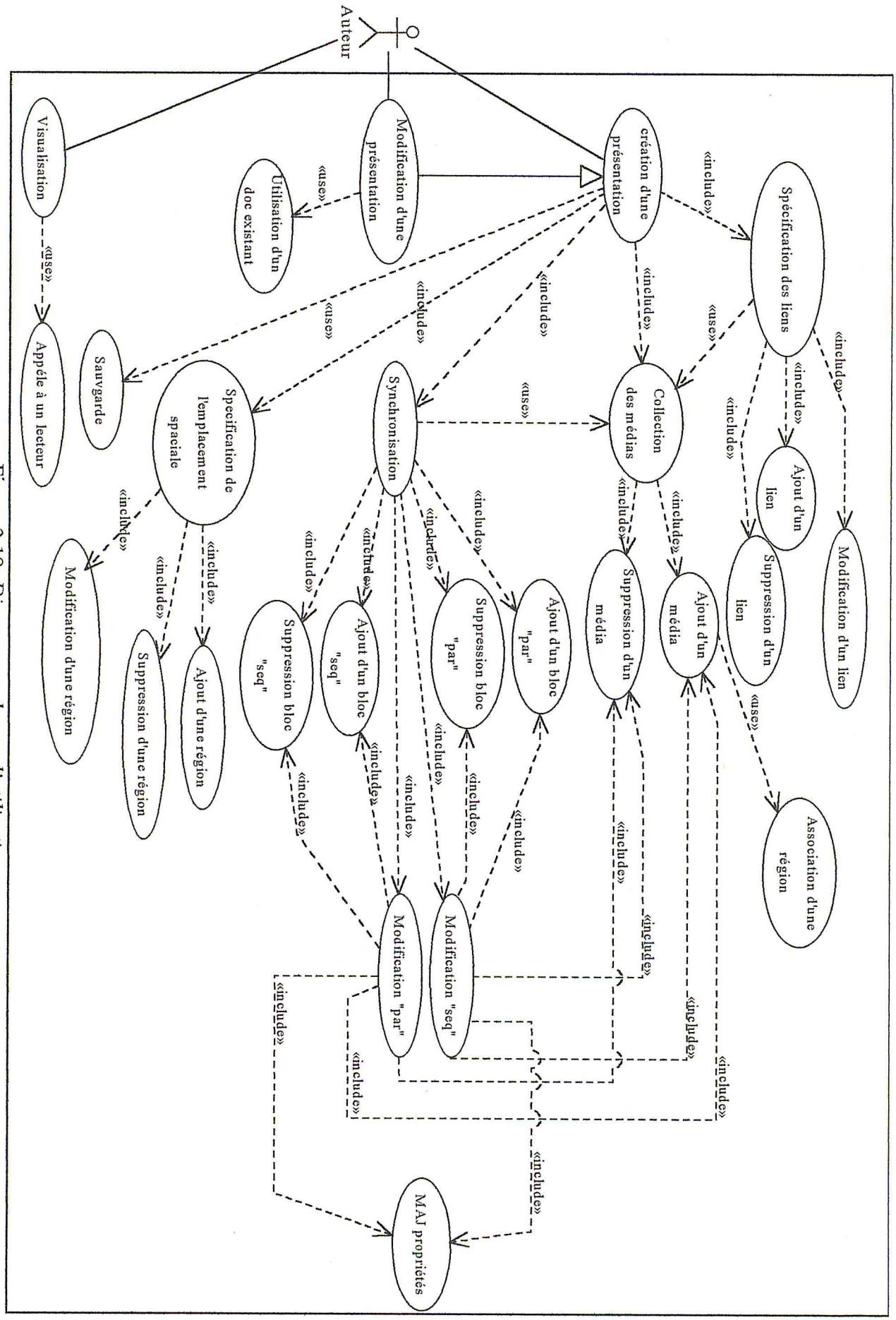


Figure 3.19: *Diagramme de cas d'utilisation*

3.4.1.1.4 **Diagramme de séquence**

Les cas d'utilisation de UML ont certes l'avantage d'être graphiquement très simples et donc faciles à appréhender. Malheureusement, cette simplicité ne va pas sans une certaine pauvreté sémantique [19], cependant les diagrammes de séquence nous permetent de bien schématiser les scénarios des cas d'utilisation et montrent les interactions entre plusieurs objets selon un point de vue temporel.

1) Création d'une présentation

Scénario

- l'auteur demande la création d'une présentation.
- le système ouvre un nouveau document (chargement espace de travail 'work space').
- l'auteur ajoute des médias, donne leurs emplacements spatiaux, établit la synchronisation puis demande la sauvegarde.
- le système demande un nom et un emplacement mémoire pour le document.
- l'auteur donne le nom et l'emplacement puis confirme.
- après l'enregistrement, le système réinitialise le processus.

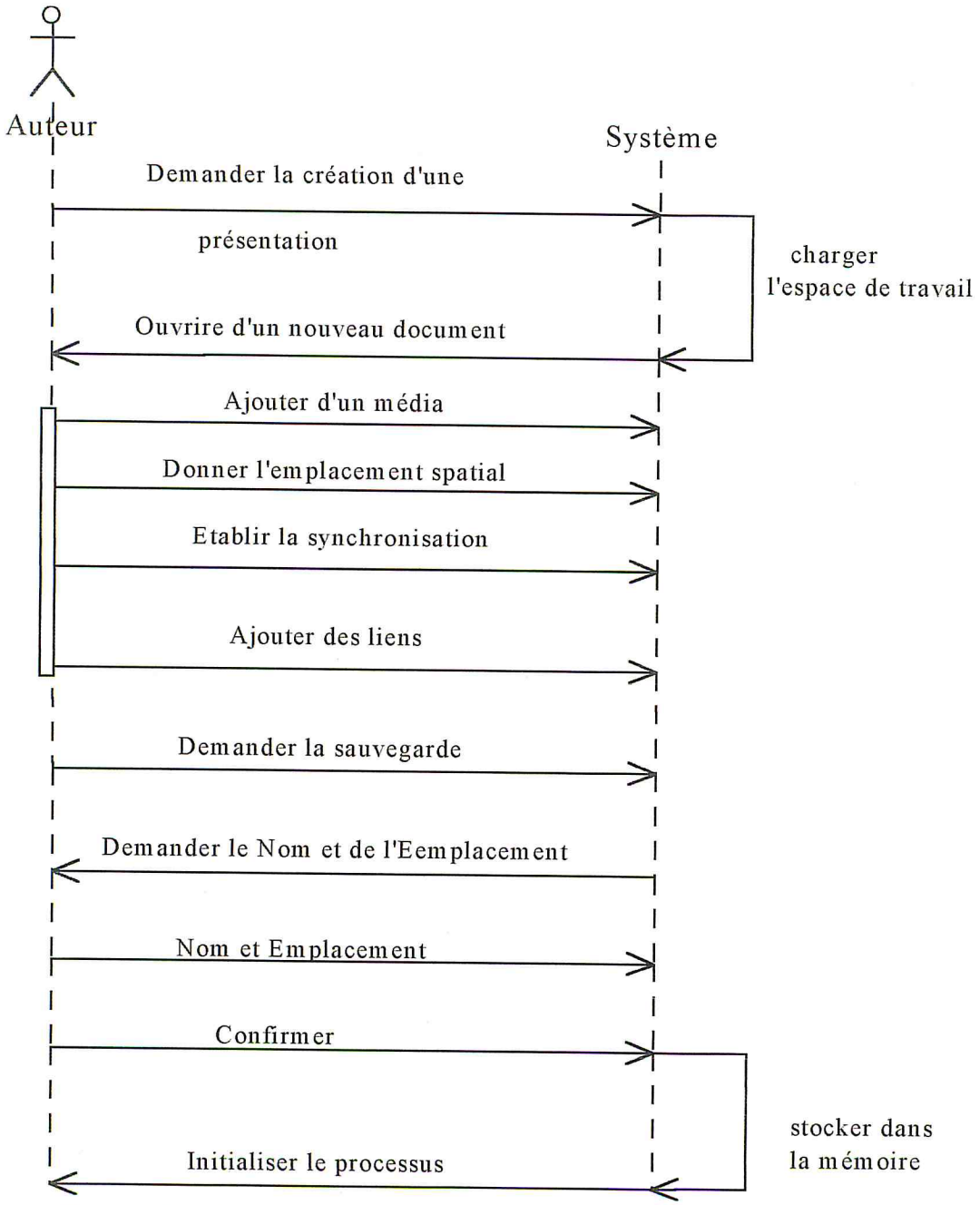


Figure 3.20:*Diagramme de séquence pour la création d'une présentation*

Ajout d'un média

Scénario

- l'auteur demande l'ajout d'un média.
- le système demande le chemin et la région du média.
- l'auteur donne le chemin ainsi que la région d'apparition du média.
- le système ajoute le média et rend la main à l'utilisateur.

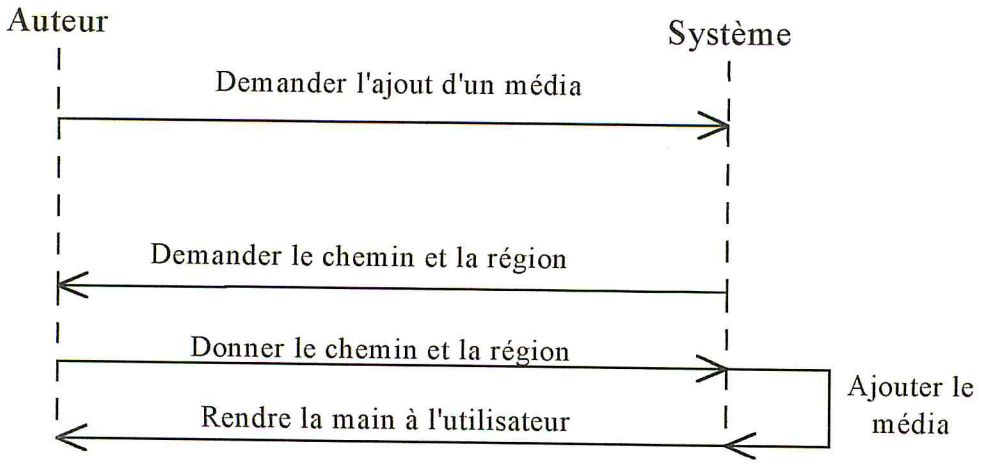


Figure 3.21:*Diagramme de séquence pour l'ajout d'un média*

Ajout d'un élément de synchronisation (par et seq)

Scénario

- l'auteur demande l'ajout d'un élément de synchronisation après la sélection de son emplacement.
- le système ajoute cet élément puis rend la main à l'utilisateur.

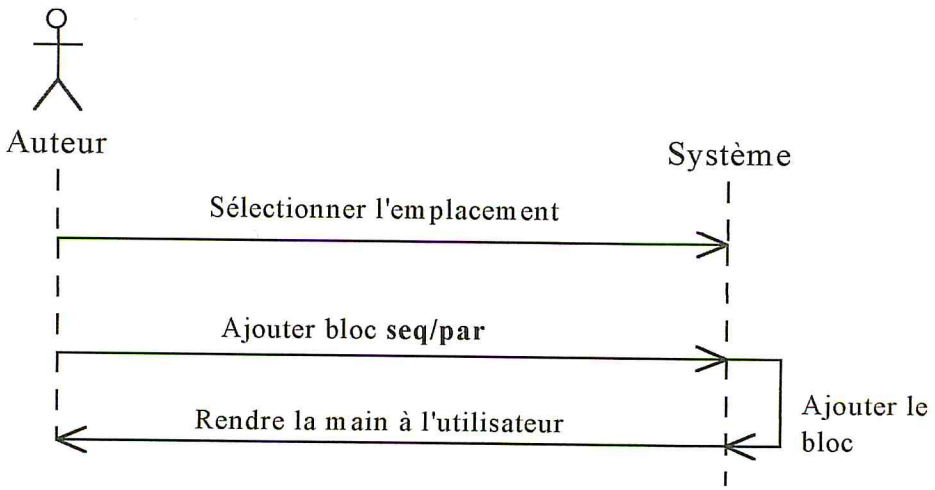


Figure 3.22: *Diagramme de séquence pour l'ajout d'un élément de synchronisation*

Ajout d'une région

Scénario

- l'auteur demande un ajout d'une région.
- le système ajoute la région puis rend la main à l'auteur.

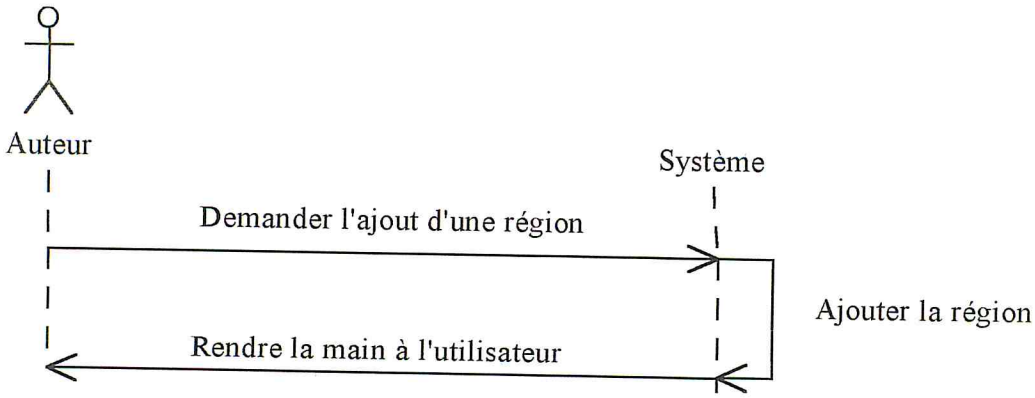


Figure 3.23: *Diagramme de séquence pour l'ajout d'une région*

Modification d'un élément de synchronisation ou d'une région

Scénario

- l'auteur effectue la modification d'un élément de synchronisation ou d'une région en modifiant ses propriétés.
- le système fait une mise à jour puis rend la main à l'utilisateur.

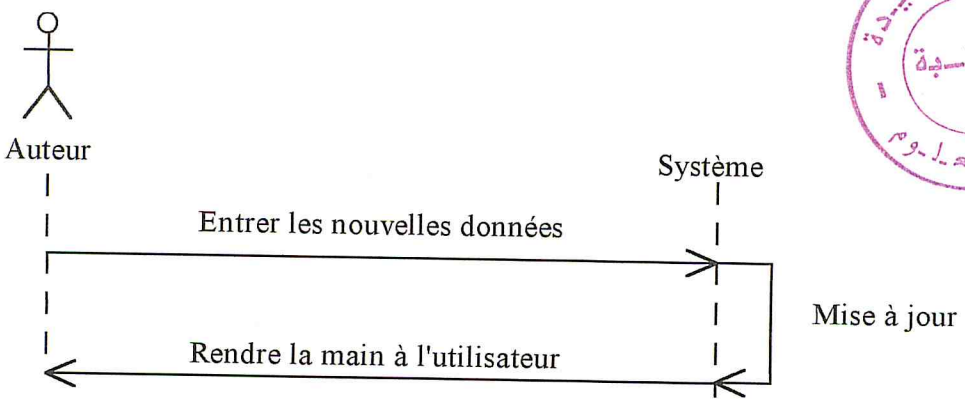


Figure 3.24:*Diagramme de séquence pour la modification d'un élément de synchronisation ou d'une région*

Suppression d'un objet

Scénario

- l'auteur sélectionne l'objet à supprimer (région, média, élément de synchronisation, lien) puis lance l'opération de suppression.
- le système demande la confirmation.
- l'auteur confirme son opération.
- le système supprime l'objet sélectionné après la confirmation puis rend la main à l'utilisateur.

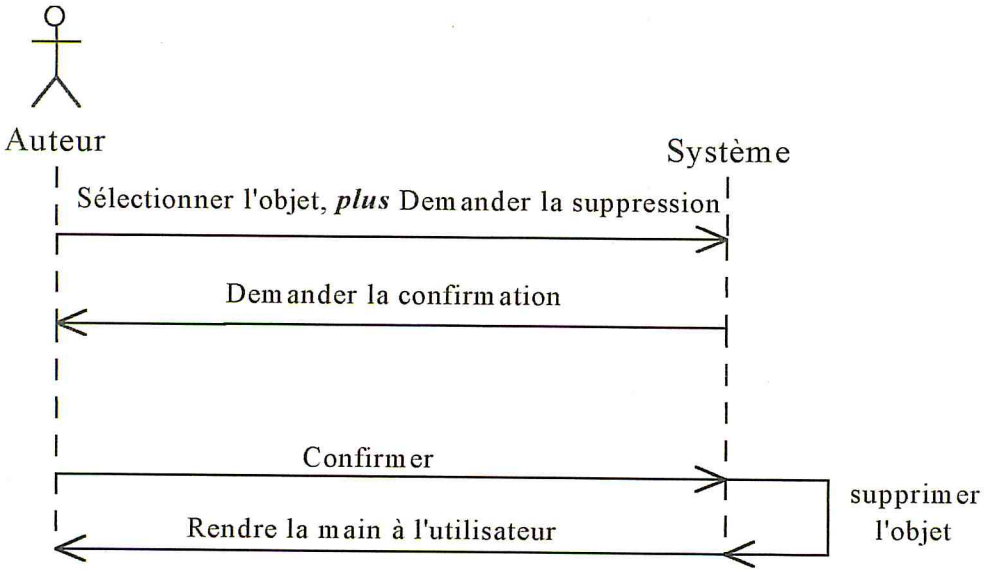


Figure 3.25:*Diagramme de séquence pour la suppression d'un objet*

Spécification d'un lien

Scénario

- l'auteur demande l'ajout d'un lien.
- le système demande l'identifiant du lien et sa cible.
- après que le système reçoit les données demandées, il affiche le rectangle des liens.
- l'auteur spécifie la taille du rectangle (la zone d'activation du lien).
- le système effectue une mise à jour du lien puis rend la main à l'utilisateur.

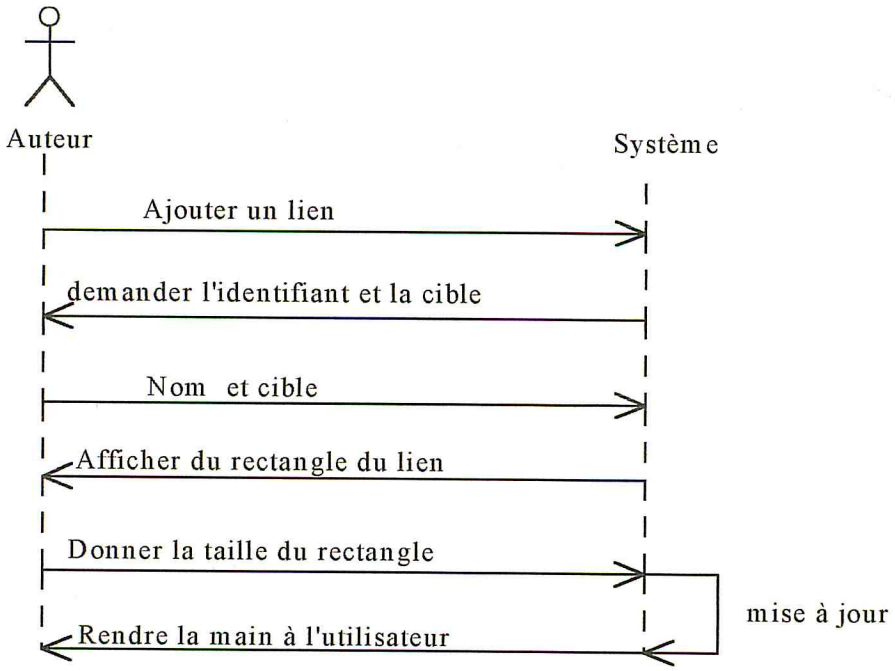


Figure 3.26:*Diagramme de séquence pour la création d'un lien*

La sauvegarde

Scénario

- l'auteur demande une sauvegarde,
- le système demande le nom et l'emplacement mémoire de la présentation,
- l'auteur donne le nom ainsi que l'emplacement mémoire puis il valide,
- après le stockage, le système rend la main à l'auteur.

