Chapitre II : L'édition de documents multimédias

1.Définition du processus de création d'un document multimédia

Le processus de création de documents multimédias est un processus complexe qui ne fait pas forcément intervenir un environnement auteur. On peut distinguer trois principales manières de créer un document multimédia (illustrées par la Figure II-1) :

1. Edition via un environnement auteur : l'outil offre un support à l'auteur via une interface graphique et un ensemble de services, par exemple :

- Visualisation du document ou de certains aspects du document en cours de construction, en offrant différentes vues : vue de la structure, vue de l'organisation temporelle.
- Indépendance de l'auteur vis-à-vis de la syntaxe du langage dans lequel sera sauvegardé le document. C'est le cas par exemple des éditeurs de documents au format HTML (Netscape, Amaya) qui permettent à l'auteur d'écrire des documents sans se soucier de la syntaxe de ce langage.
- Automatisation de certaines tâches : maintenance des liens, vérification de la cohérence.

2. Edition via un éditeur de texte: ce type d'édition est encore souvent utilisé car les auteurs ne trouvent pas d'environnement d'édition qui satisfassent pleinement leurs besoins.
3. Génération automatique de documents: dans ce type de processus, l'intervention de l'auteur est minime voire inexistante. La production du document se fait par génération à partir d'un programme. Ce sont par exemple :

- Les documents construits pour présenter au lecteur le résultat d'une requête à une base de données. Le système doit alors ordonner les résultats et les présenter au lecteur.
- Les documents qui sont produits par traduction d'autres documents.

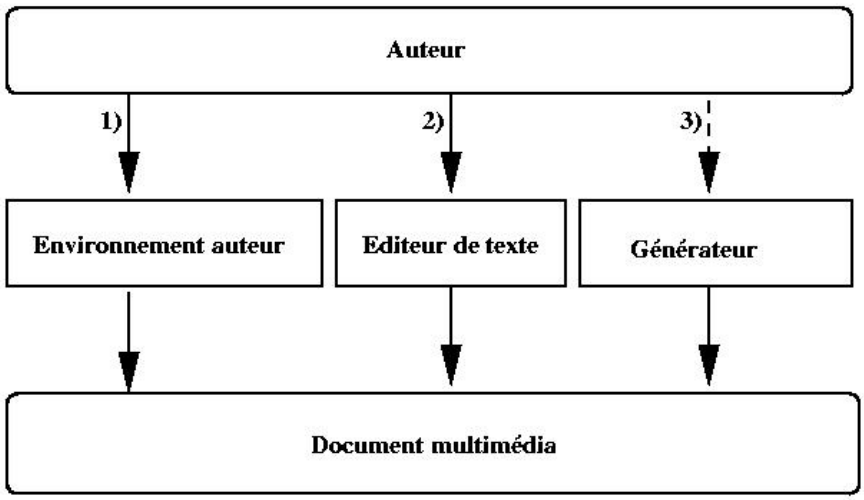


Figure II-1 : Les trois principaux modes de création de documents multimédias

Les générateurs de documents utilisent aujourd'hui la production automatique de documents. A partir d'un ensemble de données le générateur produit des informations de présentation pour permettre aux lecteurs de

les visualiser. Ce mode de création n'est pas adapté à une édition interactive.

L'objectif de cette thèse est d'apporter une réponse pertinente au problème de l'édition interactive de documents multimédias en se plaçant dans le contexte d'un environnement auteur. Nous espérons ainsi éviter le plus possible l'écriture de documents multimédias par l'intermédiaire d'éditeurs textuels. En particulier, il convient de porter une attention particulière au fait qu'avec l'émergence d'XML, de nombreux langages apparaissent. Il est donc nécessaire d'imaginer des environnements auteur qui soient capables de gérer cette multiplicité de formats.

L'objectif de ce chapitre est tout d'abord d'identifier les besoins des auteurs de documents multimédias et de confronter ces besoins aux langages et outils existants. La difficulté de ce travail vient de l'interdépendance entre langages et outils comme nous le verrons dans l'analyse de ces différents éléments.

Je commencerai donc dans la section 2 de ce chapitre par identifier les différents besoins issus du processus de création d'un document multimédia. Le but est de les confronter dans les deux sections suivantes, d'une part aux différents types de langage qui existent (section 3) et d'autre part aux outils auteur les plus représentatifs du domaine (section 4). Enfin, en conclusion de ce chapitre, je ferai un bilan des différentes approches de spécification et d'édition de documents multimédias.

2. Analyse des besoins pour la création multimédia

Dans le cadre des environnements auteur, l'auteur manipule indirectement le langage de sauvegarde. La structure de donnée interne de l'application est une représentation de ce fichier, représentation dont le rôle est de simplifier la tâche d'édition de l'auteur. L'environnement auteur a pour rôle d'assister l'auteur dans cette édition.