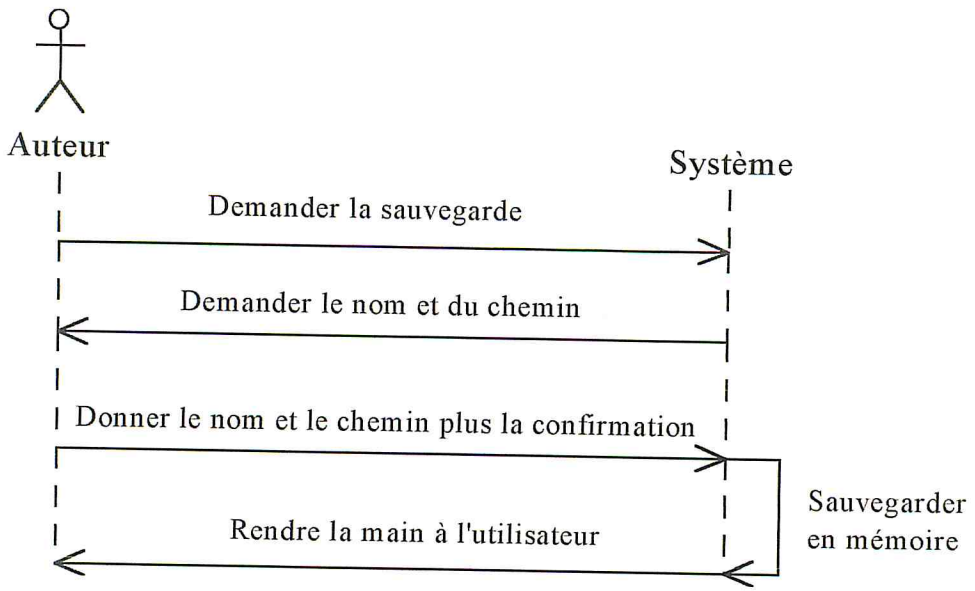


Figure 3.27:*Diagramme de séquence pour la sauvegarde*

2) Modification d'une présentation

Scénario

- l'auteur demande l'ouverture d'un document multimédia,
- le système ouvre le document s'il est disponible,
- l'auteur effectue la modification qu'il veut (ajout/suppression des médias, redéfinir l'emplacement spatial, modifier la synchronisation), puis demande de nouveau la sauvegarde,
- le système enregistre les modifications effectuées puis réinitialise le processus.

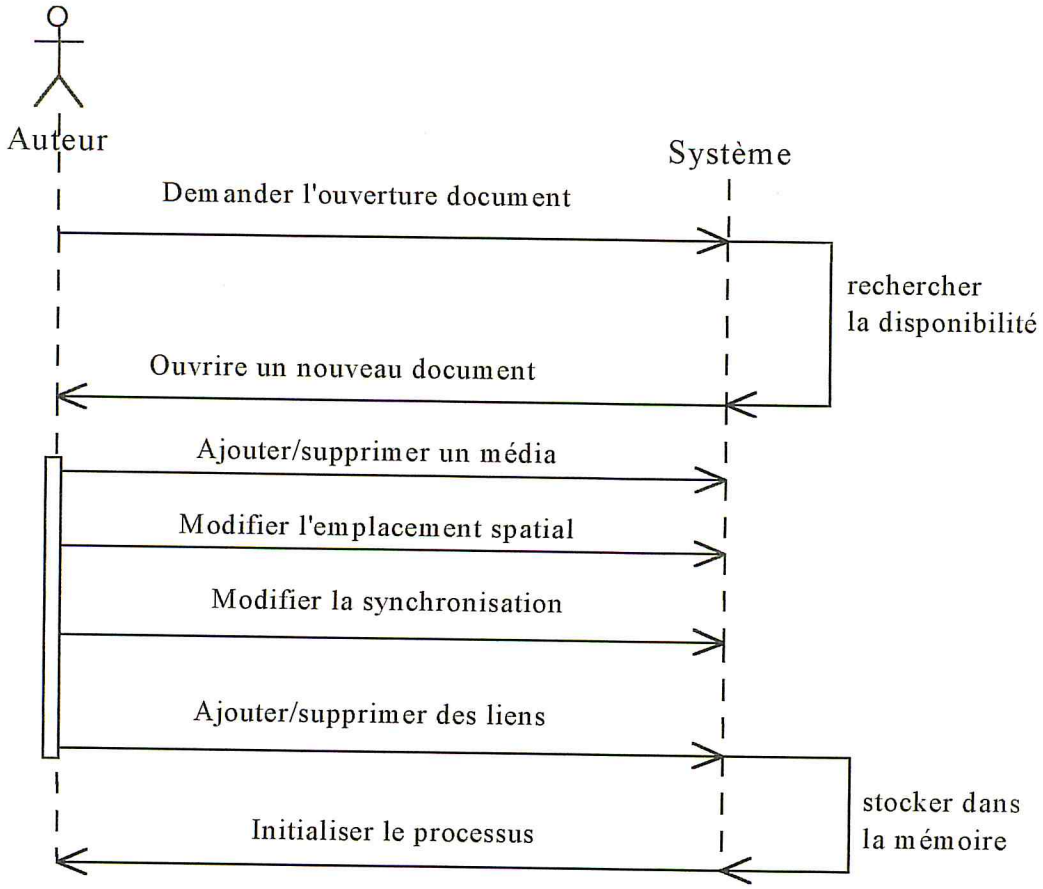


Figure 3.28:*Diagramme de séquence pour la modification d'une présentation*

3) La visualisation

Scénario

- après la sauvegarde, l'auteur demande une visualisation de la présentation.
- le système fait un appel à un lecteur qui permet la visualisation de cette présentation.

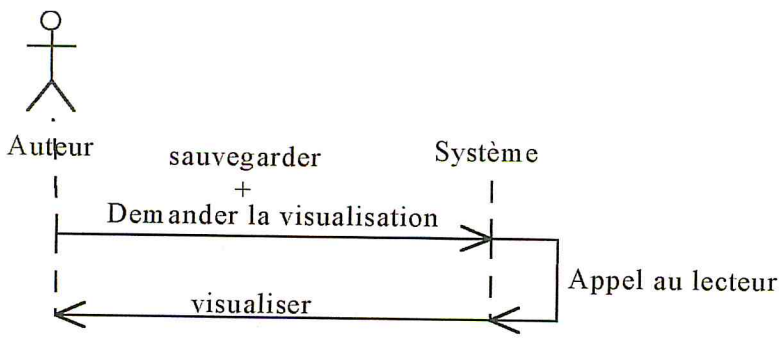


Figure 3.29:*Diagramme de séquence pour la visualisation*

3.4.1.1.5 .**Diagramme de collaboration**

Les fonctionnalités décrites par les cas d'utilisation sont réalisées par des collaborations d'objets du domaine [19], d'où il est envisageable, d'employer les diagrammes de collaborations bien que ces derniers ne sont qu'une variante des diagrammes de séquences et expriment de ce fait la même sémantique. Nous avons choisis comme exemple: la création d'un document multimédia

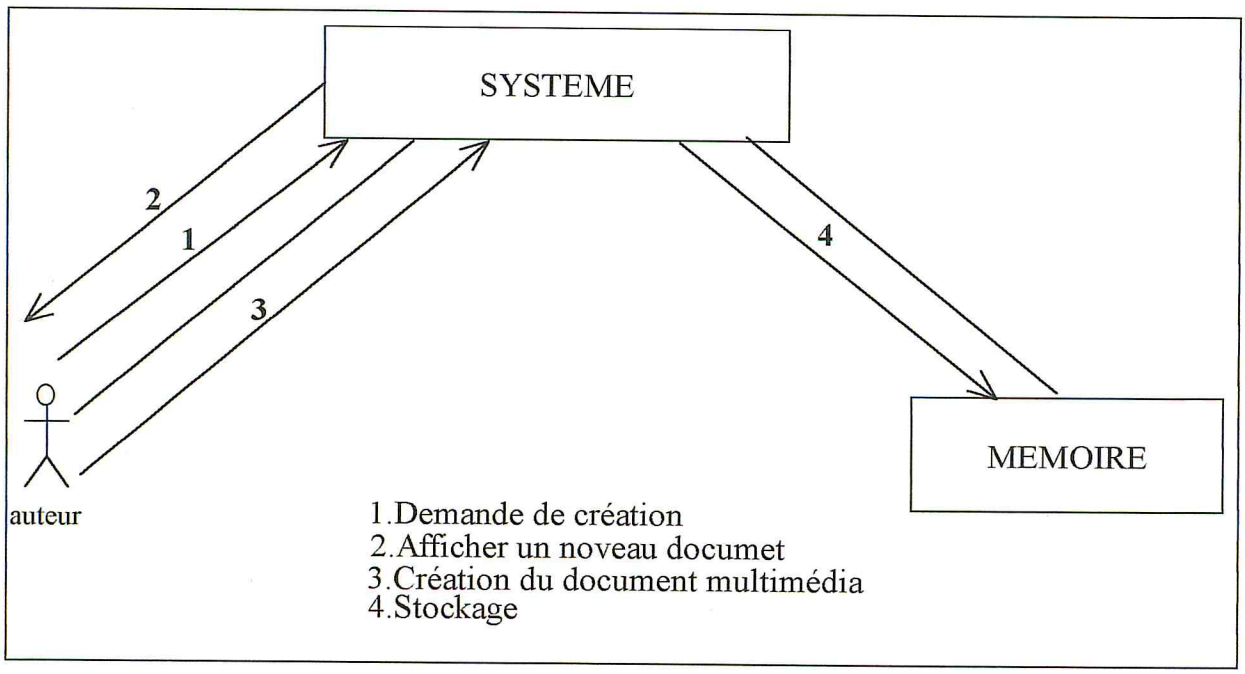


Figure 3.30: *Diagramme de collaboration de la création d'un document multimédia*

3.4.1.1.6 **Diagramme d'activité**

Ce diagramme permet de décrire le déroulement d'un cas d'utilisation. Il est possible de décrire les acteurs responsables de chaque activité par l'utilisation des «couloirs d'activités» qui permettent de répartir graphiquement les différentes activités entre les acteurs opérationnels [20]. Chaque activité est placée dans le «couloir» correspondant à l'acteur qui assume cette activité. Nous utiliserons ce formalisme pour présenter le diagramme d'activité des cas d'utilisation « création d'une présentation » et « modification d'une présentation ».

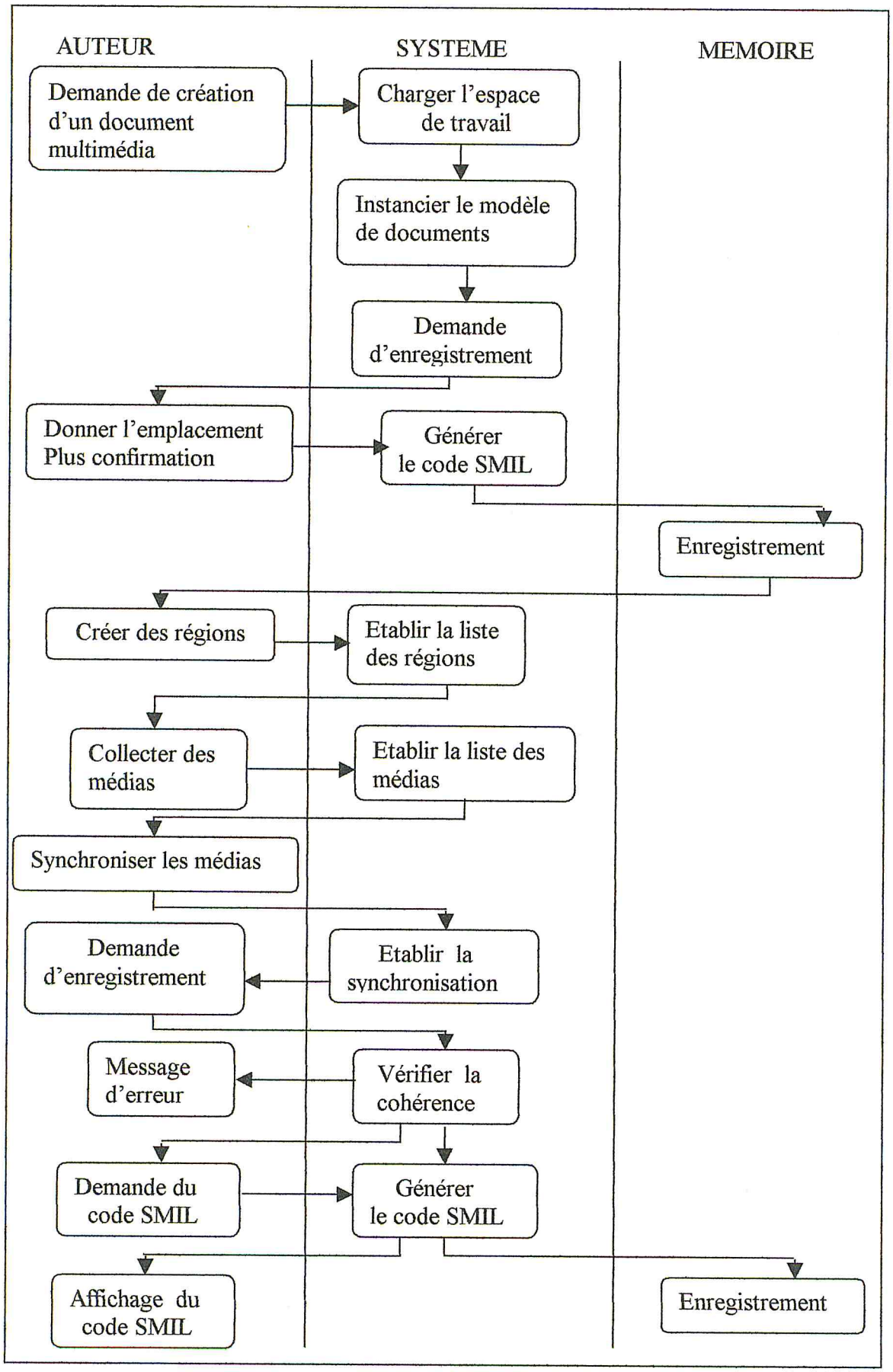
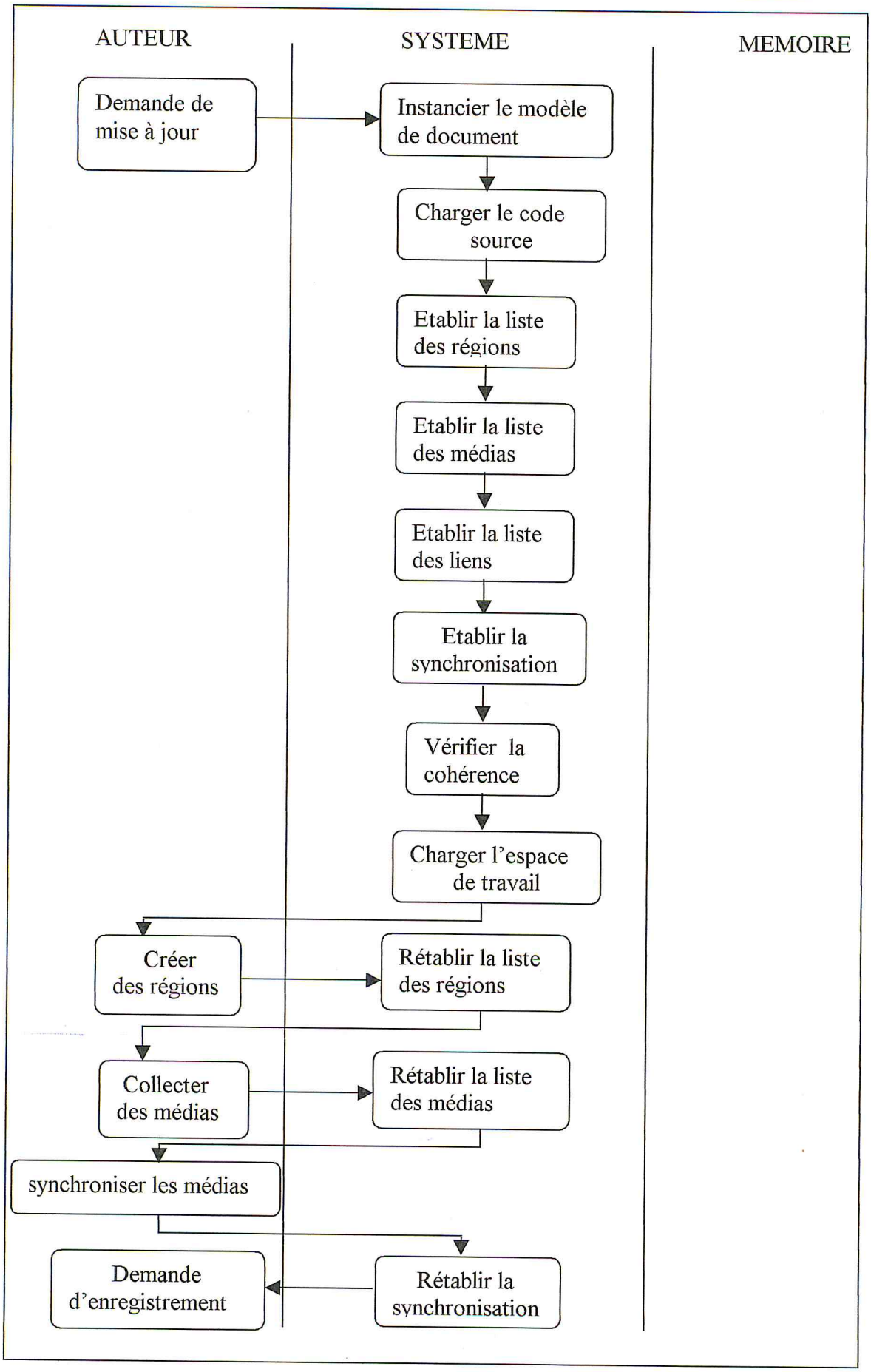


Figure 3.31: *Diagramme d'activité de la création d'un document multimédia.*


AUTEUR
SYSTEME
MEMOIRE
Demande de mise à jour
Instancier le modèle de document
Charger le code source
Etablir la liste des régions
Etablir la liste des médias
Etablir la liste des liens
Etablir la synchronisation
Vérifier la cohérence
Charger l'espace de travail
Créer des régions
Rétablir la liste des régions
Collecter des médias
Rétablir la liste des médias
synchroniser les médias
Demande d'enregistrement
Rétablir la synchronisation

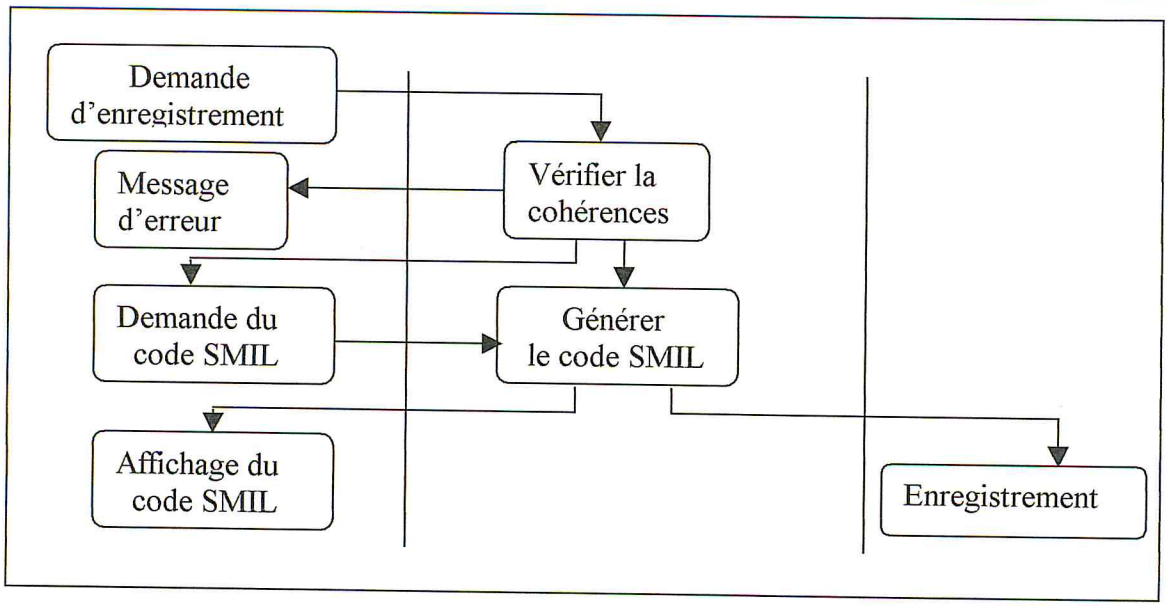


Figure 3.32: *Diagramme d'activité de la modification d'un document multimédia.*

Remarque :

Le code SMIL d'un document multimédia peut être généré par deux opérations différentes effectuées par l'utilisateur à :

- la demande de la sauvegarde,
- la demande du code SMIL

3.5 Conception

La conception s'intéresse d'abord au « comment », à savoir la solution du problème énoncé; elle commence par une conception dite « globale » qui décrit l'architecture du système, elle se poursuit par une conception détaillée.

Pour modéliser la conception, UML propose les diagrammes de classes, de composants et de packages.

3.5.1 **Conception globale**

La conception globale a pour but de décomposer le logiciel en modules et de préciser les interfaces et les fonctions de chaque module. A l'issue de cette étape, on obtient une description de l'architecture du logiciel et un ensemble de spécifications de ses divers composants [17].