Il m'a donc semblé judicieux de séparer au cours du processus d'édition :

- Les besoins couverts par le langage de sauvegarde utilisé pour décrire les documents multimédias, et plus précisément aux formalismes d'édition (absolu, relationnel) utilisés pour spécifier et sauvegarder ces documents. Nous verrons que certains langages peuvent être une combinaison de plusieurs formalismes.
- Les besoins couverts par les environnements. Plus précisément nous identifierons les services que doivent fournir de tels environnements pour offrir à la fois une édition simple et intuitive à l'utilisateur novice et une édition puissante à l'utilisateur expérimenté.

2.1 Besoins de l'auteur couverts par les langages

Pour analyser les capacités des langages, nous allons définir un ensemble de propriétés que nous avons regroupées en deux grandes catégories :

- Expressivité, c'est-à-dire ce que le langage permet de définir comme comportements ou comme propriétés sur les objets de base d'un document.
- Capacité de spécification, c'est-à-dire la simplicité avec laquelle l'auteur peut définir ces comportements à l'aide du langage.

2.1.1 Expressivité

Comparer formellement l'expressivité des langages multimédias nécessiterait d'être capable d'identifier des classes de documents (par exemple sous forme de classes d'automates) et de définir ensuite comment chaque langage permet de couvrir partiellement ou entièrement chaque classe. Sans vouloir effectuer cet important travail théorique qui sort des objectifs de cette thèse, nous allons présenter deux types de besoins nous permettant ensuite d'identifier trois classes de documents :

Spécification des médias, qui recouvre les besoins de l'auteur en termes de définition des composants élémentaires de son document.

- *Paramétrage des attributs associés aux médias*: il est important que l'auteur puisse paramétrer les informations de visualisation de ces médias que ce soit pour les attributs spatiaux (X, Y, largeur, hauteur, couleur), mais aussi temporels (début, fin, durée).
- *Médias continus / discrets :* l'auteur a besoin de spécifier dans son document à la fois des médias discrets (images, textes) et des médias continus (sons, vidéos).
- *Médias déterministes/indéterministes :*Lorsque l'auteur inclut un média continu dans un document, il doit pouvoir spécifier le comportement de ce dernier lors de l'exécution. En effet, l'exécution de média continu ne peut être prévue statiquement si l'on désire préserver tout le contenu sémantique du média. L'auteur doit donc pouvoir spécifier le comportement qu'il désire. Deux cas sont possibles :
  - Déterministe : le système de présentation contrôle l'exécution et pour satisfaire la spécification de l'auteur il peut décider d'accélérer ou de ralentir la vitesse de présentation du média.
  - Indéterministe : le système de présentation n'aura pas de contrôle sur la présentation du média. Cela soit parce qu'au moment de jouer le média il n'y a pas de moyen de connaître la durée du média (concert en direct par exemple), soit parce que l'auteur veut délivrer complètement le contenu sémantique du média.
- *Plugins / Applets* : l'auteur peut importer dans son document des modules externes qui lui permettent d'afficher, de manipuler des données structurées ayant déjà leur propre comportement comme des scènes spécifiées en VRML[VRML99].

Les comportements, qui permettent à l'auteur de modéliser l'enchaînement temporel et spatial du document ainsi que les actions autorisées au lecteur :

- *Comportement local,* permet de définir l'enchaînement temporel et spatial entre les médias ou les groupes de médias.
- *Interactions locales*, définies lors de la spécification du document, ces interactions peuvent permettre au lecteur de déclencher des actions sur le document comme le lancement ou l'arrêt d'un objet. Ces interactions modifient le cours de l'exécution en incluant (ou en supprimant) des médias dans le cours de la présentation.
- *Navigation,* définie lors de la spécification, elle permet au lecteur de parcourir le document. Cette navigation peut être structurelle, spatiale ou temporelle. Les actions de navigation sont orthogonales au comportement temporel du document en ce sens que l'on peut définir son comportement sans tenir compte de ces interactions de navigation qui, lorsqu'elles surviendront, changeront uniquement l'instant courant de présentation du document.
- *Acquisition d'information*, ce sont tous les moyens qui permettent de récupérer des informations de la part du lecteur (champ de saisie texte, bouton de sélections multiples,).

Ces différentes classes de documents permettent à l'auteur d'écrire des documents multimédias très variés. On peut cependant les classer en trois grandes classes selon le type de comportements qu'elles permettent :

- *Les documents entièrement prédictifs* **:**dès le début de la présentation du document, on sait exactement ce qui va se produire et il n'y a pas d'interaction utilisateur. Le système de présentation peut calculer les instants de début et de fin des objets statiquement. Dans le cas particulier où ces documents contiennent des objets indéterministes, ces objets seront coupés ou maintenus à l'écran en fonction de la durée calculée initialement. Ces documents recouvrent les présentations de type film ou le lecteur n'interagit pas avec le document qui lui est présenté.
- *Les documents prédictifs avec navigation :*ces documents sont constitués d'un ensemble de sous-documents prédictifs dans lequel le lecteur peut naviguer. Cette navigation peut être de deux ordres :
  - *Navigation temporelle* **:** où le lecteur peut aller à un instant précis de la présentation.
  - *Navigation hypertexte* **:** où le lecteur passe d'un sous-document à un autre.

Ces documents recouvrent les présentations de type diaporama ou présentation hypertexte classique. La navigation ne remet pas en cause le comportement des documents.

- *Les documents interactifs*: un document interactif est un document qui implique le lecteur. Dans ce type de documents il n'est plus possible de prédire l'exécution du document car celle-ci est gouvernée par des interactions locales ou par des médias. C'est le cas par exemple d'un document très simple (voir Figure II-2) contenant une bande son en boucle infinie. Lorsque le lecteur appuie sur une touche, cela déclenche une vidéo. La fin de la vidéo interrompt la bande son, et termine le document. Les interactions du lecteur peuvent modifier localement le document et remettre en cause les durées calculées initialement. De plus, les objets indéterministes ne sont plus forcément interrompus (du fait que l'on ne soit plus contraint de connaître statiquement la durée d'un objet), on peut les laisser s'exécuter jusqu'à la fin. Le système devra être en mesure de réévaluer la suite du document pour garder les synchronisations entre les objets. Cela impose une certaine réactivité du système de présentation.

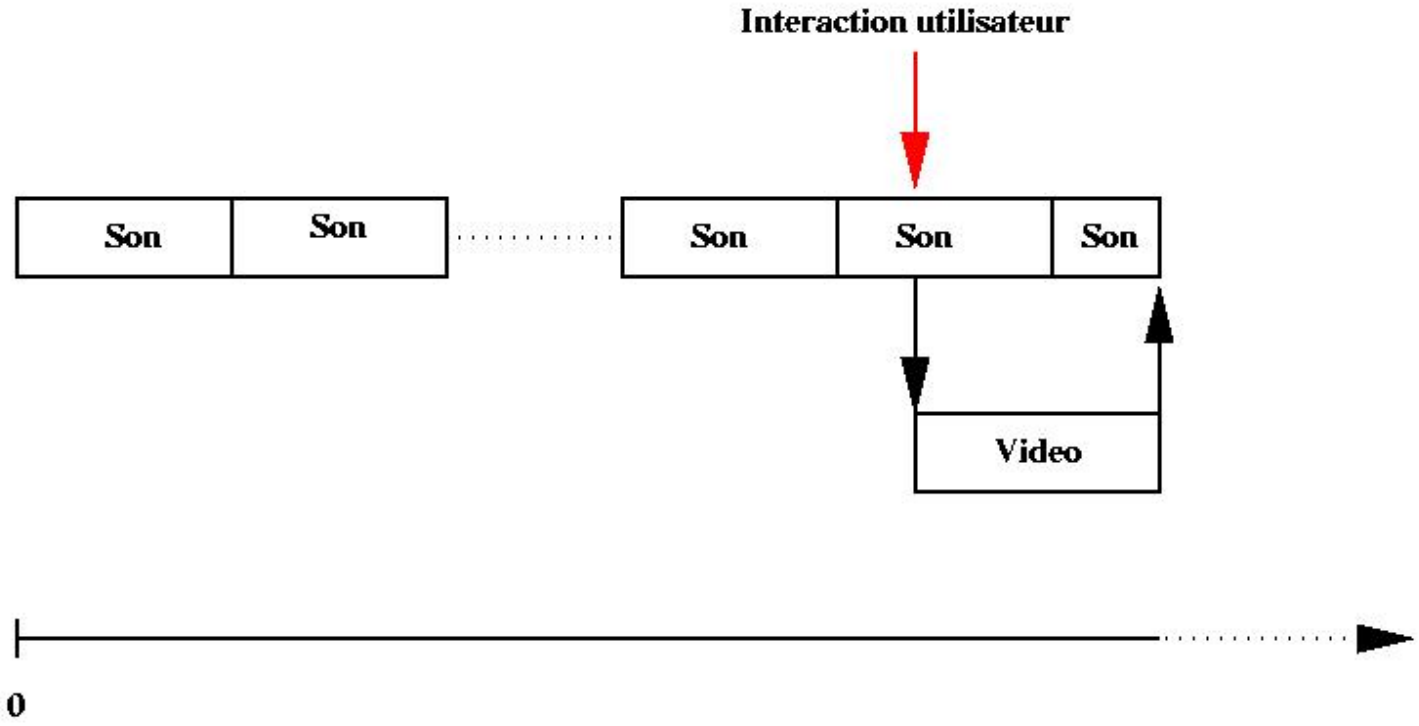


Figure II-2 : Exemple de scénario d'un document indéterministe

Nous verrons par la suite que la difficulté d'édition d'un document est liée en grande partie à la classe à laquelle il appartient.