Voici le diagramme de collaboration qui montre les différents modules composant notre système, et leurs interactions :

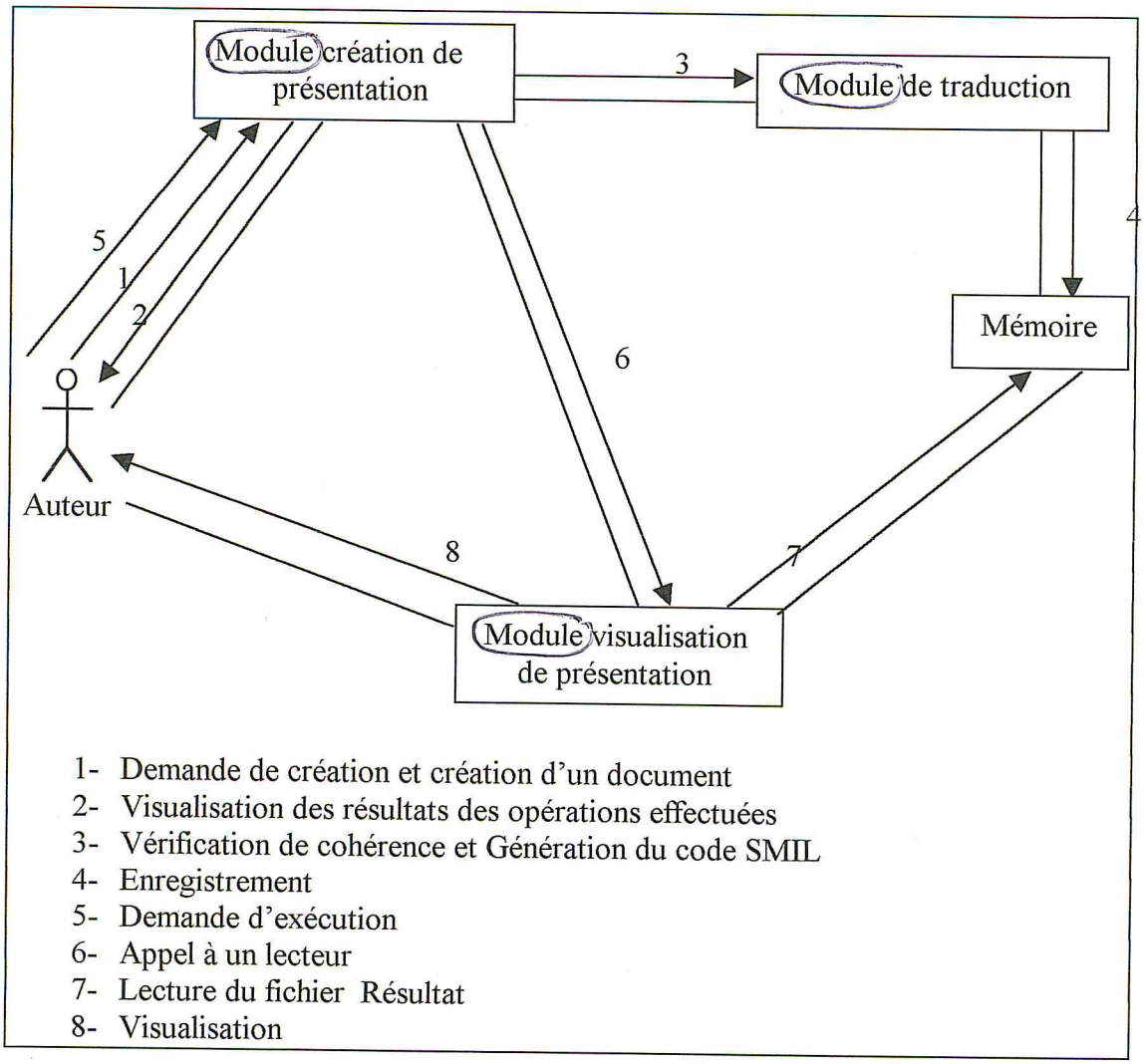


Figure 3.33 : *Diagramme de collaboration des modules du système.*

3.5.1.1 **Architecture de notre éditeur**

En se basant sur le diagramme de collaboration des modules décris auparavant (Figure3.33), nous proposons l'architecture suivante pour notre éditeur:

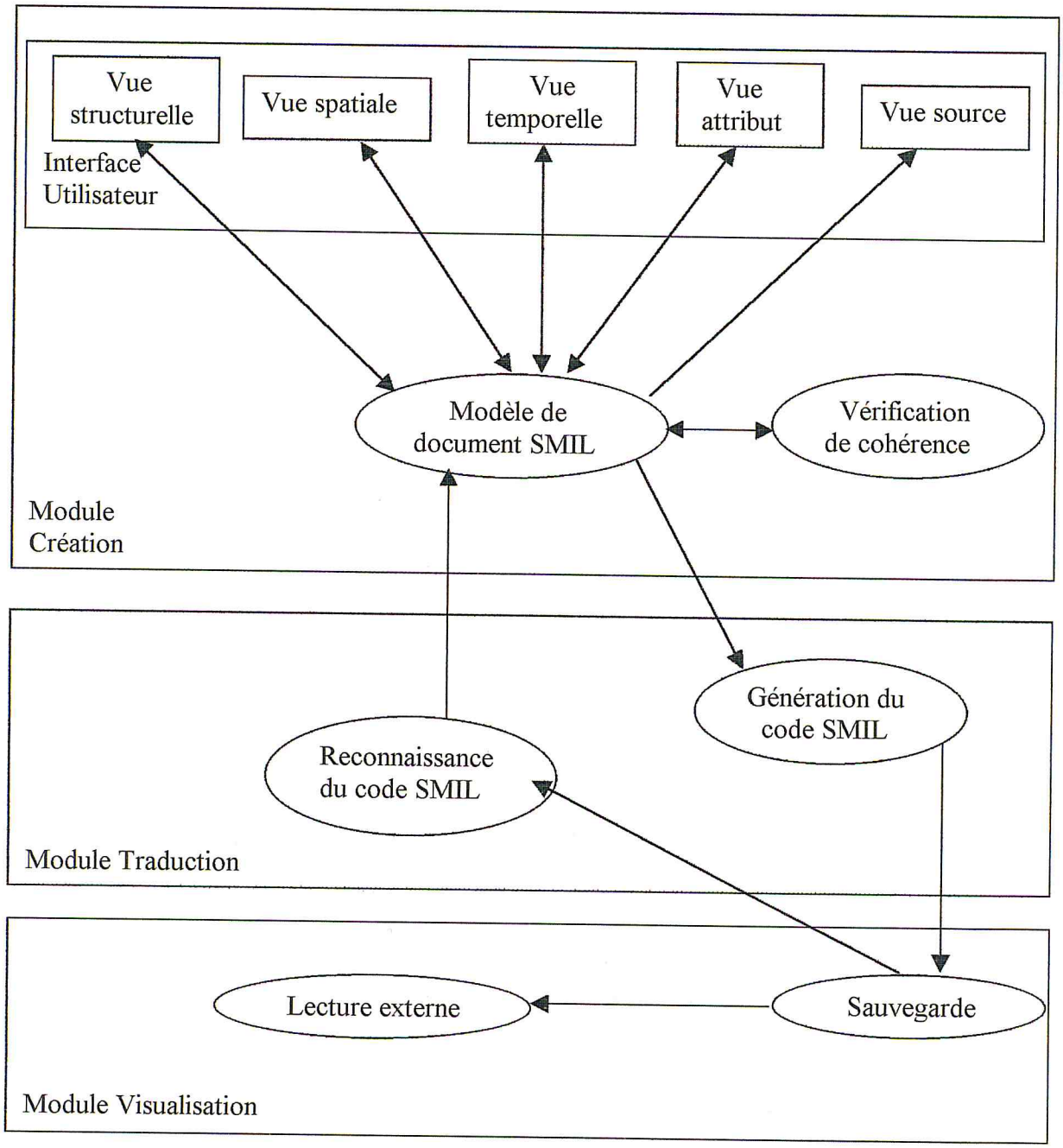


Figure 3.34 : *Architecture de notre éditeur multimédia.*

Dans l'architecture que nous avons proposée (Figure 3.34) nous distinguons trois modules qui sont :

- Module1 : Création
- Module2 : Traduction
- Modele3 : Visualisation

- le module «Création» se charge de la création d'un document multimédia. Le scénario qui lui correspond est le suivant :

  l'auteur collecte les médias de son document en spécifiant leurs chemins, ainsi que leurs zones d'apparition dans le document, en même temps qu'il réalise son scénario temporel; après que le système ait vérifié la cohérence du document, l'utilisateur reprend la main et peut l'enregistrer.

- le module «Traduction» a comme objectif la génération et la reconnaissance du code SMIL correspondant au document.

- le module «visualisation» est spécifié pour la lecture du document multimédia. Le scénario qui lui correspond est le suivant :

  après l'opération de sauvegarde, l'auteur demande la visualisation d'un document multimédia, pour cela, le système fait appel à un lecteur externe, ce dernier lit le fichier SMIL qui correspond à ce document.

En se basant sur notre conception globale, l'architecture de notre système peut être représentée en UML à l'aide de diagramme de composants qui est illustré dans la figure suivante (Figure 3.35) :

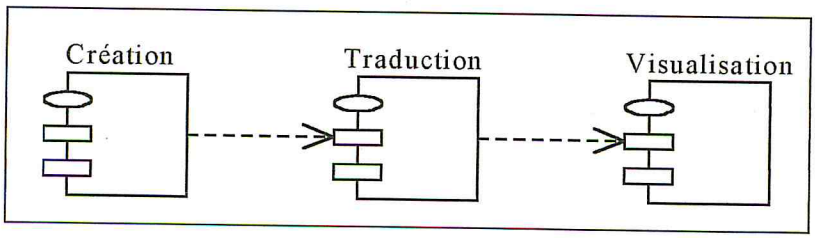


Figure 3.35 : *Diagramme de composants de notre système.*

3.5.2 **Conception détaillée**

La conception détaillée fournit pour chaque module, une description détaillée de la manière dont les fonctions du composant sont réalisées: algorithmes et représentation des données [22].

3.5.2.1 **Le module « Création »**

Nous allons présenter sous forme d'un paquetage, les différents éléments que nous avons utilisé pour l'établissement du module de création. Cela est donné dans la figure suivante (Figure 3.36) :

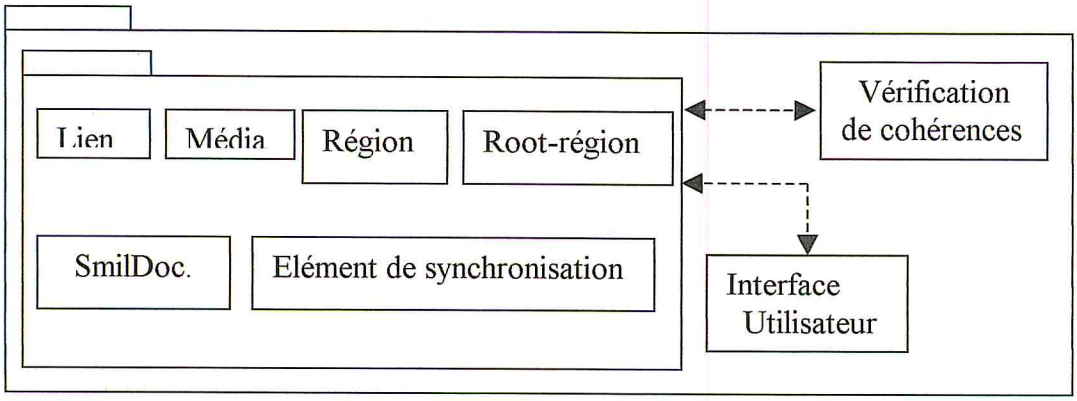


Figure 3.36 : *les éléments du module création.*

Ce module englobe deux éléments de modélisation qui sont : l'interface utilisateur, la vérification de cohérences et le modèle de document SMIL, ce dernier est représenté par un autre paquetage qui contient les éléments suivants : SmilDoc, root-région, lien, média, région, élément de synchronisation et stockage.

3.5.2.1.1 .**Interface utilisateur**

Notre éditeur a une interface contenant :

- une zone décrivant l'arborescence qui sert d'une part à donner un aperçu de tous les éléments composants le document, et d'autre part, de décrire l'organisation hiérarchique de celui-ci, cette zone est appelée **vue structurelle**.

- une zone graphique dans laquelle seront définies les régions, qui vont déterminer l'espace réservé à l'affichage des médias, ainsi qu'une représentation des liens, cette zone est appelée **vue spatiale**.

- une zone où seront affichés les différents attributs d'un objet sélectionné dans la vue hiérarchique, cette zone est appelée **vue attribut**.

- une zone texte qui sert à visualiser le code SMIL correspondant à la présentation en cours de création ou de modification, cette zone est appelée **vue source**.

- une zone temporelle qui permet la visualisation du scénario temporel du document, cette zone est appelée **vue temporelle**.

3.5.2.1.2 .**Modèle de document SMIL**

Le modèle de document SMIL nous permet de définir les différents aspects d'un document multimédia, à savoir la dimension logique, spatiale, hypermédia et temporelle, cette dernière définit le scénario temporel du document.

Dans notre modèle, nous avons défini trois dimensions : la dimension spatiale, en utilisant des régions, la dimension temporelle en utilisant des éléments de synchronisation qui définissent le scénario temporel, et la dimension hypermédia en utilisant des objets liens.

Notre modèle contient les éléments suivants :

- *SmilLDoc* : cet élément est l'élément principale du modèle, car il contient la représentation des objets (médias, régions, éléments de synchronisation ...etc.) composant le document, en les stockant dans des listes, il définit aussi la synchronisation entre les objets médias.

- *Média* : représente une occurrence de chaque média utilisé, nous définissons cette occurrence en donnant le chemin du média, et son emplacement spatial en l'associant à une région.

- *Région* : chaque média a une région, cette dernière est la zone d'apparition du média. Donc pour chaque document correspond une liste de régions.

- *Root-layout* : cet élément est nécessaire dans notre système puisque c'est le support de toutes les régions du document, c'est la « fenêtre principale ».

- *Elément de synchronisation* : l'utilisation des éléments de synchronisation est obligatoire pour la définition du scénario temporel, ces éléments sont les blocs parallèles et les blocs séquentiels.

- *Lien* : pour naviguer dans une présentation multimédia, l'auteur doit définir une liste de liens.

3.5.2.1.3 .**Vérification de cohérence**

Il existe des cas d'incohérences qui peuvent se présenter dans la création d'un document SMIL, ces incohérences ne permettent pas l'exécution du document. Nous essayons dans notre éditeur de détecter ces incohérences, et de les signaler à l'utilisateur.

Voici les incohérences qui peuvent figurer dans un code SMIL :

- dépassement de la taille spécifiée pour la « fenêtre principale », c'est-à-dire qu'il existe une ou plusieurs régions qui sont définies en dehors des limites spécifiées pour la fenêtre principale.

Exemple:

```
<head>
 <layout>
  <root-layout background-color="black" height="600" width="600"/>
  <region id="r1" left="0" top="0" height="500" width="400" />
  <region id="r2" left="400" top="0" height="500" width="300" />
 </layout>
</head>
```

Le problème dans cet exemple est que la deuxième région (mentionnée en couleur) dépasse la largeur de root-layout (la fenêtre principale), cette dernière est égale à 600 pixels tandis que la largeur de la région plus son top est égale à 700pixels (top=400, width=300).

La solution apportée à ce cas est de limiter à l'utilisateur la définition des valeurs des paramètres d'une région par rapport à la fenêtre principale, que ce soit à partir de la vue attribut ou spatiale.

- présence de plusieurs médias, à l'exception du son, dans une même région à des périodes qui se chevauchent, ce qui est interdit dans le langage SMIL.

Exemple:

```
<par>
 <audio src="\les - Hotel california.mp3" region="r2"/>
 <text src="\Titre.rt" region="r2" dur="30"/>
 <img src="\Firewks3.jpg" region ="r2" dur="30"/>
</par>
```

Dans cet exemple, le texte **Titre.rt** et l'image **Firewks3.jpg** partagent la même région au même temps car ils sont dans un bloc parallèle.

Nous proposons une solution qui résout une partie du problème posé, cette partie consiste en l'association d'une même région à deux médias appartenant directement au même bloc parallèle. La solution est d'omettre les régions associées aux objets médias, à l'exception de l'audio, appartenant au même bloc parallèle au moment où l'utilisateur va associer une région à ce nouvel objet média.

- présence de cycle : ce cas là, peut se produire quand il y a une dépendance temporelle entre plusieurs médias (à partir de trois médias), dans une telle situation, le document ne sera jamais exécuté même s'il est syntaxiquement correcte.

Pour mieux comprendre ce problème, nous présentons l'exemple suivant :

Exemple :

```
<par>
 <audio src="\ph.mp3" region="r2"  id ="son"     begin="id(texte)(begin)" />
 <text src="\Titre.rt" region="r2"  id ="texte"  begin="id(image)(begin)"/>
 <img src="\Firewks3.jpg" region ="r1"  id ="image" begin="id(son)(begin)" />
</par>
```

Dans cet exemple, les médias : audio, texte et image sont dépendant temporellement ; le début de chaque média dépend de l'autre, ce qui introduit une boucle qui empêche l'exécution du document.

Nous ne traitons pas ce cas, car notre éditeur ne permet pas de faire des dépendances.

- utilisation d'un même identifiant pour plusieurs objets : ce cas doit être évité, car un identifiant est unique, sinon des conflits peuvent avoir lieu lors de l'affectation d'une région ayant un identifiant, qui existe déjà, à un média, aussi lors des références des éléments dans les déclarations des relations temporelles.

  Exemple :

```
<head>
 <layout>
  <root-layout background-color="black" height="600" width="600"/>
  <region id="r1" left="0" top="0" height="500" width="400" />
  <region id="r1" left="400" top="0" height="500" width="300" />
 </layout>
</head>
```

  Dans une telle situation, nous signalons le problème à l'utilisateur qui se chargera de le corriger.

- valeur de l'horloge syntaxiquement erronée, dans ce cas nous devons présenter à l'utilisateur la syntaxe correcte utilisée dans notre éditeur pour la spécification du temps.

3.5.2.2 **Le module « Traduction »**

Le module « Traduction » contient deux fonctions s'appliquant sur le modèle qui sont : la génération du code SMIL et une fonction de reconnaissance du code SMIL.

Ces éléments sont présentés dans la figure suivante (Figure 3.37) sous le format d'un paquetage :

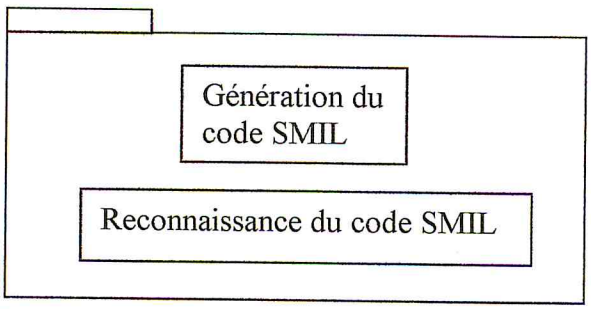


Figure 3.37 : *les éléments du module traduction.*

3.5.2.2.1 .**Génération du code SMIL**

La génération du code se fait en partant du modèle, ceci en générant le code définissant l'aspect spatial à partir des régions contenues dans le document, et en générant ensuite la partie du code qui correspond à l'aspect temporel défini par le modèle, cette dernière est faite en générant le code des objets décris par le scénario temporel, ces objets peuvent être des médias, des liens ou des éléments de synchronisation.

3.5.2.2.2 .**Reconnaissance du code SMIL**

Dans cette étape, l'éditeur va lire un fichier SMIL qui existe, nous allons, à partir de ce fichier, identifier les différentes balises composants le code SMIL de ce fichier, et créer pour chaque balise l'objet lui correspondant dans le modèle de document.

3.5.2.3 **Le module « Visualisation »**

Le module « Visualisation » permet à l'auteur de visualiser le résultat, qui est un document multimédia. Notons qu'avant chaque visualisation, il faut une sauvegarde.
Ce module comprend les éléments suivants : Stockage, appel à un lecteur externe.

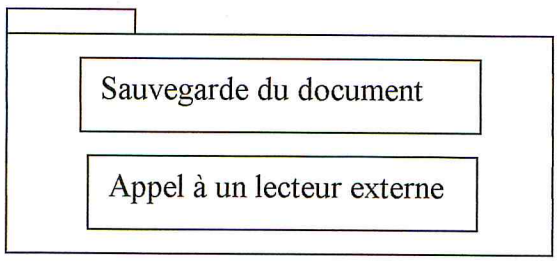


Figure 3.38 : *les éléments du module visualisation*

3.5.2.3.1 .**Sauvegarde du document**

Quand l'auteur décide de visualiser son document, le code SMIL généré doit être sauvegardé sous l'extension **.smi* ou **.smil* .

3.5.2.3.2 .**Appel au lecteur externe**

Quand l'utilisateur veut visualiser son document, le système fait appel à un lecteur qui se charge de la lecture du document multimédia (fichier SMIL).

3.5.2.4 **Diagramme de classes**

Pour exprimer de manière générale la structure statique de notre système, nous utilisons le diagramme de classe qui représente cet aspect statique en termes de classes et

de relations entre ces classes, et il décrit de manière abstraite les liens potentiels d'un objet vers d'autres objets [21].