2.1.2 Capacité de spécification

Les besoins de l'auteur lors de la spécification de documents multimédias peuvent être classés en quatre catégories :

- *Possibilité de structuration du document*pour aider l'auteur dans le découpage de son document, ceci afin de simplifier la phase d'édition. La structuration permet de définir des groupes dans un document et d'associer des propriétés communes aux objets du groupe. Elle permet ainsi de découper la tâche d'édition en sous-tâches plus simples. De plus l'auteur peut réutiliser certaines parties dans d'autres documents (ou dans le même). On peut rapprocher cette notion de celle de module dans les langages de programmation avec toutes les difficultés qui s'y rattachent (partage, mise à jour de version). Plusieurs structures peuvent cohabiter dans le document, ces structures peuvent être spatiale, hypertexte, logique mais aussi temporelle.
- *Support pour la modification du document*, pour que l'auteur puisse changer facilement la présentation

ou le contenu d'un document existant. Modifier un document consiste soit à modifier les attributs d'un objet, soit à modifier la structure du document, soit à modifier le contenu des médias. Dans tous les cas, ces modifications doivent pouvoir se réaliser sans remettre totalement en cause le document et le travail effectué jusque-là.

- *Rapidité de spécification de documents*. Lors de la phase d'édition, l'auteur doit pouvoir exprimer le comportement spatial et temporel de son document avec une difficulté proportionnelle à la complexité du comportement voulu. Il doit pouvoir par exemple exprimer des comportements complexes sans avoir à exprimer tous les comportements élémentaires nécessaires à sa réalisation. Par exemple, si l'auteur veut exprimer le déplacement d'un objet d'un point A vers un point B, il ne doit pas avoir à spécifier toutes les positions intermédiaires.
- *Support pour la spécification de documents adaptables*. L'auteur pourra définir un ensemble de comportements dépendant des conditions de présentation pour son document.

Nous allons maintenant détailler les deux derniers points.

Rapidité de spécification : c'est le besoin élémentaire qui permet, par exemple, à l'utilisateur novice de spécifier facilement un petit document en n'utilisant qu'une partie de l'environnement auteur. Sans faire un inventaire de toutes les instructions qu'aurait un langage de programmation, nous allons présenter les principales instructions utiles pour un auteur de documents :

- *Relations entre les objets* : elles permettent à l'auteur de mettre des relations prédéfinies entre les objets, lui évitant ainsi d'avoir à spécifier l'ensemble des comportements élémentaires qui sont communs à tous les documents. C'est par exemple le cas des relations temporelles *avant* et *après* ou des relations spatiales *centré* ou *aligné*. On peut noter que certaines relations sont flexibles (ou partielles), c'est-à-dire que l'auteur ne spécifie pas explicitement un unique placement temporel (ou spatial) mais un ensemble de placements possibles. Par exemple, lorsque l'auteur dit "A *avant* B", il n'explicite pas exactement le placement temporel de A, par rapport à B.
- *Evénements*: ils permettent à l'auteur de spécifier des comportements relativement complexes, en lui permettant d'associer des actions à un événement dynamique (condition externe), et dont l'instant d'occurrence n'est pas connu au moment où l'auteur écrit son document. Nous appellerons par la suite *événement*, l'événement et l'action qui lui sont associés. Nous présenterons de manière plus précise cette notion dans la section 3.3.
- *Instructions conditionnelles* : il est important que l'auteur puisse réaliser des tests, soit sur l'état de certains objets (pause, actif, affiché), sur la valeur d'attributs, sur la valeur de variables ou sur l'état du système (que ce soit le système de présentation ou le système de la machine lui-même).
- *Boucles finies ou infinies* : il peut être utile de définir des boucles pour la présentation d'un média ou d'un groupe de médias. Par exemple pour spécifier que pendant toute la présentation du document une musique de fond est jouée en boucle.

Support pour la spécification de documents adaptables :cette catégorie de besoins est celle qui consiste à écrire des documents qui s'auto-adaptent au contexte de présentation. Par contexte de présentation, nous entendons les conditions liées au système ou à l'utilisateur. Ces différents paramètres sont [Layaida99] :

- *Le système :*
  - *Le périphérique de restitution* : si l'auteur veut que son document soit affiché de façon optimale en fonction, par exemple, la taille de l'écran utilisé ou alors de façon plus générale quel que soit le périphérique de sortie (ordinateur, borne interactive,).
  - *Le débit réseau* : en cours de présentation le débit du réseau peut évoluer, l'auteur doit pouvoir exprimer comment son document réagit à ce genre d'évolution. Par exemple, si les conditions deviennent insuffisantes, il doit pourvoir changer les médias utilisés ou même changer l'agencement temporel des objets de manière à diminuer les ressources nécessaires.
  - *La charge de la machine*: de la même manière que le réseau, la charge de la machine sur laquelle est présenté le document peut évoluer. La mémoire disponible ou la charge du processeur peuvent changer en cours de présentation.

- *L'utilisateur :*
  - *La langue* : par exemple pour changer la langue de présentation de son document. Lorsque l'auteur veut faire un document dans plusieurs langues avec les mêmes enchaînements temporels et spatiaux, il peut être utile de lui permettre de spécifier un média pour chacune des langues qui l'intéressent.
  - *Le niveau d'expertise* : si l'auteur veut définir un comportement différent de son document en fonction du niveau de l'utilisateur. Par exemple on peut imaginer qu'un professeur ayant des élèves de niveaux différents oblige les élèves débutants à effectuer les exercices liés à un chapitre alors qu'il obligerait la lecture du résumé du cours pour les élèves en période de révisions.
  - *Les déficiences*: le lecteur du document peut avoir des déficiences momentanées ou permanentes. Ces déficiences visuelles ou auditives par exemple peuvent nécessiter une adaptation du document pour une lecture optimale. On peut, par exemple, imaginer que le système utilise des polices de taille plus grosse pour les personnes malvoyantes, ou qu'il remplace les médias textuels par des médias sonores.

2.2 Besoins de l'auteur couverts par l'environnement d'édition

2.2.1 Le processus d'édition

L'auteur suit le même processus d'édition quel que soit le formalisme d'édition qui lui est offert. Nous allons donc dans un premier temps nous intéresser à la définition du cycle d'édition avant de définir, dans un deuxième temps, les besoins qui en découlent.

On peut définir le processus d'édition en cinq étapes (voir Figure II-3):

1. Ouverture du document, c'est la phase pendant laquelle l'auteur ouvre un document existant (transition 1), le système vérifie sa cohérence et le formate (transition 2) c'est-à-dire qu'il calcule les placements spatiaux et temporels de chaque objet pour l'afficher à l'écran (transition 3).
2. Perception du document et navigation : c'est la phase ou l'auteur visualise chacune des dimensions (spatiale, temporelle, logique, hypertexte) du document. La présentation de ces différentes informations et les mécanismes de navigation offerts à l'auteur à l'intérieur de ces informations sont donc essentiels.
3. Modification du document, parmi ces opérations on peut distinguer :

- *Les opérations qui modifient la structure et le contenu du document*, ce sont toutes les opérations qui permettent d'ajouter ou de supprimer de nouveaux médias dans le document, ou qui permettent de structurer le document.
- *Les opérations qui modifient la présentation du document*, c'est-à-dire toutes les opérations qui modifient le placement temporel ou spatial, ou les attributs des médias. Le système applique alors les modifications sur le document et fait appel aux services de vérification de cohérence.

4. Vérification de la cohérence et formatage : cette dernière étape correspond à la réaction du système à l'opération d'édition faite par l'auteur (transition 5), c'est-à-dire que le système doit, dans un premier temps, vérifier la cohérence de cette opération, et appliquer les différents changements dans le document de manière à prendre en compte cette modification. Il doit alors afficher le résultat et le processus d'édition recommence à l'étape 2 (transition 3). C'est pour cela que l'on parle de processus d'édition cyclique.
5. Sauvegarde du document. L'environnement sauvegarde la représentation interne du document dans un fichier de sauvegarde (transitions 6 et 7).