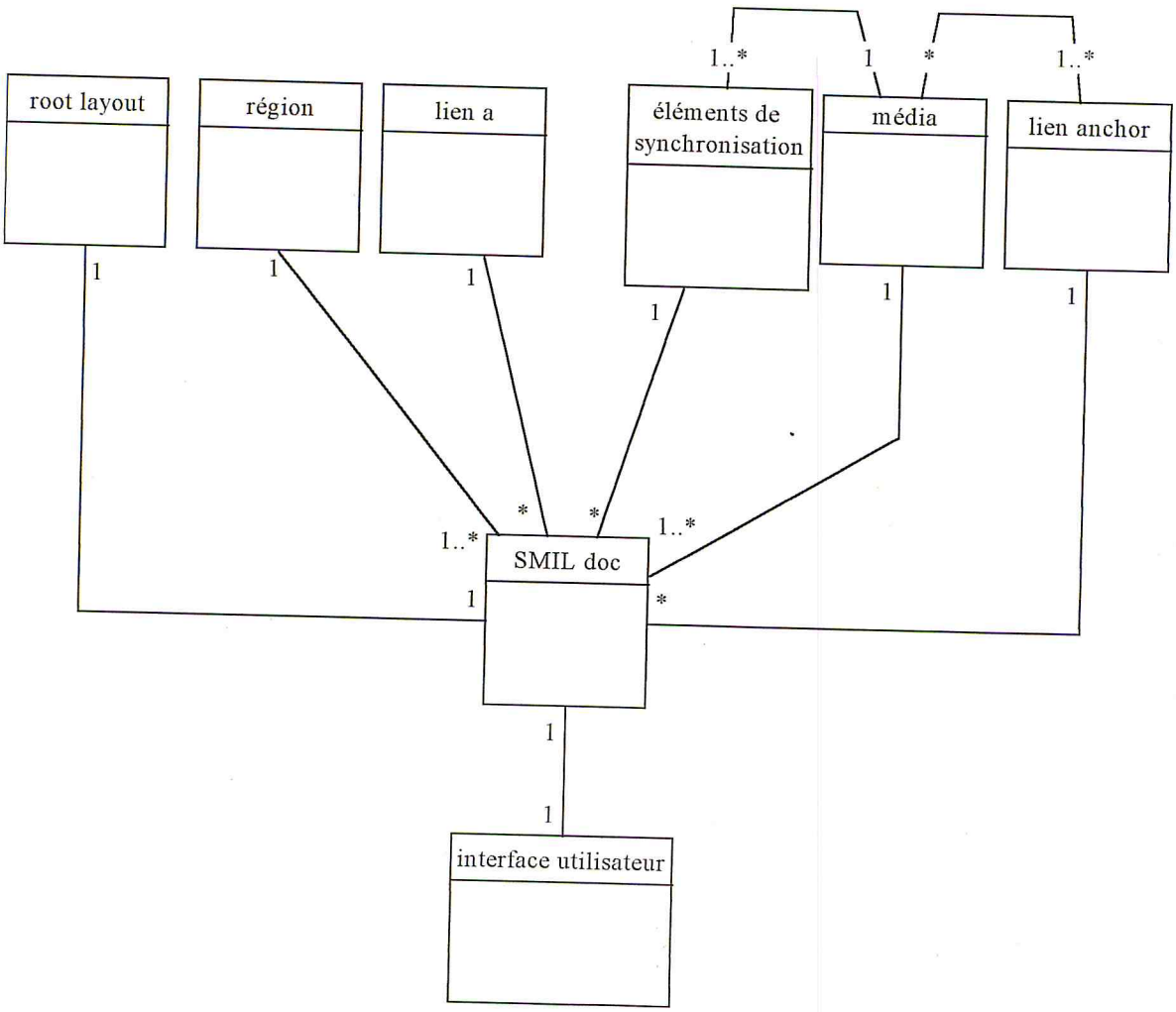


Figure 3.39 : *Diagramme de classes correspondant à notre éditeur*

3.6 Implémentation

Dans cette phase nous exprimons la partie implémentation de notre projet, dont il s'agit d'implémenter la solution évoquée dans la partie conception. Autrement dit, l'implémentation est une phase aux cours de la quelle les algorithmes (dans notre cas les diagrammes de séquences) définis dans la partie conception sont traduits dans un langage de programmation.

3.6.1 **Contexte matériel et logiciel**

L'éditeur que nous avons conçu est implémenté sous le système d'exploitation Windows. Cette implémentation est effectuée sur un PC (Personal Computer) en utilisant le langage java Builder. Nous avons choisis ce langage car il est adapté à la programmation orientée objet, aussi il est portable et permet d'implémenter des interfaces plus agréables.

3.6.2 **Implémentation des modules**

Dans cette partie, nous allons détailler les différents modules réalisés de notre architecture :

3.6.2.1 **Implémentation du module « création »**

Comme nous l'avons déjà cité dans la section (3.5.2), ce module contient trois composants: un modèle de document SMIL, un composant permettant la vérification de cohérence et une interface utilisateur. Donc, dans ce qui suit, nous allons détailler l'implémentation de ce module qui revient à détailler l'implémentation de ses composants.

3.6.2.1.1 **Implémentation du modèle de document**

Notre modèle contient les éléments qui permettent la définition des différents éléments formant un document SMIL. Pour sa (le modèle) réalisation, nous avons défini pour chaque élément le composant, une classe.

Nous avons donc :

- une classe région : pour représenter l'élément « Région » du modèle.
- une classe média : pour représenter l'élément « Média » du modèle, ce type contient une liste de lien « anchor », car en SMIL, ce type de lien doit être défini par rapport à un objet média source, et que ce dernier peut avoir plusieurs liens qui lui sont associés. Nous définissons alors cette liste de liens.

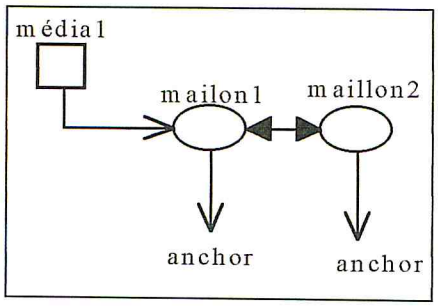


Figure 3.40 : *Représentation de la classe média*

- une classe « élément de synchronisation » : elle représente l'élément « Elément de synchronisation » du modèle, elle définit les objets qui sont soit en parallèles, soit en séquentiels, de ce fait, nous définissons dans cette classe une liste contenant ces objets, ces derniers peuvent être soit des objets de classe « média » ou des objets de la classe « élément de synchronisation » elle-même.

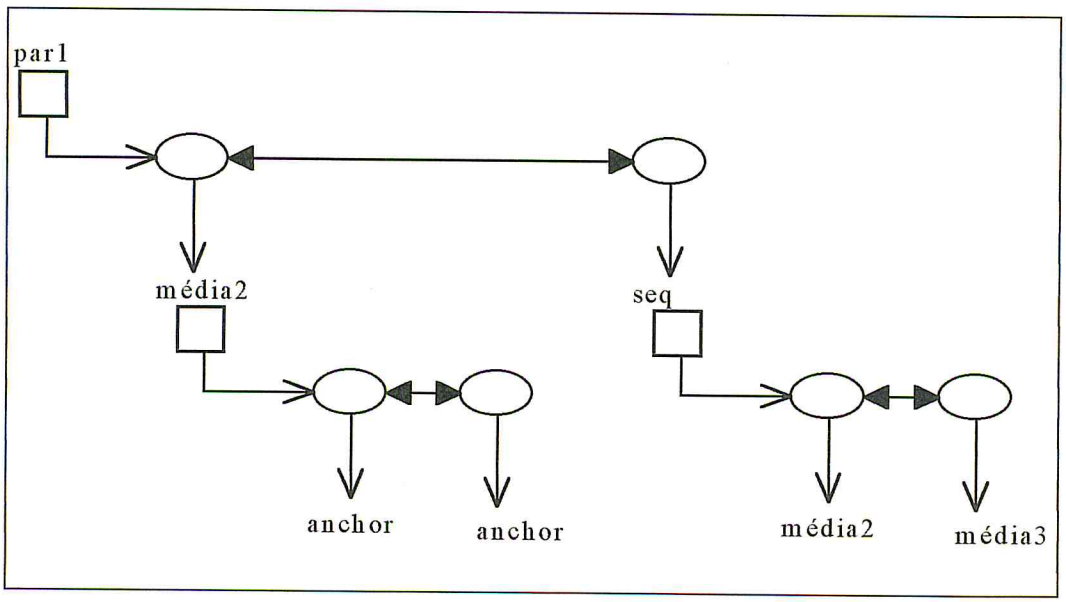


Figure 3.41 : *Représentation de la classe « élément de synchronisation ».*

- une classe « lien a » et une classe « lien anchor » : puisque SMIL utilise deux genres de liens : 'a' et 'anchor' pour représenter l'aspect hypermédia d'un document, nous définissons deux types de liens : « lien a » et « lien anchor », correspondant à l'élément « Lien » de notre modèle.

Pour stocker les occurrences des objets des différentes classes citées au dessus, nous avons utilisé une structure de liste doublement chaînée que nous nommons « element_list »,dont chaque maillon de cette liste référence un seul objet d'une seule classe parmi les classes citées auparavant. Nous définissons alors le maillon comme suit :

```
maillon { media m ;
          region r ;
          ES es ;
          Lien a a ;
          Lien anchor anch ;
          Maillon suivant ;
          Maillon précédent;
          };
```

Et voici la représentation de la liste « element_list » :

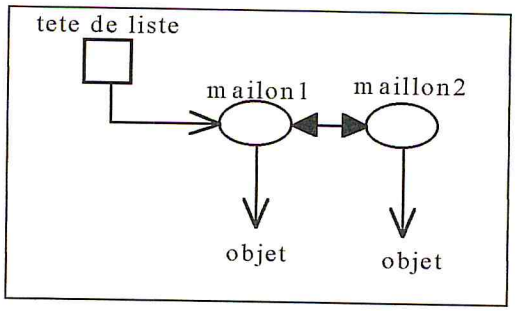


Figure 3.42 : *Représentation de la liste « element_list »*

Pour représenter les différents aspects du document multimédia, nous avons dans notre modèle l'élément SmilDoc. Ce dernier est une classe regroupant un objet de type « root-layout » qui définit l'élément « Root-Layout » du model, ainsi qu'une liste de régions « Region_list » pour représenter l'aspect spatial du document. Nous avons utilisé aussi une liste « Media_list » représentant la liste des occurrences des objets médias utilisés par l'auteur, et deux autres listes « List_anchor » et « List_a » qui définissent respectivement les liens de type 'anchor' et de type 'a'.

Pour établir la synchronisation entre les différents objets médias, nous avons utilisé des objets de classe « élément de synchronisation » en plus d'une liste nommée «Body» qui définit le scénario temporel.

Les différentes listes que nous avons cité dans SmilDoc, sont toutes de type « element_list ».

Nous avons défini dans SmilDoc, tous les aspects représentant notre modèle de document. Les actions effectuées par l'auteur reviennent à faire des ajouts/suppressions

dans les différentes listes composant SmilDoc; nous définissons ainsi les fonctions suivantes :

- Ajouter une région : correspond à l'ajout d'une région, et a comme valeur de retour un objet de classe « région ».


```
Ajouter une région ( )
  Début
    Création d'une nouvelle région ;
    M-a-j des attributs (identifiant de région, top, left ) ;
    Encapsulation de la nouvelle région dans un nouveau maillon ;
    Ajouter un maillon (le nouveau maillon) dans Region_list ;
    Retourner la nouvelle région ;
  Fin
```

- Ajouter un média (l'emplacement, le chemin, le *type,* la région associée) : le paramètre 'emplacement' désigne l'endroit dans le « body » où l'on ajoute le nouveau objet de classe « Media », le paramètre 'type' représente le type de l'objet média (texte, image…etc.), et le 'chemin' désigne l'emplacement mémoire du média, la 'région associée' est l'emplacement spatial du nouveau média dans le document. Cette fonction retourne un objet de classe « média »

```
Ajouter un média (emplacement, type, région associer, chemin)
  Début
    Création d'un nouveau média ;
    M-a-j des attributs (identifiant de média, type, région associer, chemin) ;
    Encapsulation du nouveau média dans un nouveau maillon mail 1;
    Ajouter (mail 1) dans Media_list ;
    Encapsulation du nouveau média dans un nouveau maillon mail 2;
    Insérer un maillon (mail2, emplacement) ; // insérer mail 2 dans body
    Retourner le nouveau média ;
  Fin
```

- Ajouter un élément de synchronisation (emplacement, type) : 'emplacement' a la même signification que dans Ajouter un média (…).
  'type' désigne le type de l'élément si c'est un élément *par* ou *seq* par rapport à SMIL.

Ajouter un élément de synchronisation (emplacement, type)
Début
Création nouveau élément de synchronisation ;
M-a-j des attributs (identifiant de l'élément de synchronisation, type);
Encapsulation du nouvel élément de synchronisation dans un nouveau maillon;
Insérer un maillon (le nouveau maillon) ; // insérer dans body
Retourner le nouvel élément de synchronisation ;
Fin

- Ajouter lien anchor (emplacement, destination, identifiant) : 'emplacement' a la même signification que dans Ajouter un média (…).
'destination' désigne l'objet de destination quand le lien sera activé (ex :un média ou un document SMIL), 'identifiant' est l'identifiant du lien.

Ajouter un lien anchor (identifiant du lien, destination, emplacement)
Début
Création nouveau lien anchor;
M-a-j des attributs (identifiant du lien, destination, emplacement);
Encapsulation du nouveau lien anchor dans un nouveau maillon mail 1;
Ajouter (mail 1) dans Anchor_list ;
Encapsulation du nouveau lien anchor dans un nouveau maillon mail 2;
Insérer un maillon (mail2, emplacement) ; // insérer mail 2 dans body
Retourner le nouveau média ;
Fin

Toutes ces fonctions reviennent toujours à des ajouts, des insertions dans des listes de type « element_list ». Nous définissons maintenant ses différentes fonctions :

- Ajouter un maillon (mail): ajouter un maillon dans une liste de type «element_list». Le paramètre « mail » représente le maillon à ajouter

```
Insérer un maillon (emplacement, mail)
Emplacement non trouvé ; mail non inséré;
Début
  Si (emplacement non défini) Alors Ajouter (mail) ;
   sinon
   Début
    temp=tête de liste ;
    Tantque ((temp non vide) et (emplacement non trouvé) et (mail non inséré))
     Faire
     Si (temp correspond à l'emplacement )
       Alors Début
               emplacement trouvé ;
               sortir de la boucle tant que;
             Fin
       sinon
         Si (temp contient un objet de type 'élément de synchronisation' ou de
                                                                  type 'Media' )
          Alors
           Si (liste de l'objet de type 'élément de synchronisation' ou 'Media'
                                                                      non vide )
            Alors
            Début
              Insérer un maillon (emplacement, mail) dans cette liste ;
               Si ( mail inséré)
                alors Début
                        mail inséré ;
                        arrêter la boucle;
                      Fin
                 sinon temp passe au suivant ;
               Fin si
              Fin
             sinon temp passe au suivant ;
           Fin si
          sinon temp passe au suivant ;
        Fin si
     Fin si
    Fait
   Fin
  Fin si
  Si (emplacement trouvé)
    Alors Début
            si (emplacement correspond à avant) alors insertion avant ;
            si (emplacement correspond à après) alors insertion après ;
            si (emplacement correspond à ajouter  dans la liste)
               Alors  Ajouter (mail) dans la liste de temp;
          retourner mail inséré ;
         Fin
   sinon si (mail non inséré) alors retourner mail non inséré ;
 Fin si
Fin
```

```
Ajouter_maillon (maillon mail)
Début
  si (liste vide) alors tête de liste=mail;
    sinon
      Début
        temp=tête de liste ;
        tantque (temp non vide)
          pre=temp ;
          temp=suivant de temp ;
        Fin tan que
        suivant de pre devient mail ;
        précédant de mail devient pre ;
      Fin
  Fin
Fin
```

- Insérer un maillon (emplacement, mail) : cette fonction permet d'insérer le maillon *mail* à l'endroit *emplacement* dans une liste de type « element_list ».

  Le paramètre « emplacement » définit deux parties :

  1 le maillon auquel nous nous référençons pour insérer le nouveau maillon.

  2 la position à laquelle nous insérons le nouveau maillon, si avant, après ou dans la liste que contient l'objet contenu dans le maillon référence, ces objets peuvent être de soit de type « élément de synchronisation » ou de type «Média».

emplacement=((maillon référence) et ((avant) ou (après) ou (dans la liste)))

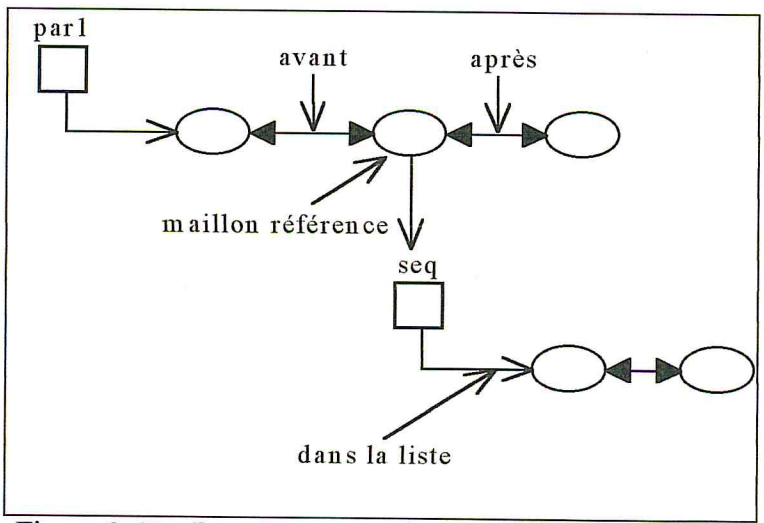


Figure 3.43 : *Représentation descriptive de l'emplacement*

3.6.2.1.2 **Implémentation de la vérification de cohérence**

Dans cette phase nous allons présenter quelques algorithmes que nous avons implémenter pour la détection des incohérences citées dans la section (3.5.2.2.1)

Cas de dépassement de la taille de fenêtre principale :

La détection de cette incohérence et possible avec l'algorithme suivant :

```
Début
  Si ((largeur d'une région + positionnement sur X de cette région) > largeur de la
                                                                  fenêtre principale)
    Alors
      Début
        Signaler à l'utilisateur par un message ;
        Rétablir les anciennes valeurs de région ;
      Fin
  Fin si

  Si ((hauteur d'une région+top de cette région) > hauteur de la fenêtre principale)
    Alors
      Début
        Signaler à l'utilisateur par un message ;
        Rétablir les anciennes valeurs de région ;
      Fin
  Fin si
Fin
```

Cas de partage d'une même région entre deux médias (non audio) parallèles :

La détection de cette incohérence est possible avec l'algorithme suivant :

```
Début

  Charger une liste des régions à associer;
  Si (liste de ce par est non vide)
    Alors temp=tête de liste ;
      Si ( temp correspond à un média non audio) Alors supprimer région associée à ce
                                                    media de la liste des régions à associer ;
      Fin si
      temp passe au suivant ;
    Fin si
Fin
```

3.6.2.2 **Implémentation du module « traduction »**

Comme il été déjà expliqué dans la partie conception, ce module contient une partie de génération de code SMIL et une autre pour la reconnaissance.

```
Génération du code ()
Début
  Si (la liste des régions non vide)
    Alors Début
            Maillon temp=tête de liste de la liste des régions ;
            Tant que (temp non nulle)
            Faire
              Donner le code de la région à laquelle correspond temp ;
              temp passe au suivant ;
            Fait
          Fin
  Fin si
  Si ( body non vide)
    Alors Début
            Maillon temp1= tête de liste de body ;
            Tant que (temp1 non nulle)
            Faire
              Si ( temp1 correspond à un objet de type Media)
                Alors  donner code de l'objet media ;
              Fin si

              Si ( temp1 correspond à un objet de type Lien a)
                Alors  donner code de l'objet Lien a ;
              Fin si

              Si (temp1 correspond à un objet de type élément de
                    synchronisation )
                Alors  donner code de l'objet élément de synchronisation ;
              Fin si
              temp1 passe au suivant ;
            Fait
          Fin
  Fin si
Fin
```

```
Donner code d'un objet Media ()
Début
  Ecrire code correspondant aux attributs de l'objet Media ;
  Si (la liste des liens 'anchor' de l'objet Media non vide)
    Alors Début
            Maillon temp= tête de la liste ;
            Tant que (temp non nulle)
            Faire
              Donner code anchor ;
              temp passe au suivant ;
            Fait
  Fin si
Fin
```

```
Donner code d'un élément de synchronisation ()
Début
  Ecrire code correspondant aux attributs de l'objet Media ;
  Si (la liste des objets contenus dans cet  élément de synchronisation est non vide )
    Alors
      Début
        Maillon temp2=tête de la liste ;
        Tant que (temp2 non nulle)
        Faire
          Si (temp2 correspond à un objet de type Media)
          | Alors   donner code de l'objet media ;
          Fin si
          Si (temp2 correspond à un objet de type Lien a)
          | Alors   donner code de l'objet Lien a ;
          Fin si
          Si (temp2 correspond à un objet de type élément de synchronisation)
          | Alors   donner code de l'objet élément de synchronisation ;
          Fin si
          temp2 passe au suivant ;
        Fait
      Fin
  Fin si
Fin
```

3.6.2.3 **Implémentation du module « visualisation »**

Pour ce module nous n'avons implémenté que la partie de la sauvegarde, alors que le lecteur reste un objectif à atteindre.