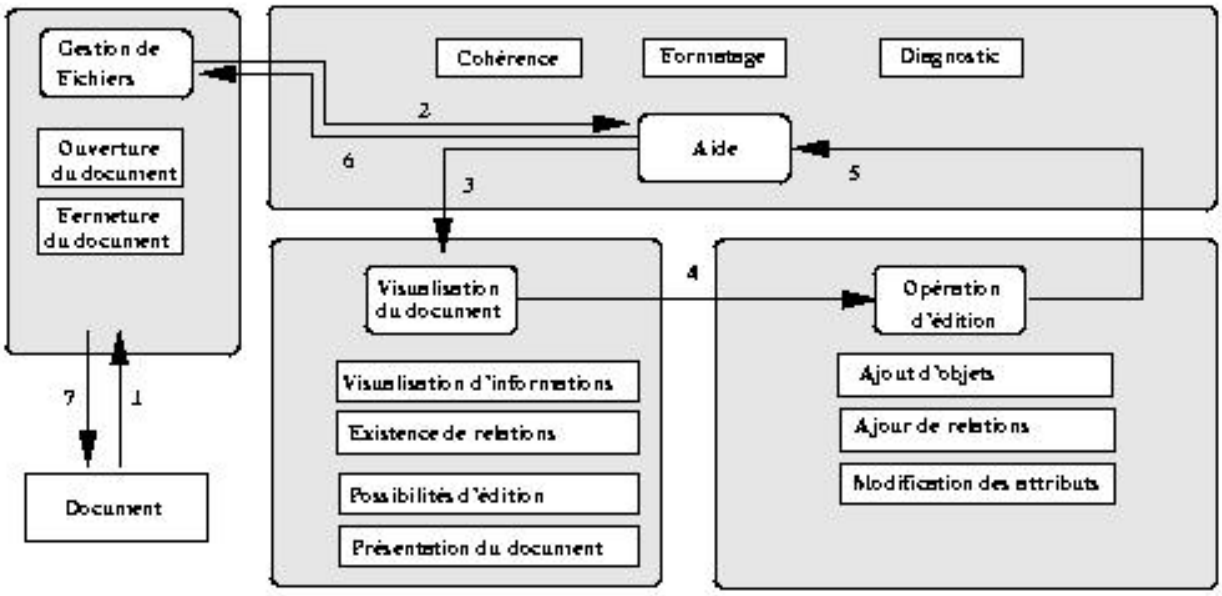


Figure II-3 : Définition du cycle d'édition d'un document multimédia

Du cycle d'édition défini précédemment, on dégage les quatre grandes fonctionnalités suivantes (voir Figure II-3):

- La gestion de fichiers (chargement, sauvegarde) ;
- La visualisation des informations contenues dans le document et la présentation du document ;
- L'édition proprement dite ;
- L'aide à l'auteur pour faciliter et automatiser certaines tâches : cohérence, formatage,

2.2.2 Les besoins des auteurs suscités par le processus d'édition

Du processus d'édition on peut dégager cinq besoins qui sont liés :

- *A trois des grandes fonctionnalités du système auteur* (visualisation, édition, aide).
- *A l'enchaînement des phases d'édition.*
- *A la vie du document en dehors d'une session d'édition* : version, stockage, diffusion, droits,

Visualisation/présentation :

- *Perception du document dans toute sa complexité :* il est important, pendant le processus d'édition, que l'auteur puisse percevoir les informations relatives aux différentes entités constituant son document. Par exemple, il doit pouvoir en fonction de ses besoins d'édition visualiser les informations spatiales et temporelles de son document. Cette visualisation ne doit pas seulement être une visualisation sous une forme textuelle ou via une palette d'attributs. Elle doit se faire selon différents modes de représentation. Ces modes doivent permettre à l'auteur de se faire une représentation mentale du document tel qu'il sera présenté au lecteur. De plus, ces représentations doivent fournir à l'auteur une explication sur le placement des objets en visualisant par exemple les relations (ou les attributs) qui induisent ce placement. Par exemple la dimension spatiale doit être visualisée telle que le lecteur la verra lors de la présentation, cette vue vérifie partiellement le principe du WYSIWYG (What You See Is What You Get) du fait des nombreux périphériques de lecture possibles pour le document. Dans le cas de la dimension temporelle, celle-ci peut être visualisée à l'aide d'une représentation graphique dans laquelle on peut référencer un temps absolu.
- *Présentation* **:**il est important que l'auteur puisse, à chaque étape du processus de conception, visualiser une des présentations possibles du document. Cette étape est essentielle à la compréhension du document car contrairement aux documents classiques, la phase d'édition ne s'effectue pas complètement sur la forme finale du document. En effet, en cours d'édition, l'auteur ne peut visionner

complètement l'aspect dynamique de la spécification de son document, il doit donc visualiser son exécution pour en avoir un aperçu complet.

- *Perception de l'espace de solutions :* l'auteur de documents multimédias de par la flexibilité des relations et la spécification partielle des médias ne définit pas une présentation, mais un ensemble de présentations possibles. Le système auteur doit rendre accessible à l'auteur cet espace de solutions.

Fonction d'édition : Le système auteur doit permettre à l'auteur d'éditer le document en fonction de ses capacités. Il doit aussi faciliter autant que possible cette tâche. De plus, il doit permettre à l'auteur d'exprimer tout ce que le langage de restitution permet de spécifier. Ces fonctions peuvent être des fonctions élémentaires comme l'insertion d'un objet dans le document, ou des fonctions plus évoluées comme la copie d'attributs de présentation ou de synchronisation. Par exemple, on peut imaginer qu'un auteur définisse un ensemble d'attributs spatiaux et temporels pour trois objets de son document, et qu'il désire appliquer le même placement spatial et temporel à trois autres objets de son document.