3.6.2.3.1 .**La sauvegarde**

Dans cette phase, nous présentons l'algorithme qui nous a permit de sauvegarder le fichier créé sous l'extension *.smi, c'est-à-dire sous le format SMIL.

```
Début
  Ouverture d'un fichier texte en écriture ;
  Transfert du code généré vers le fichier ;
  Enregistrement du fichier sous le nom X.smi ;

Fin
```

3.6.3 **Interface Utilisateur**

Notre éditeur comprend une interface graphique afin de permettre à tout utilisateur, quel que soit son niveau et sa pratique en informatique, de pouvoir créer, modifier et visualiser une présentation multimédia. Comme le montre la fenêtre suivante l'éditeur est modélisé comme suit :

- un récipient contenant cinq interfaces graphiques :
    1. Une interface pour la vue structurelle.
    2. Une interface pour la vue timline.
    3. Une interface pour la vue source.
    4. Une interface pour la vue spatiale
    5. Une interface pour la vue attribut.
- un menu.
- une barre standard.

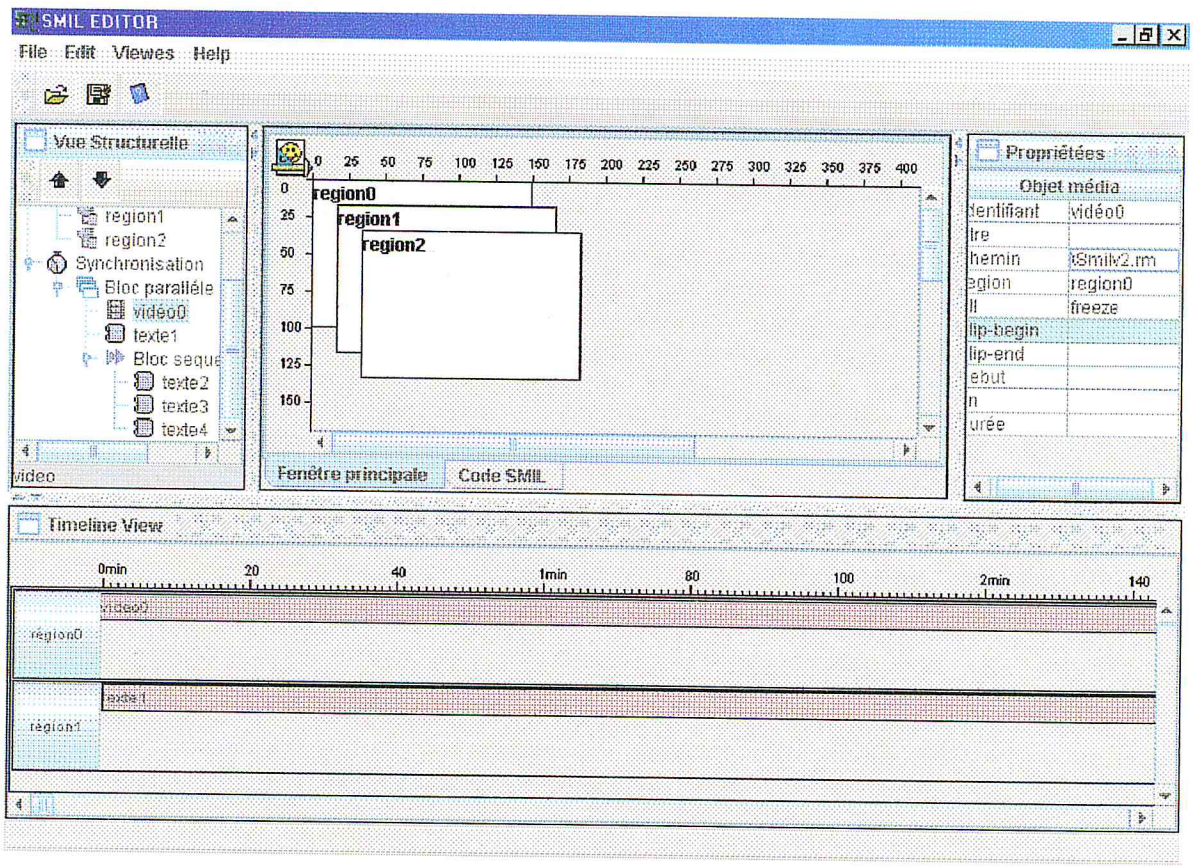


Figure 3.44 : *Vue globale de notre éditeur SMIL*

- Récipient

Ce récipient comme son nom l'indique, représente l'espace de travail où les fenêtres s'ouvrent, il occupe la zone centrale de l'application. Autrement dit, c'est le support d'affichage des différentes fenêtres (vues).

- Menu

Le menu est conçu pour que l'utilisateur manipule les diverses fonctionnalités du système. Notre menu, comme le présente la figure précédente est composée de « quatre » éléments qui sont : *Fichier*, *Edition*, Affichage et A propos.

- Barre standard

Cette barre est conçue pour faciliter quelques opérations à l'utilisation, comme la sauvegarde, l'ouverture d'un fichier pour une mise à jour …etc.

Notre éditeur se compose des vues suivantes :

La vue structurelle

L'intérêt de cette vue est de donner à l'utilisateur une représentation globale de la structure de son document, sous forme d'un arbre similaire à celui utilisé par les gestionnaires de fichiers des systèmes d'exploitation, où chaque nœud de la structure peut être déployé.

Cet arbre, qui sera affiché dans l'interface graphique, permet une navigation rapide au sein de la structure du document et visualise l'ensemble des éléments.

Nous construisons cette vue en se basant sur l'organisation hiérarchique des documents SMIL, à savoir une partie spatiale et une partie temporelle. La racine de l'arbre est définie par l'élément **«smil»** du document, de cette racine partent trois branches représentant respectivement, l'emplacement spatial, la synchronisation et les liens.

L'élément **«Fenêtre principale»** est le premier nœud de la structure hiérarchique spatiale, ce nœud contient un seul fils de type **«région»**, ce dernier ne contient aucun fils.

L'élément **«synchronisation»** est le premier nœud de la structure hiérarchique temporelle. En effet, la sémantique de cet élément est comme celle de l'élément **«seq»**, c'est à dire une mise en séquence de ses fils, un document peut donc très bien n'avoir que cet élément pour spécifier son comportement temporel. L'élément **«synchronisation»** peut avoir des fils qui sont à leur tour des nœuds, le cas des éléments **seq** et **par**, ou des feuilles, le cas des différents médias.

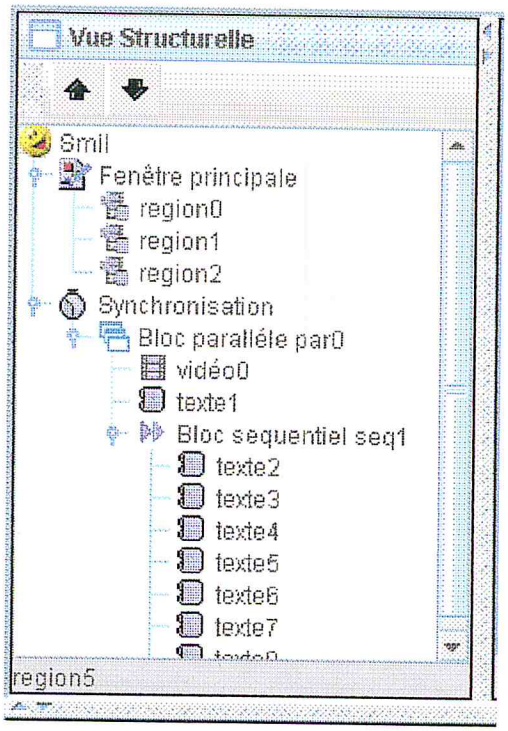


Figure 3.45 : *L'interface graphique de la vue structurelle*

Dans la vue structurelle nous pouvons effectuer les opérations suivantes :

Ajout/Suppression d'un bloc séquentiel.

Ajout/Suppression d'un bloc parallèle.

Ajout/Suppression d'un média.

Ajout/Suppression d'un lien.

Changement de l'emplacement temporel d'un média.

Changement de l'emplacement d'un bloc séquentiel dans l'arbre.

Changement de l'emplacement d'un bloc parallèle dans l'arbre.

Pour effectuer ces opérations, il suffit de sélectionner un nœud dans l'arbre par un simple clique de la souris, nous avons ajouté deux boutons qui permettent le déplacement d'un nœud dans l'arbre.

La figure suivante le montre :

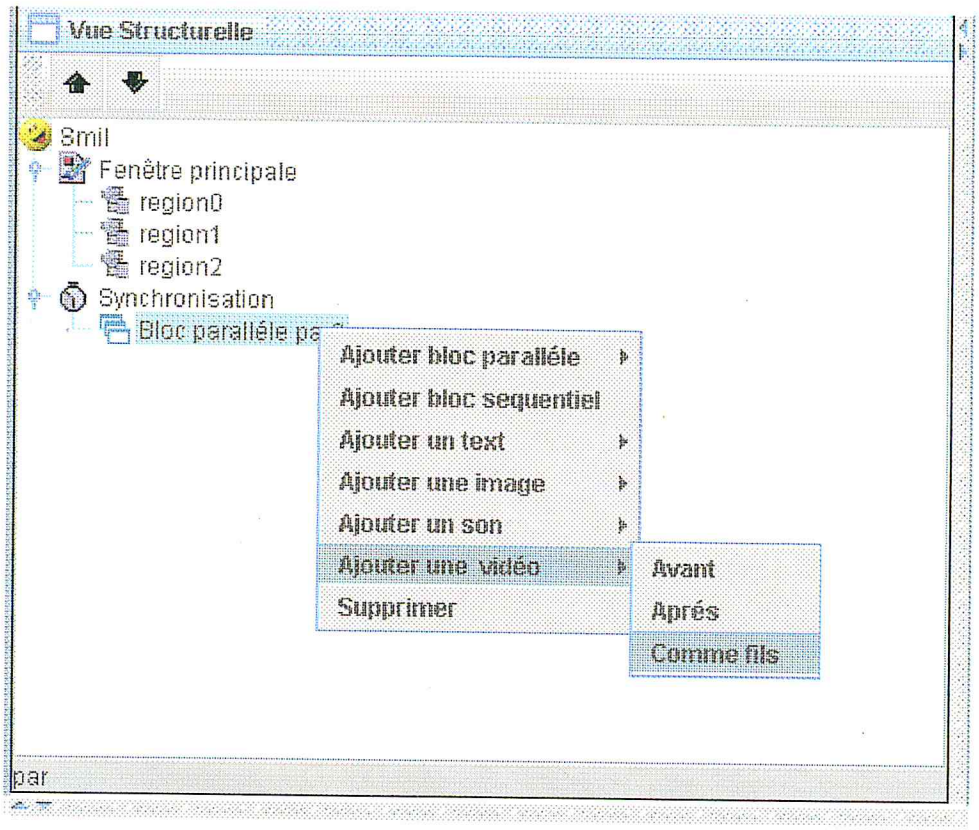


Figure 3.46 : *Le menu des opérations possibles sur un nœud (exemple bloc parallèle).*

Description de la vue temporelle

Cette vue permet à l'utilisateur d'affiner le scénario temporel de son document. L'intérêt de ce type d'interface est d'offrir une représentation graphique du déroulement du document par rapport à un axe de temps. Autrement dit, donner à l'auteur une meilleure perception de l'enchaînement temporel des objets multimédias constituant son document, ainsi que l'ensemble des médias appartenant à la même région, puisque nous avons spécifié pour chaque média la région qui lui est affectée.

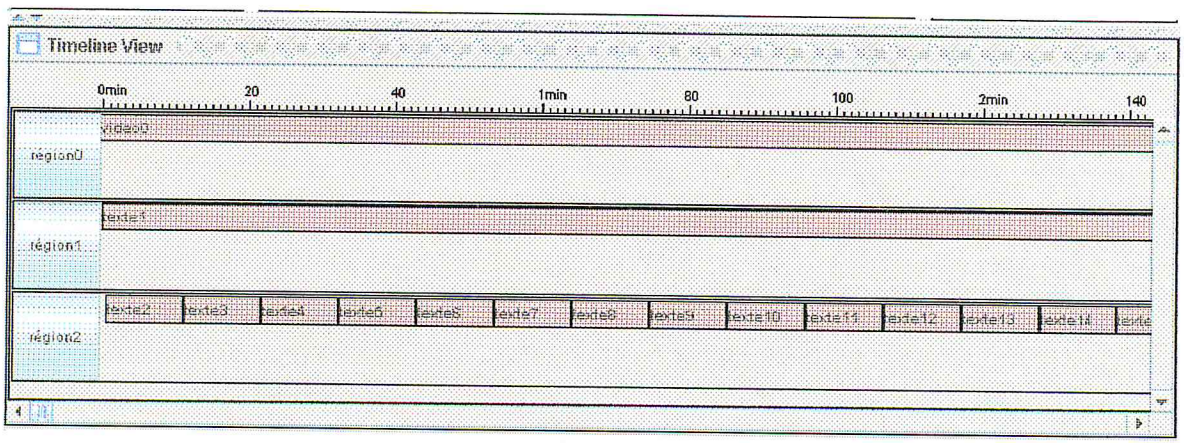


Figure 3.47 : *L'interface graphique de la vue temporelle.*

La vue spatiale

Cette zone graphique permet d'afficher l'ensemble des régions définies par l'auteur dans l'entête du document SMIL. La modification de la taille et de l'emplacement, ainsi que la suppression de ces régions peuvent s'effectuer directement dans cette vue.

Le langage SMIL a une particularité pour ce qui concerne l'aspect spatial du document, c'est qu'il a la possibilité d'affecter à une même région plusieurs médias. En effet, la définition des régions se fait indépendamment des médias dans l'entête du document, alors que les affectations se font dans le corps.

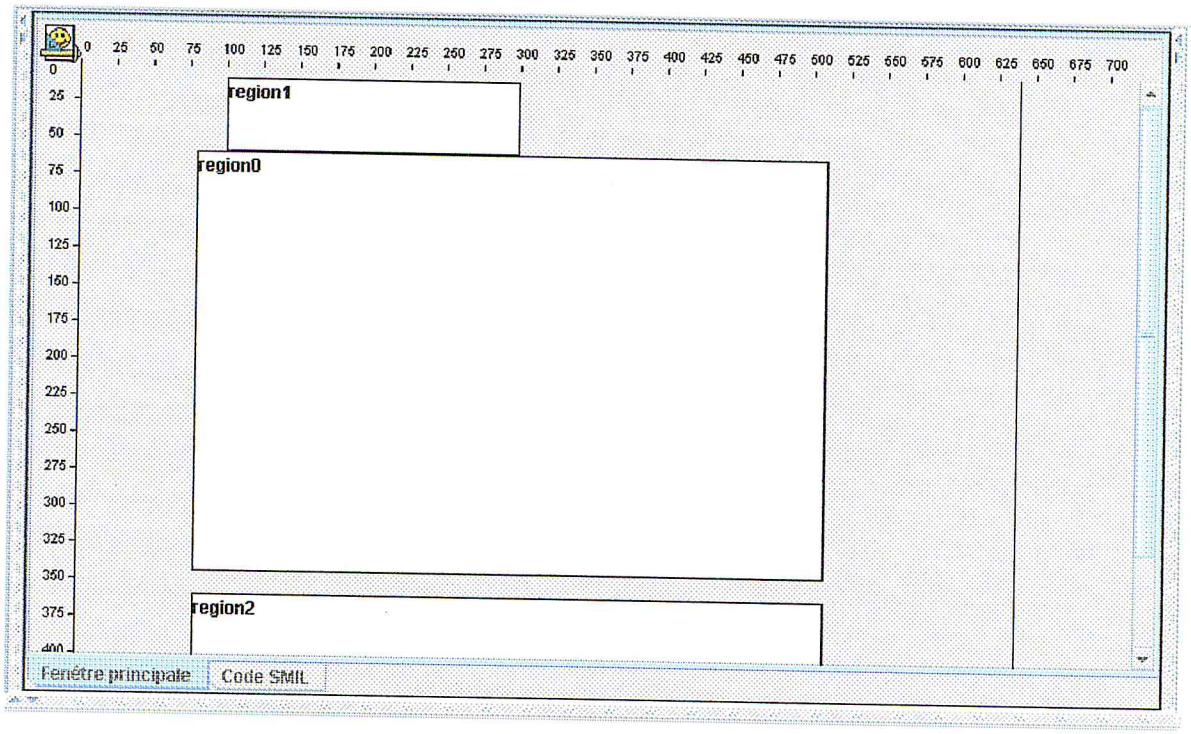


Figure 3.48 : *L'interface graphique de la vue spatiale.*

La vue source

Cette vue sert à éditer le code SMIL du document à générer. Les modifications apportées par les autres vues seront reflétées directement dans le code visionné dans la vue source. Par exemple, lors de l'insertion d'un média, le code associé sera inséré dans le bon emplacement c'est-à-dire dans la ligne correspondante.

Le choix d'ajouter cette vue à l'ensemble des vues se justifie pour que les auteurs inexpérimentés en langage SMIL puissent en tirer profit dans le sens de l'apprentissage et

le développement de leurs connaissances. Ainsi pour chaque modification, ils aperçoivent la modification apportée au code au fur et à mesure qu'ils manipulent les autres vues.
Le contenu de cette vue sera sauvegardé dans un fichier ayant l'extension **.smi* ou **.smil* . Une fois sauvegardé, l'auteur pourra jouer son document avec un lecteur à travers le bouton associé dans la barre d'outil.

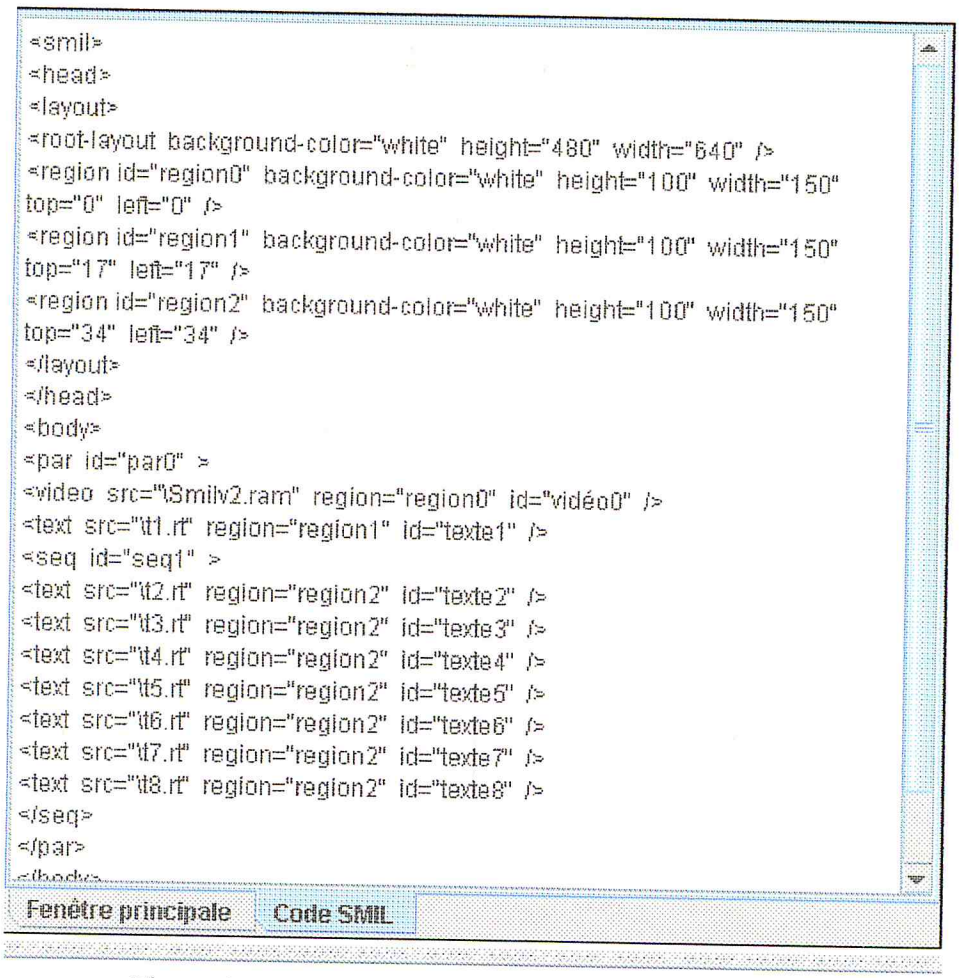


Figure 3.49 : *L'interface graphique de la vue source.*

Description de la vue attribut

Cette vue est composée de plusieurs champs d'édition pour la saisie des valeurs des attributs spatiaux ou temporels de l'élément sélectionné dans la vue structurelle. Par exemple, si on sélectionne l'élément **seq**, l'ensemble de ses attributs ainsi que leurs valeurs seront visionnés dans la vue attribut.

Notons que dans le langage SMIL certains attributs ont des valeurs bien déterminées. Par exemple l'attribut ***fit*** de l'élément **région** peut prendre comme valeur :*fill*, *scroll*,

hidden, *meet* ou *slice*, pour cela, nous proposons dans la vue attribut des listes de choix comportant ces valeurs.