Aide à l'auteur:

- *Formatage*: un des intérêts d'une spécification relative du comportement des objets, est que le système peut déduire, à partir des informations données par l'auteur, le placement absolu des objets. Ce calcul, appelé *formatage,* s'applique aussi bien dans la dimension spatiale que temporelle. L'auteur n'a pas ainsi, à chaque modification de son document, à recalculer le placement de chacun de ses objets, c'est le système qui réalise cette tâche fastidieuse. Cette fonctionnalité est donc essentielle pour la visualisation des informations temporelles et spatiales du document.
- *Vérification de la cohérence* **:** lors d'une action d'édition, le système doit s'assurer que l'action n'engendre pas d'incohérence dans le document. Par exemple, s'il a été spécifié des informations contradictoires ou une référence à un objet non défini.
- *Diagnostic*: En plus de la vérification de la cohérence, le système doit aider l'auteur à comprendre les raisons des erreurs de spécification. C'est ce qu'on appelle le *diagnostic*.

Cycle d'édition :le processus de création d'un document multimédia est un processus cyclique (voir Figure II-3). C'est-à-dire que l'auteur ne spécifie pas complètement son document avant de le présenter, mais construit plutôt petit à petit son document en utilisant à chaque étape les services de visualisation et de vérification offerts par l'environnement. De ce fait, le système doit prendre en compte le fait que l'édition n'est pas un processus linéaire et doit aider l'auteur au mieux dans ce processus. Les deux points qui nous semblent essentiels dans cette tâche d'édition sont :

- *La rapidité de prise en compte des opérations d'édition*: une qualité importante d'une interface est sa capacité à répondre rapidement aux actions de l'utilisateur. Une non-réponse du système dans un délai raisonnable peut amener l'utilisateur à se lasser et à ne plus utiliser le système, ou il peut l'amener à répéter sa commande, ce qui peut avoir des effets indésirables. Il est donc important lors de la phase d'édition que le temps de prise en compte des opérations soit le plus court possible [Coutaz90]. Cela inclut la phase de vérification de cohérence.
- *La localité des modifications :* lors d'une opération d'édition, le système auteur doit modifier le moins possible la solution courante de manière à ne pas perturber l'utilisateur dans la phase de visualisation.

Cycle de vie du document :nous venons de voir les différents besoins de l'auteur lors de la spécification de son document. La vie du document ne s'arrête pas au moment où l'auteur a fini de l'écrire. Il est important que l'auteur puisse diffuser, échanger, modifier ou réutiliser tout ou partie de son document après sa spécification. Nous allons donc maintenant nous intéresser à ces différents besoins :

- *Diffusion :* un des besoins important est de permettre au document d'être largement diffusé que ce soit par CD-ROM, sur le Web, .
- *Echange avec d'autres personnes / outils :* un des besoins est l'échange des documents en phase d'édition par exemple avec d'autres personnes et/ou d'autres outils.
- *Réédition :* l'auteur peut vouloir rééditer un document, même s'il a déjà fait l'objet d'une diffusion. Il faut donc que le système auteur ait une pérennité temporelle, ou qu'il permette de sauver le document sous une forme *standard*.

- *Réutilisation de parties de document :* lors de la conception d'un document multimédia, on a souvent besoin de réutiliser certaines parties de documents précédents (voir structuration 2.1.2), il faut donc que le système permette à l'auteur d'isoler et de récupérer certaines sous-parties de documents existants. Cependant cette réutilisation n'est pas toujours une chose aisée du fait des synchronisations ou des liens externes sur certaines parties du module.

3. Les standards de documents multimédias

Comme nous l'avons dit en introduction, le document en cours d'édition est représenté par plusieurs entités physiques. Au cours de cette section nous allons nous intéresser aux langages de sauvegarde et nous les illustrerons au travers d'environnements auteur réels utilisant les formalismes du langage de sauvegarde comme formalisme d'édition.

Il existe aujourd'hui de nombreux langages pour décrire des documents multimédias, cependant ils peuvent tous se classer dans une ou dans une combinaison des quatre catégories suivantes :

- *L'approche absolue* : l'auteur décrit de façon explicite le positionnement des objets dans le temps et l'espace.
- *L'approche programmation*: l'auteur décrit le comportement de ses objets médias grâce à un langage de programmation de type impératif.
- *L'approche événementielle*: l'auteur décrit le comportement de ses objets médias grâce à des règles événement / conditions / actions.
- *L'approche relationnelle*: l'auteur décrit des relations (avant, en même temps, au-dessus,) entre les objets.

Au cours de cette analyse, nous allons présenter chacune de ces approches en les illustrant par un ou plusieurs langages représentatifs. Pour chaque approche, nous dégagerons d'un part la façon dont les langages de description de documents multimédias répondent aux besoins auteurs liés au langage (section 2.1) et d'autre part comment un environnement d'édition répond aux besoins définis dans la section 2.2.