Nous pouvons également vérifier l'unicité des identificateurs utilisés.

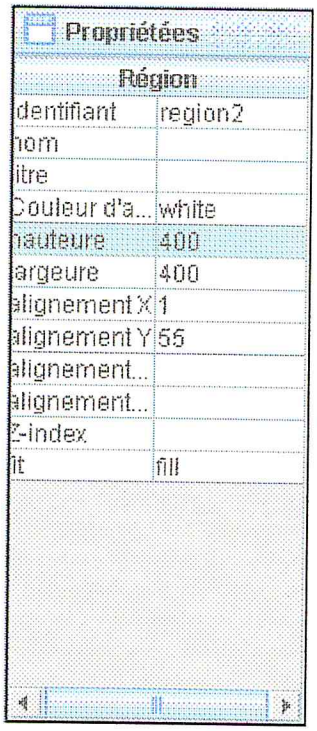


Figure 3.50 : *L'interface graphique de la vue attribut (exemple attributs de l'élément Région)*

3.7 Tests et validation

Dans cette partie nous allons tester quelques cas d'utilisation, qui se résument en la création et la visualisation d'une présentation multimédia. Au cours de cette phase, nous essayons, en utilisant notre éditeur, de créer une présentation (tout en suivant les scénarios décries par les diagrammes de séquence dans la section (3.4.1.1.4) permettant la création d'une présentation) dont le contenu est sur la deuxième version de SMIL, en quelque sorte nous allons créer un cours assisté par ordinateur (CAO). Cet exemple, que nous avons adopté, consiste en la synchronisation entre deux médias divers: une vidéo qui permet de donner une explication orale, et du texte qui permet de donner l'explication écrite (sous titrage).

3.7.1 **Cas d'utilisation création d'une présentation**

Pour la création d'une présentation multimédia synchrone, l'utilisateur doit passer par les étapes suivantes :

1- la demande de la création d'une nouvelle présentation, en demandant le chargement d'un nouvel espace de travail. De cet effet, le système demande de spécifier l'emplacement mémoire du nouveau fichier SMIL. Ceci est représenté dans la figure suivante :

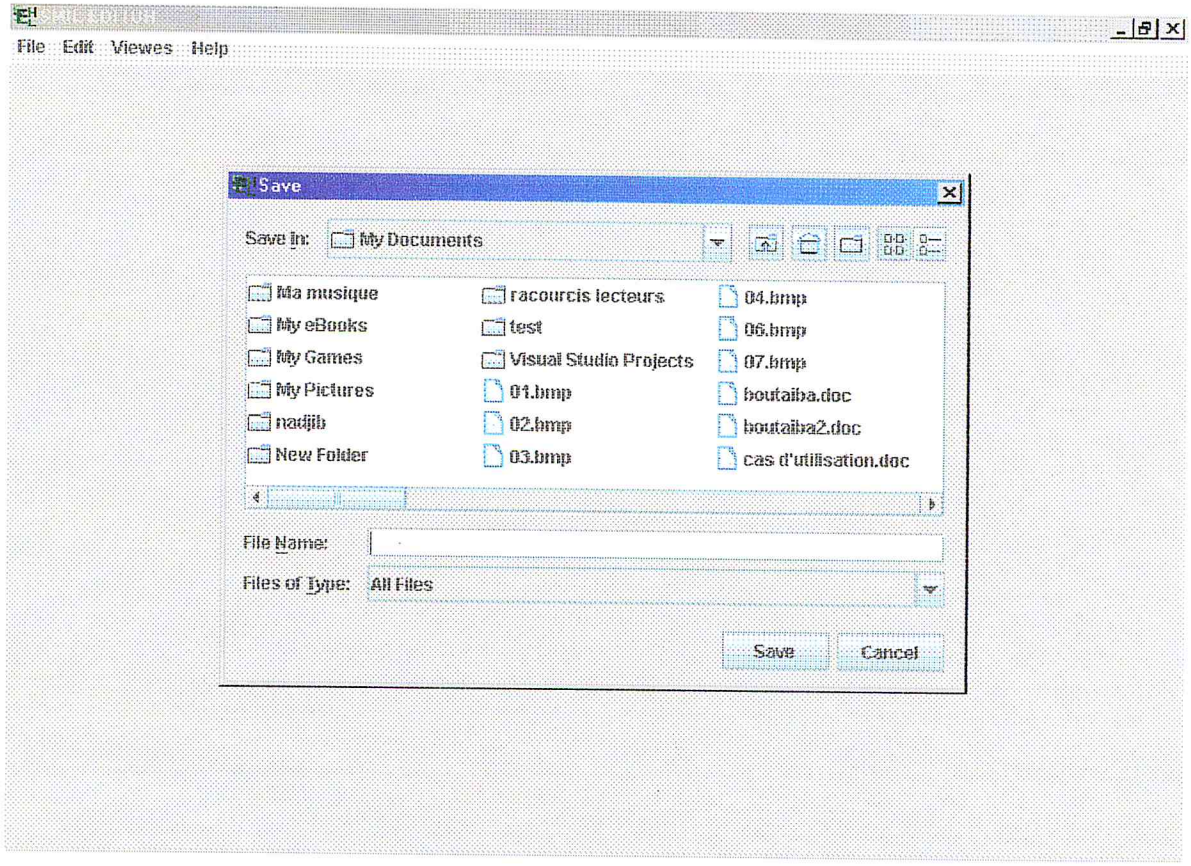


Figure 3.51 : *Enregistrement du nouveau fichier SMIL*

2- l'utilisateur ajoute des régions dans lesquelles seront affichés les médias, cet ajout se fait d'une manière simple, ce qui est représenté dans la figure suivante :

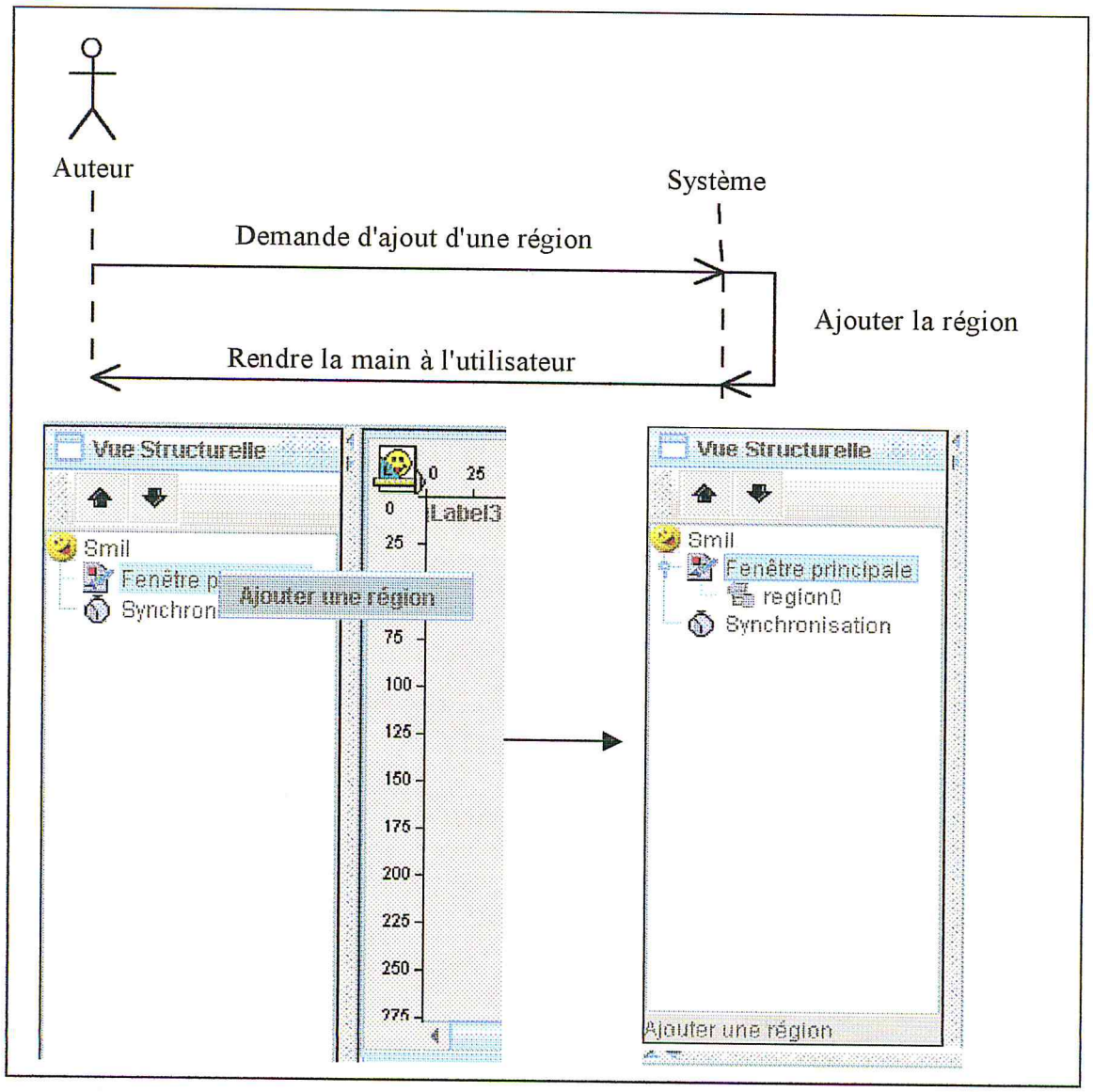


Figure 3.52 : *validation du cas d'utilisation ajout d'une région.*

3- Pour établir la synchronisation des médias, l'utilisateur est obligé d'ajouter des éléments de synchronisation, que ce soit un bloc parallèle ou un bloc séquentiel, ces derniers vont contenir, respectivement, des médias qui s'exécutent en parallèle et d'autres qui s'exécutent en séquentiel. Cet ajout se fait de la manière suivante :

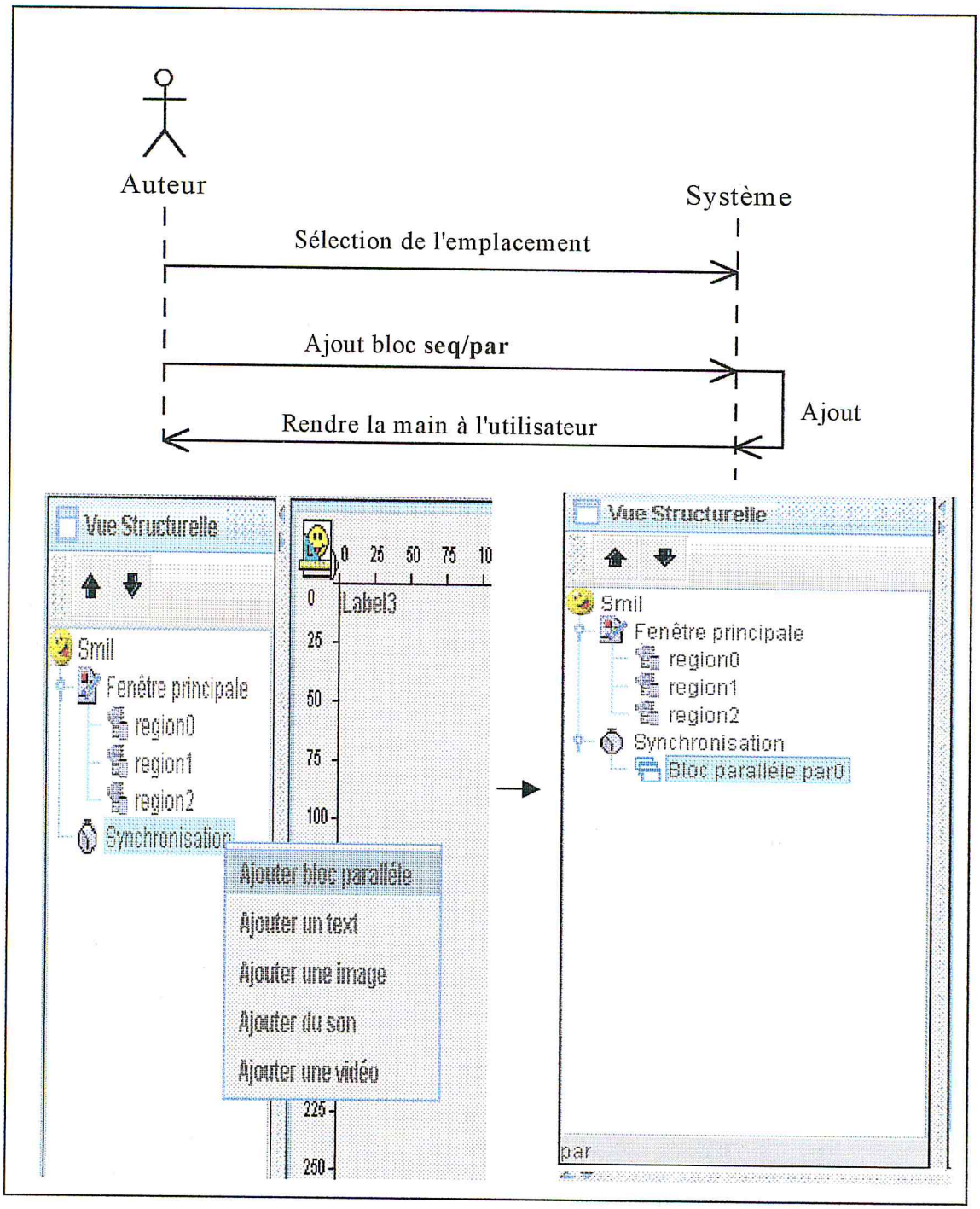


Figure 3.53 : *Validation du cas d'utilisation ajout d'un élément de synchronisation.*

4- l'utilisateur fait des ajouts de médias en spécifiant leurs chemins et leurs zones d'apparition (régions), ainsi que leurs emplacements dans le scénario temporel (dans l'arbre de synchronisation). La spécification du chemin et de la région se font par l'utilisation des boites de dialogues. Ce cas est représenté dans la figure suivante :

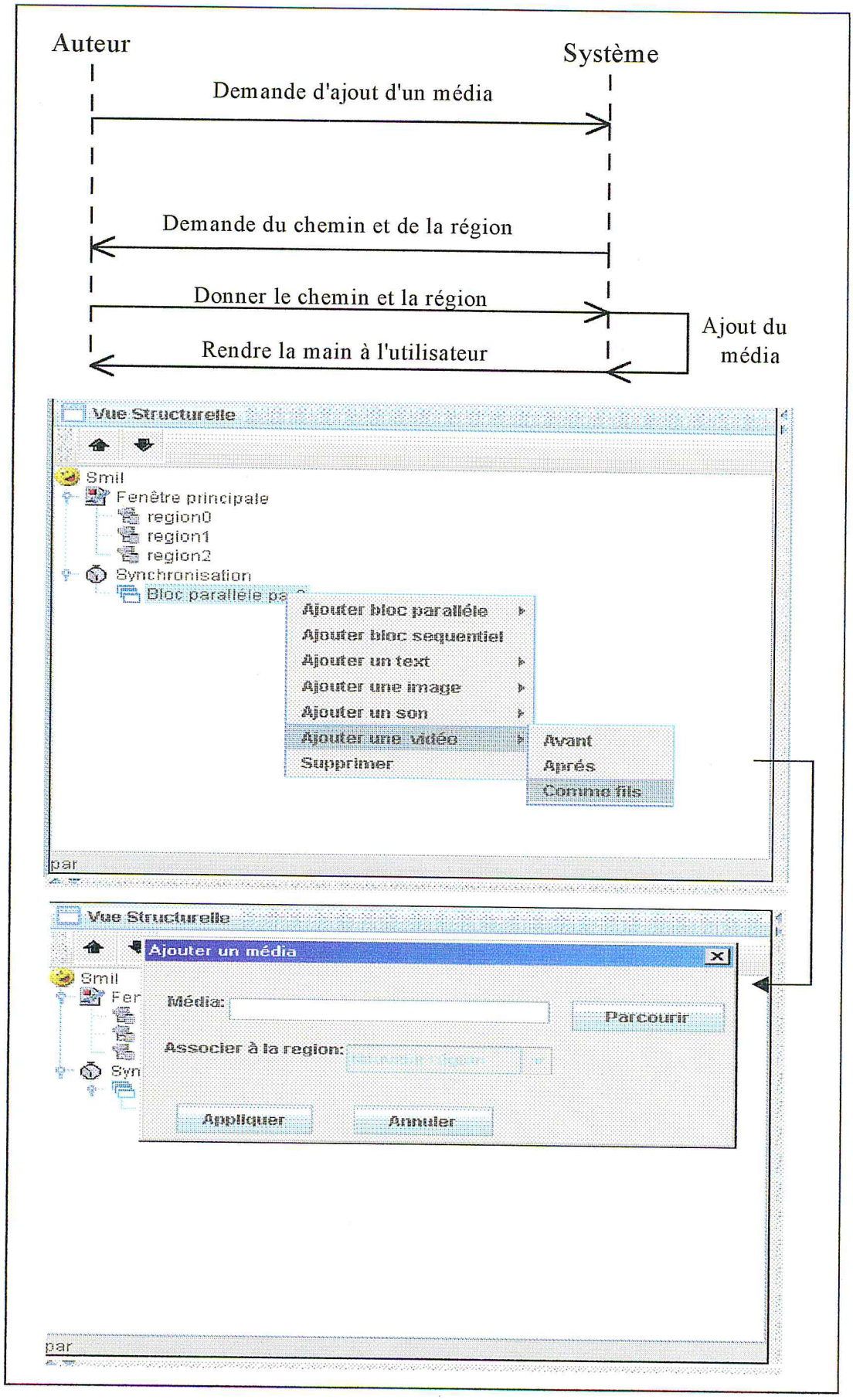


Figure 3.54 : *Validation du cas d'utilisation ajout d'un média.*

3.7.2 **Cas d'utilisation visualisation de la présentation.**

Après toutes les étapes citées au-dessus, l'utilisateur effectue l'opération de sauvegarde, puis il lancera la lecture du fichier résultat (fichier SMIL) à l'aide d'un lecteur externe. Ce cas est représenté dans la figure suivante :

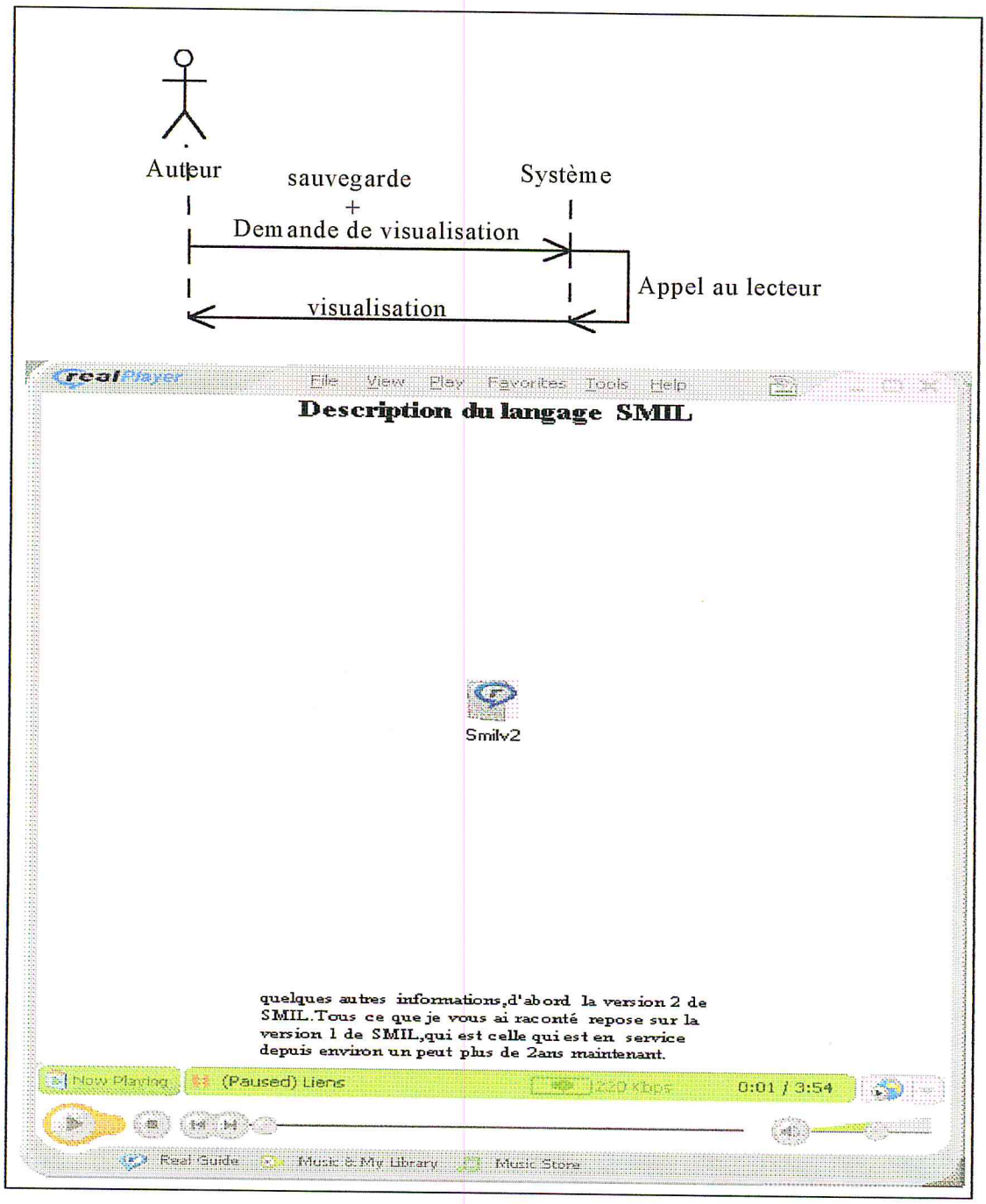


Figure 3.55 : *Validation du cas d'utilisation sauvegarde et lecture.*

Le modèle correspondant à cette exemple est le suivant :

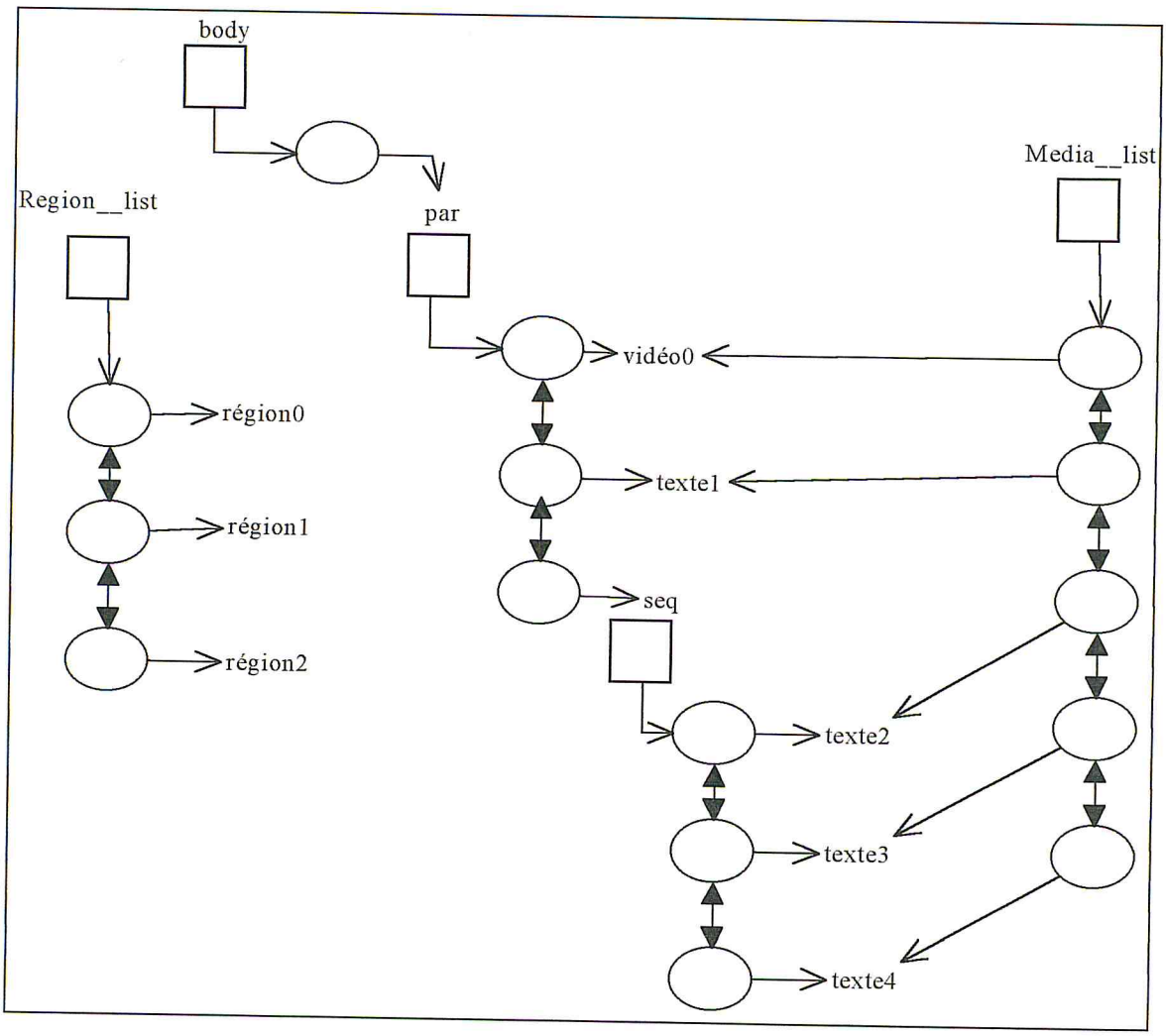


Figure 3.56 : *schéma du modèle de document correspondant à l'exemple.*

3.8 Conclusion

Nous avons présenté, au cours de ce chapitre, la mise en œuvre d'un outil d'édition de documents multimédias synchronisés sous le format SMIL. Pour cela, nous avons utilisé une modélisation et une conception orienté objet.

Nous avons commencé la démarche de développement par l'identification des différents besoins d'un utilisateur face à un éditeur multimédia, en se basant sur ces besoins, nous avons défini l'architecture de notre éditeur qui peut répondre à ces exigences. Cette architecture est décomposée en trois parties : la partie création permettant la création d'un document multimédia, et la vérification de la cohérence entre les médias appartenant à ce document , ainsi que la partie traduction dans laquelle l'éditeur effectue la génération du code SMIL correspondant au document créé. La dernière partie qui est la partie visualisation permet la lecture du fichier résultat par un lecteur externe.

Notre système d'édition permet à l'utilisateur de créer son document multimédia de façon incrémentale tout en apercevant les modifications apportées à travers l'interface graphique. Pour cela, la conception de notre éditeur a été réalisée en se basant sur les différents besoins d'un environnement d'édition, et ceux de l'édition de documents multimédias synchronisés cités au premier chapitre, auxquels nous avons essayé de répondre.

Conclusion générale

Au terme de notre étude, nous devons de signaler que notre travail repose essentiellement sur la conception et la réalisation d'un système auteur pour l'édition de documents multimédias au format SMIL, nommé HLE, qui intègre des éléments de type texte, image, audio et vidéo. Dans ce type de documents, l'intégration de la dimension temporelle est un élément indispensable.

Parmi les domaines qui tirent profit de cet outil, nous pouvons citer deux secteurs représentatifs mais non exclusifs : l'enseignement assisté par ordinateur (EAO) et la médecine.

L'EAO peut tirer parti des caractéristiques des différents médias pour réaliser des supports pédagogiques qui soient d'une part plus attractifs grâce aux images, aux animations et au son, et d'autre part plus interactifs et adaptables aux élèves grâce aux fonctions de navigation hypermédia. De même les données issues de l'imagerie médicale, comme les images par rayons X, par résonance magnétique ou par échographie, peuvent être exploitées sous forme numérique et intégrées à des données textuelles (par exemple les informations relatives au patient), et sonores (les commentaires du médecin), pour former des documents multimédias médicaux qui peuvent être consultés à distance par les médecins ou le patient lui-même.

Nous avons eu la chance d'être intégré à un projet important faisant appel à des technologies novatrices. Travaillant sur ce projet, nous avons rencontré quelques difficultés, vue que UML a des notations multiples, et que le langage SMIL a été récemment créé ce qui fait que la communauté qui le maîtrise est limitée.

Cependant, ces difficultés nous ont permis d'acquérir un esprit de synthèse et de toucher à plusieurs domaines, tel que SMIL, le multimédia, l'édition de documents multimédias,...etc.

Compte tenu des progrès en la matière, nous restons optimistes quand aux perspectives multiples qui pourront être développés à partir de notre travail. A titre indicatif, l'intégration de l'élément ***switch*** qui a été décrit dans le chapitre 2 et l'implémentation de la deuxième version du langage SMIL (SMIL 2.0), ainsi que l'intégration d'un lecteur et d'un éditeur de texte propres à notre éditeur permettant respectivement la lecture du fichier SMIL créé, et la création des fichiers Rich Texte (*.rt).

Enfin, nous espérons que ce travail représentera une plate forme féconde pour toute étude éventuelle que les étudiants tenteront de faire dans le domaine multimédia

Références

[1] F. Rousseau, *« Présentations Multimédia Synchronisées pour le WWW »*, Thèse en Informatique : Systèmes et Communication pour le grade de DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE, janvier 1999.

[2] S. L'haire, *« L 'Enseignement Assisté par Ordinateur et le Traitement Automatique du Langage Naturel »*, Mémoire de diplôme d'études supérieures en : Linguistique Informatique. Université de Genève, février 2000.

[3] Le petit Larousse 2002.

[4] *LE MULTIMEDIA* -http://membres.lycos.fr/apd/PC.HTML

[5] N. Layaïda, *« Madeus : système d'édition et de présentation de documents structurés multimédia »*, Thèse en Informatique préparée dans le cadre du projet Opéra, pour obtenir le titre de Docteur de l'Université Joseph Fourier – Grenoble 1, juin 1997.

[6] F. Duluc[1] et C. Roisin[2] et L. Tardif[2] et L. Villard[2] *« Un système multimédia complet pour la documentation technique aéronautique ».*
[1]Aérospatiale Matra Airbus et IRIT
[2]Unité de Recherche Rhône-Alpes, Projet OPERA

[7] F. Hammad et L. Maaradji, *« Système auteur pour l'édition de documents multimédias au format SMIL »*, Thèse en Informatique : Software pour le titre d'ingénieur en informatique, Universite des sciences et de la technologie Houari Boumediene, octobre 2002.

[8] A. Belaid, « *Nouveau langage introduit par le Consortium W3C pour gérer le multimédia Il s'appelle SMIL : Synchronized Multimedia Integration Language.* »

[9] D. Courtaud, « *SMIL (Synchronized Multimedia Integration language)* », DESS Ingénierie Documentaire et Multimédia, Module M4A.1998.

[10] P. Hoschka W3C, «*Know Errors in SMIL 1.0 Specification*». http ://www.w3c.org/Audio Video/SMIL/errata/July 1998

[11] GRiNS Editor for SMIL version 1.5 –GRiNS/SMIL Tutorial Guide, http://www.oratrix.com/GRins/,June,2000

[12] *JUST SMIL* –http://www.allhtml.com/smil/justsmil.html

[13] « *Synchronized Multimedia Integration Language (SMIL) 1.0 Specification*», W3C Recommandation, http://www.w3.org/TR/REC-smil, 15 juin 1998.

[14] A. Michard, « *XML langage et applications* » ,Eyrolles, Paris 2001.

[15] E. Rusty Harold et W. Scott Means, « *XML In a Nutshell* » édition O'REILLY ,18 rue Séguier 75006 Paris, France

[16] Sausage Software-SMILcomposer. http ://www.sausage.com.

[17] J. Bres, ateliers de génie logiciel, Masson, 1993.

[18] N. Layaïda, « *Adaptabilité :piste d'étude pour la définition d'une infrastructure d'accès au contenu multimédia pour des machines hétérogènes* », Rapport de contrat, INRIA Rhône-Alpes, Octobre 1999.

[19] P.A. Muller, « *modélisation objet avec UML* », EYROLLES, 1997.

[20] P.A. Muller, « modélisation objet avec UML », EYROLLES, 2001.

[21] P.A. Muller, « *modélisation objet avec UML* », EYROLLES, 2002.

[22] L. Somerville, le génie logiciel et ses applications, paris, 1988.

[23] F. Bernardi, Méthode d'analyse orienté objet UML, 2002.

[24] Rémy Fannader, Hervé Lerroux, UML principes de modélisation, DUNOD, 2000.

[25] Joseph Gabay, Merise vers OMT et UML, Masson 1998.

Annexes

Annexe A. Le modèle en cascade

1. Introduction

Ce modèle est décri par Royce en 1970, il a été largement employé depuis, pour la description générale des activités liées aux logiciels [19].

Le modèle «en cascade» présente un cycle de vie d'un logiciel par une suite de phases ou d'étapes (analyse, conception, implémentation, test et validation) (Figure A.1) qui s'enchaînent dans un déroulement linéaire depuis l'analyse des besoins jusqu'à la maintenance [19]. Les résultats de chaque étape sont testés, et on ne passe à l'étape suivante que s'ils sont validés.

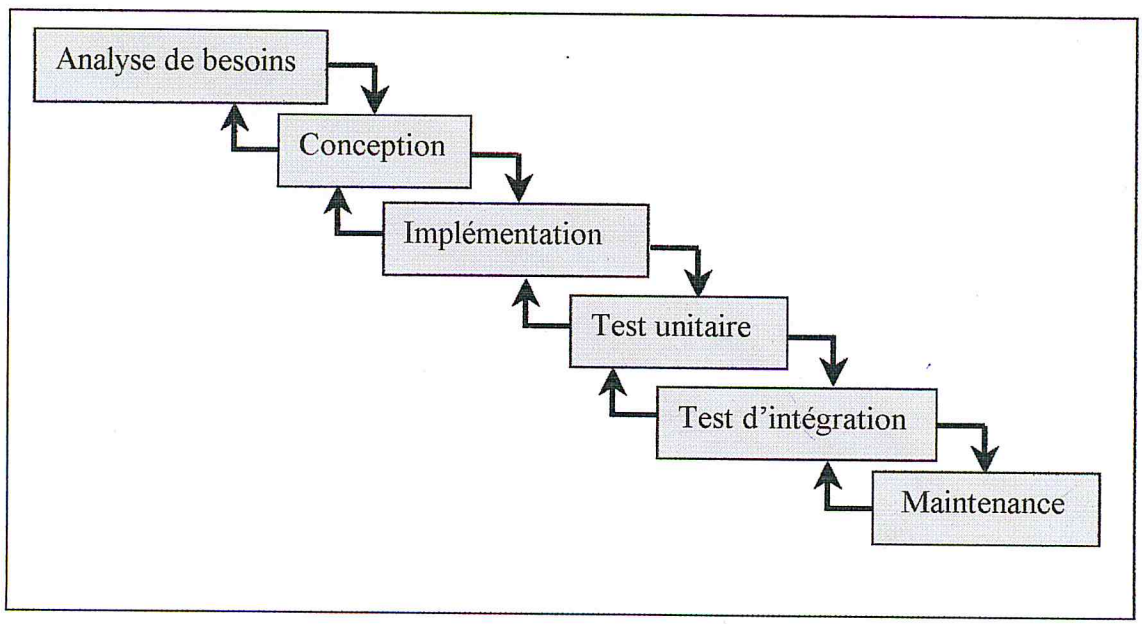


Figure A.1 : *Le modèle de développement en cascade plus détaillé.*

a. *Analyse et définition des besoins*

Les fonctionnalités du système et les contraintes sont établies en consultant les usagers (utilisateurs), elles doivent être définies d'une façon compréhensible ce dernier et par l'équipe du développement.

b. *Spécification et conception*

En partant de l'analyse des besoins, on représente les divers fonctions du système d'une manière permettant d'obtenir un ou plusieurs programmes réalisant ces fonctions.

c. *Réalisation et test unitaire*

On réalise un ensemble d'unités de programme en langage exécutable. Les tests unitaires permettent de vérifier que ces unités répondent à leurs spécifications.

d. *Test du système*

On intègre les unités du programme et on réalise des tests globaux pour être sûr que les besoins ont été satisfais, le système est alors livré au client.

e. *Exploitation et maintenance*

C'est la plus langue étape, elle consiste à :

- Corriger les erreurs qui n'ont pas été découvertes lors des étapes précédentes.
- Améliorer la réalisation des unités du système, et augmenter ses fonctionnalités au fur et à mesure que de nouveaux besoins apparaissent.

Annexe B. UML

1. Historique

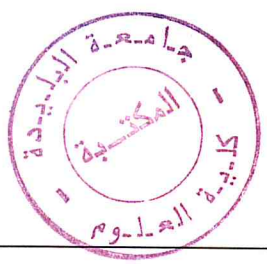
En 1994, plus de 50 méthodes orienté objet de modélisation existent, mais leurs notations graphiques sont toutes différentes [25], afin d'unifié ces méthodes, les analystes ont élaboré une méthode de modélisation qui assemble les avantages de ces méthodes : il s'agit de la notation unifié UML.

UML (*Unified Modeling Language*) est un formalisme de modélisation orienté objet, résultat de l'unification des travaux de *James RUMBAUGH*, *Grady BOOCH* et *Ivar JACOBSON* [24].

2. Avantages de l'UML

Le choix du langage de modélisation porte sur le langage UML de part le fait qu'il constitue une unification des méthodes objets, tirant donc profit des avantages de chacune de ces méthodes de l'origine de l'unification, et qu'il constitue un standard pour la modélisation orientée objet. Ce choix est également justifié par plusieurs autres avantages qu'offre UML par rapport à d'autres méthodes objet et par rapport à nos besoins. Nous les décrivons dans ce qui suit :

- UML se veut un langage non fermé : les éléments de modélisation qu'il propose sont génériques, extensibles et configurables par l'utilisateur [20].
- La simplicité et la capacité d'expression visuelle qu'offre l' UML, et qui permettent de faciliter la communication autour des besoins entre les différents acteurs du système. UML offre une bonne communication avec les utilisateurs. Les concepts manipulés en UML correspondent à des réalités concrètes pour l'utilisateur.
- La diversité des diagrammes qu'offre UML (au nombre de neuf) permet de présenter plusieurs perspectives ou vues de l'architecture du système à développer.
- Les besoins du futur système concernant les utilisateurs sont traduits par les cas d'utilisation.



3. Concepts et notations

Nous décrivons dans cette partie les concepts de base du langage UML. Ce dernier est l'acronyme de « Unified Modeling Language », il représente l'état de l'art des langages de modélisation objets.

UML s'appuie sur des concepts, des relations et des diagrammes :

- Des concepts (structurels, comportementaux, annotationnels, groupements).
- Des relations (association, généralisation, agrégations, compositions…etc.).
- Des diagrammes (statiques et dynamiques).

3.1. Les concepts

UML supporte quatre types de concepts :[23]

- *Les concepts structurels* : représentés par les classes, les interfaces, les collaborations…
- *Les concepts comportementaux*: représentés par les interactions et les états des objets.
- *Les concepts annotationnels* : représentés par les notes. Une note est un commentaire attaché à un ou plusieurs éléments de modélisation [19].

Figure B-1: Représentation d'une note

- *Les concepts de groupement*: représentés par les sous-systèmes et les paquetages.

Les paquetages : Le concept de paquetage unifié les concepts de catégorie et de sous système [24]. Un paquetage regroupe des éléments de la modélisation [25]: cas d'utilisation, classes, objets, modules ou composant. L'importation permet aux éléments d'un paquetage d'accéder aux éléments d'un autre paquetage. Cette relation est à sens unique et est représentée par une relation de dépendance associée à un stéréotype « import ». L'accès d'un paquetage à un autre paquetage est présenté par le stéréotype « accède ».

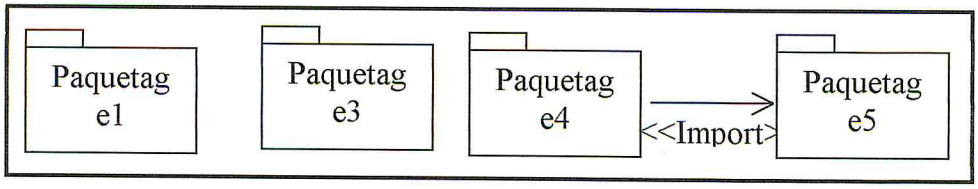


Figure B.2: Réprésentation graphique d'un paquetage

3.2. Les relations

Elles permettent de relier les concepts entre eux. On distingue quatre types de relations, les associations, les généralisations, les agrégations, les compositions …etc [23].

3.2.1. *L'association*

Relation structurelle précisant que les objets d'un élément sont reliés aux objets d'un autre élément.

3.2.2. *La généralisation*

Relation entre un élément général et un élément dérivé de celui-ci, mais plus spécifique (désigné par sous-élément ou élément fils).

Le plus souvent, la relation de généralisation est utilisée pour représenter une relation d'héritage.

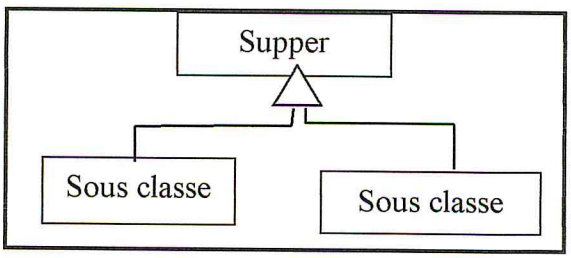


Figure B.3: Représentation graphique d'une généralisation

3.2.3. *L'agrégation*

Représente une association non symétrique dans laquelle une extrémité joue un rôle prédominant par rapport à l'autre extrémité [19].

3.2.4. *La composition*

Relation d'agrégation mettant en avant une notion de propriété forte et de coïncidence des cycles de vie. Les parties sont créées et détruites en même temps que le tout [23]. Elle est distinguée par un losange plein.

3.3. Les diagrammes d'UML

UML définit neuf types de diagrammes qui sont présentés dans l'extrait du méta modèle suivant :

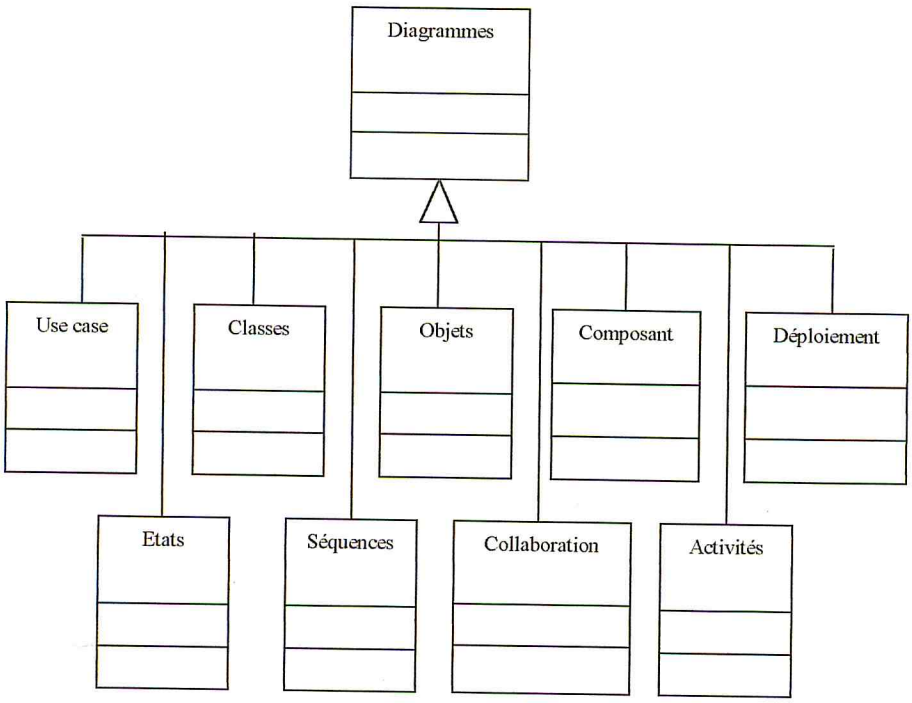


Figure B.4: Les diagrammes d'UML

3.3.1. *Diagramme de cas d'utilisation*:

Un diagramme de cas d'utilisation permet de décrire les interactions entre les acteurs de l'organisation et le système dans chacun des cas d'utilisation envisagés. Il décrit le comportement d'un système au point de vue de l'utilisateur et fixe les limites du système et les relations entre le système et l'environnement [19].

Les divers cas d'utilisation du système vont être présentés dans les diagrammes de cas d'utilisation. Les cas d'utilisation peuvent être liés les uns aux autres par trois types de relations.

- *La relation d'utilisation :* lorsque une ou plusieurs tâches sont utilisées régulièrement, on peut les factoriser dans un même *use case* et faire de telle sorte que d'autres *use cases* l'utilisent en le pointant par une flèche [25].

Cet *use case* est en fait une sous partie de chaque *use case* qui l'utilise. Ce qui permet de décomposer un *use case* complexe en plusieurs *uses cases*.

- *La relation d'inclusion :* le cas d'utilisation source comprend également le comportement de son cas d'utilisation destination. Cette relation a un caractère obligatoire (à la différence de la généralisation) et permet ainsi de décomposer des comportements partageables entre plusieurs cas d'utilisation différents [25].

- *Le relation d'extension :* le cas d'utilisation source ajoute son comportement au cas d'utilisation destination. L'extension peut-être soumise à condition. Cette relation permet de modéliser des variantes de comportement d'un cas d'utilisation [25].

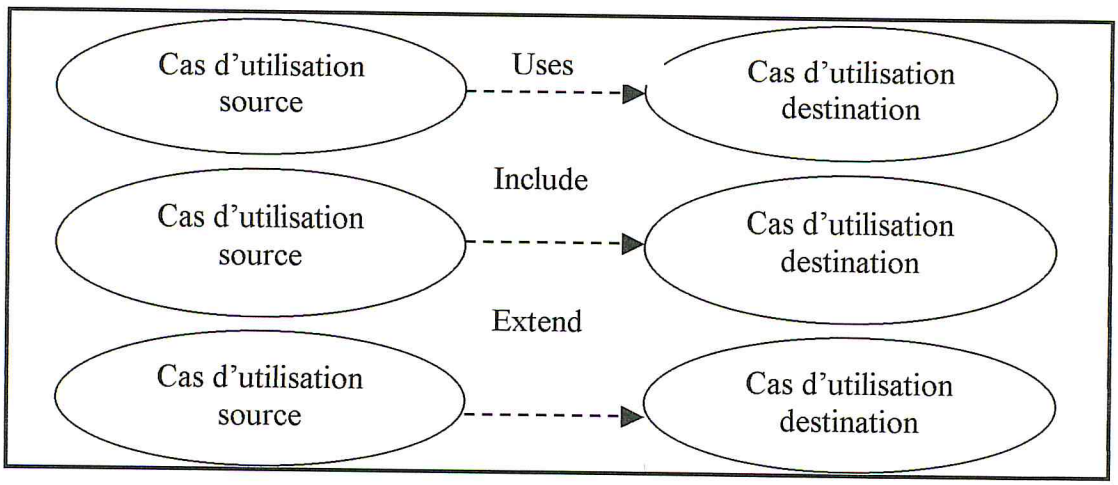


Figure B.5 Les trois relations entre cas d'utilisation

3.3.2. *Diagramme de classes*

Les diagrammes de classes montrent la structure du système et les éléments des classes tels que: les classes, les relations d'héritages entre classes, les associations, dont les agrégations, les attributs, les opérations et la spécification d'opérations et contraintes au niveau des entités [23].

- *Stéréotype :* permet de définir une utilisation particulière d'éléments de modélisation existants ou de modifier la signification d'un élément [23].

Les stéréotypes de base d'UML sont :

Figure B.6: Les stéréotypes d'UML

- *Acteur* : représente les rôles des interlocuteurs du système.
- *Objet interface* : représente les limites du système.
- *Objet contrôle* : représente les classes effectuant des traitements internes au système.
- *Objet entité* : représenté les objet de domaine [24].

3.3.3. *Diagramme des activités*

Ce diagramme permet de décrire le déroulement d'un cas d'utilisation particulier. Il est possible de décrire les acteurs responsables de chaque activité par l'utilisation des «couloirs d'activités» qui permettent de répartir graphiquement les différentes activités entre les acteurs opérationnels [20]. Chaque activité est placée dans le «couloir» correspondant à l'acteur qui assume cette activité.

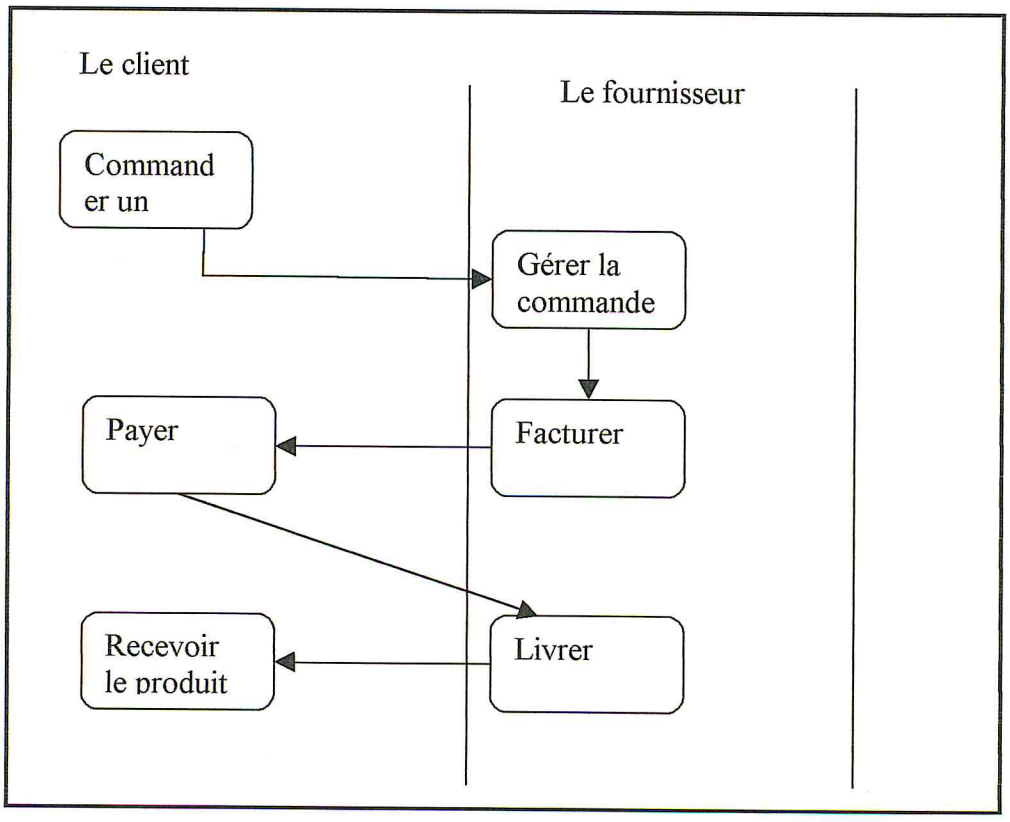


Figure B.7: Exemple d'un diagramme d'activité [19].

3.3.4. *Diagrammes de composants*:

➢ *Composant*: élément physique qui représente une partie implémentée d'un système. Il peut être du code, un script, un fichier de commandes, etc. les composants présentent un ensemble d'interfaces [23].

Les diagrammes de composants permettent de décrire l'architecture physique et statique d'une application en terme de modules : fichiers sources, librairie, exécutables etc. [20]. Ils décrivent les éléments physiques et leurs relations dans l'environnement de réalisation, ils montrent les choix de réalisation [19].

Les composants du systèmes : le sous système, le module, le programme et le sous programme, le processus et la tâche.

➢ *Module* : un système peut être décomposé en modules, chaque module correspond à un ensemble d'éléments physiques (fichiers, bibliothèques, sous ensembles de logiciel.) [25]. La figure suivante donne le formalisme d'un module.

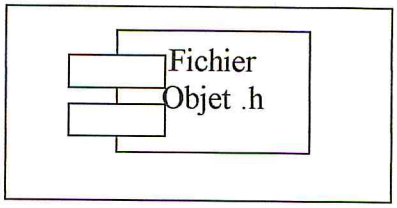


Figure B.8: Représentation d'un composant (fichier)

3.3.5. *Diagramme de séquences*:

Les diagrammes de séquences permettent de représenter les interactions entre objets en précisant la chronologie des échanges de messages. Ils peuvent être utilisés pour représenter les scénarios d'un cas d'utilisation donnée[25].

➢ *Interactions*: modélisent un comportement dynamique entre objets [19]. Elles se traduisent par l'envoi de messages entre objets. Un diagramme de séquence représente une interaction entre objets, en insistant sur la chronologie des envois de messages [23].

➢ *Les messages* : Les messages échangés sont représentés au moyen de flèches horizontales partant de l'émetteur vers le récepteur. L'ordre de l'envoi est donné par la positions sur l'axe vertical [24].

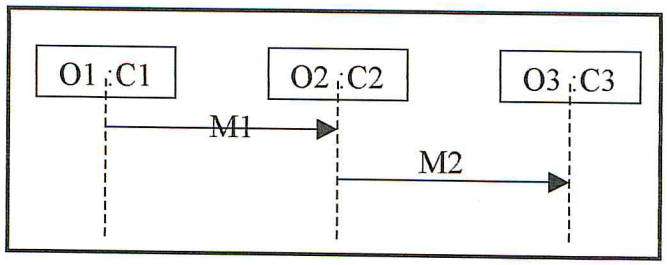


Figure B.9: Agencement de messages

Le diagramme de séquence distingue 5 types de messages prédéfinis qui sont présentés dans le tableau suivant :

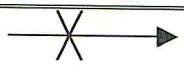
Type de message	La signification
	Message simple, exemple : à une passation de contrôle en mono tâche.
	Message asynchrone, pas de réponse attendue par l'émetteur.
	Message synchrone, réponse nécessaire du destinataire
	Message dérobant, le destinataire doit être à l'écoute
	Message minuté, l'émetteur est bloque pendant un laps de temps.

Tableau B.1 : Les différents types de messages

➢ *Période d'activation*

Correspond au temps pendant lequel un objet effectue une action, soit directement, soit par l'intermédiaire d'un autre objet.

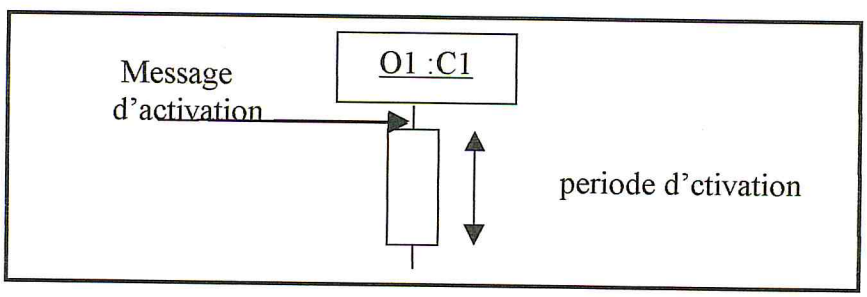


Figure B.10: Activation d'un objet de manière simple.

3.3.6. *Diagrammes de collaboration*

Les Diagrammes de collaboration montrent des interactions entre les objets et les acteurs. Ils permettent de représenter le contexte d'une interaction, car on peut y préciser les états des objets qui interagissent et peuvent être utilisés pour identifier les principaux objets [25]. La figure suivante présente le formalisme de base d'un diagramme de collaboration : un échange de message entre deux objets.

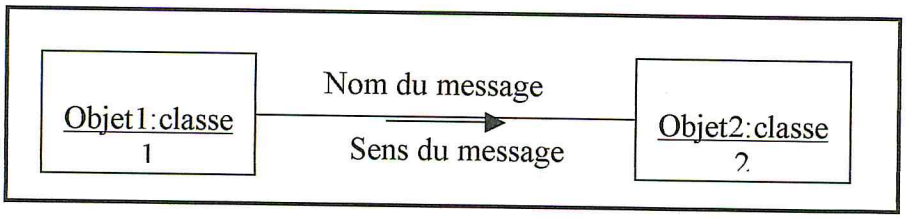


Figure B.11.: Formalisme de base du diagramme de collaboration

III.3.7.Diagramme d'état-transition :

Un diagramme d'état-transition est un graphe constitué de nœuds représentants des états ainsi que des flèches représentant des transitions, portants des paramètres et des noms d'événements[24]. Les diagrammes d'états permettent de définir le comportement d'un objet particulier vis-à-vis des sollicitations internes ou externes auxquelles il peut être soumis [25]. Ils permettent aussi de décrire l'évolution dans le temps les états des objet d'une certaine classe, les événements auxquels ils réagissent et les transitions qu'ils effectuent.

Les diagrammes d'états visualisent des automates (Figure suivante) d'états finis, du point de vue des états et des transitions [19].

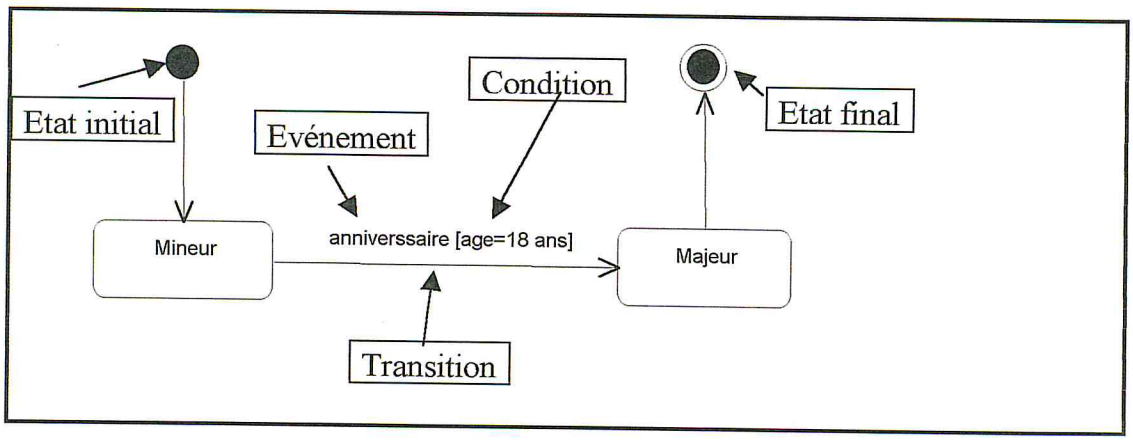


Figure B.12: Exemple de diagramme d'état-transition

➢ *Points d'exécution des opérations*

Il existe six points pour spécifier les opérations qui doivent être exécutées. Ces points sont dans l'ordre d'exécution :

- L'action associée à la transition d'entrée (op1).
- L'action d'entrée d'état (op2).
- L'activité dans l'état (op3).
- L'action de sorite d'état (op4).
- L'action associée aux événements internes (op5).
- L'action associée à la transition de sortie de l'état (op6).

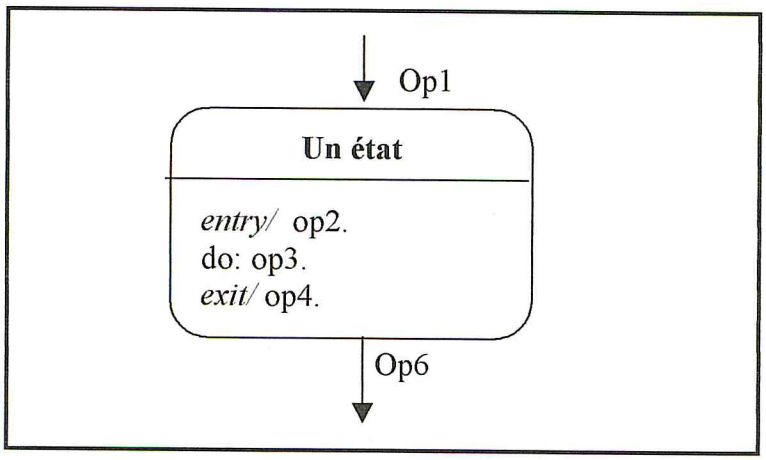


Figure B.13: Les points d'exécution pour un état [19]

Annexe C. XML

1.XML (eXtended Markup Language)

XML Est un langage de description et d'échange de documents structurés. Il permet d'écrire la structure de documents principalement textuels à l'aide d'un système de balises, permettant de marquer les éléments qui composent la structure et les relations entre ces éléments.

XML offre de très grande possibilités en matière d'abréviation, et encourage les entreprises à créer des bases d'abréviations renvoyant à des entités textuelles communes et réutilisable.

Plusieurs propriétés graphiques ou typographiques du document finale sont absentes du source XML, elles sont définie dans la feuille de style. Une séparation de la description structurelle de documents et de la description de leur réalisation physique a comme avantages :

- Facilité d'échange et de production coopérative de documents.
- Indépendance par rapport à des logiciels particuliers.
- Pérennité des documents sur très long terme.

XML pourra être utilisé comme un formalisme puissant et commode pour l'échange de données entre les applications informatiques les plus diverses.

2.Origine du XML

XML est le résultat obtenu par des chercheurs partenaires du World Wide Web Consortium (le W3C), il a pour but de faciliter l'échange des documents complexes sur le web en dépassant les limites imposées par l'HTML, et la complexité qui réside en SGML (Standard Generalized Markup Language).

Les travaux de conception du XML ont débutés en juin 1996 avec la création d'un groupe de travail spécialisé, en février 1998 la première étape a été franchie avec la publication d'une recommandation pour la version 1.0 du langage de base, dans sa première version qui n'inclut pas de mécanisme de liens hypertextes(XML 1.0), ce qui dans le jargon de ce consortium désigne une norme industrielle de fait. En fait XML est défini par tout un ensemble de recommandations déjà parues ou à paraître.

3.HTML et XML

Plusieurs utilisateurs d'HTML disent que ce formalisme est facile à apprendre, il permet l'inclusion d'images, de vidéos et de sons…etc. Il donne la possibilité de générer automatiquement des documents par une application.

Comment peut-on donc justifier la création et l'utilisation de XML ? :

- les balises disponibles en HTML et la sémantique qui leurs est associée sont prédéfinies par la norme, tendis que, l'utilisateur peut créer son propre jeux de balises.
- En HTML la description physique et la description structurelle sont mêlées, se qui conduit à deux difficultés majeures :
    - Les documents ainsi créés ont une espérance de vie relativement courte, et limitée par la pérennité des technologies capable de donner aux éléments de mise en page l'interprétation attendue par les auteurs.
    - Un même document ne peut pas être facilement réalisé selon des modèles physiques différent, ce qui limite son accessibilité. XML résout complètement ce problème en séparant les données physiques et structurelles.

Les apports décisifs de XML

La question qui se pose est, quels sont les avantage que XML apporte au niveau application, qui peuvent motivés le choix de cette norme pour un système documentaire ? :

- Extensibilité et structure balisés.
- Modularité.
- Contrôle de validité.
- Accès à des source d'informations hétérogènes…etc.