Dans un premier temps, nous présenterons l'approche absolue utilisée notamment dans Director qui est un des outils les plus utilisés pour concevoir des documents multimédias. Dans un deuxième temps, nous présenterons les approches à base de langages de programmation en les illustrant au travers du langage Java. Dans un troisième temps, nous présenterons un langage de description à base d'événements (MHEG [MHEG93]) qui a été développé par le groupe de standardisation ISO. Nous présenterons ensuite deux approches relationnelles : SMIL, qui est une recommandation du W3C pour spécifier le comportement temporel des documents et Madeus qui est un langage à base de contraintes développé au sein du projet OPERA. Une des difficultés de l'analyse ci-dessous vient du fait que les langages actuels mélangent les différentes approches.

3.1 L'approche absolue

La caractéristique essentielle de cette catégorie est que l'auteur spécifie de façon absolue le placement spatial et temporel des objets. Cela se traduit par le fait que les instants de début des objets sont placés par rapport au début du document, et que la position spatiale des objets est donnée par rapport à un point de référence de l'écran (par exemple en haut à gauche).

Aucun standard n'existant dans cette approche, les exemples décrits ci-dessous sont basés sur une syntaxe ad hoc.

Dans la Figure II-4 on peut voir un exemple de document décrit de façon absolue. Ce document est composé de cinq éléments textes qui s'affichent successivement dans le temps toutes les 10 secondes. Les cinq éléments disparaissent ensemble de l'écran à la cinquantième seconde du document. Spatialement, les objets sont affichés les uns en dessous des autres sur l'axe Y, et les uns à côté des autres sur l'axe X.

Temporel :

```
Objet Titre1
       Début = 0
       Durée=50

Objet Titre2
       Début = 10
       Durée=40
Objet Titre3
       Début = 20
       Durée=30
Objet Titre4
       Début = 30
       Durée=20
Objet Titre5
       Début = 40
       Durée=10
```

Spatial :

```
Objet Titre1
       X = 120
       Y =  60
       Texte = "Institut"
Objet Titre2
       X = 180
       Y=80
       Texte = "National"
Objet Titre3
       X = 240
       Y=100
       Texte = "de Recherche"
Objet Titre4
       X = 300
       Y=120
       Texte = "en Informatique"
Objet Titre5
       X = 360
       Y=140
       Texte = "et Automatique"
```

Figure II-4 : Description absolue du placement temporel et spatial

L'avantage de cette approche est la facilité de description de documents simples. L'écriture de petites animations, que ce soit par spécification directe dans un fichier textuel ou par manipulation dans une interface graphique, est relativement simple. En effet, il est très simple d'imaginer une interface basée sur une métaphore *table de montage,* où l'auteur place les objets sur un axe de temps absolu.

Les inconvénients majeurs de cette approche sont :

- La difficulté de modification de ce que l'on a spécifié : par exemple, si on veut insérer le texte "INRIA" en haut de l'écran de sorte qu'il soit présent pendant toute l'animation décrite précédemment, il faut modifier le placement temporel et spatial de tous les objets. C'est-à-dire qu'il faut décaler le début de tous les objets de 10 secondes, et décaler tous les objets de 10 unités vers le bas.
- La difficulté de décrire des comportements complexes : l'auteur n'a pas d'instructions disponibles pour exprimer le comportement de son document ce qui limite son pouvoir d'expression.
- L'impossibilité de décrire des documents adaptables en fonction du lecteur ou du périphérique de présentation.
- Pas de synchronisation avec des objets indéterministes. L'auteur ne spécifiant pas de dépendance entre les objets, il ne peut y avoir de synchronisation temporelle avec la fin d'objets indéterministes.

Les générateurs automatiques de documents ([Real99]) produisent souvent des documents sous une forme absolue. De ce fait les documents produits sont relativement peu modifiables et ne sont pas rééditables.

Un outil représentatif de cette approche est Director, il permet à l'auteur de spécifier le placement temporel et spatial de ses objets de manière absolue. Il compense la limitation d'expressivité par l'utilisation de scripts. Nous présenterons Director dans la section 4.3.1.

3.2 Les langages de programmation

Une approche plus générale pour concevoir des documents multimédias est l'utilisation d'un langage de programmation. Il existe deux grandes catégories de langages de programmation :

- L'utilisation de langages de scripts intégrés ou non dans un environnement auteur. C'est par exemple le cas de Lingo intégré dans l'environnement Director.

- L'utilisation de langages plus généraux comme Java ou C++. Il existe depuis quelques années des bibliothèques (Java Media Framework [JMF00]) permettant de manipuler les différents types de médias existants réduisant ainsi le coût de conception de documents multimédias. De plus ces bibliothèques sont portables sur différentes plates-formes, permettant ainsi une plus grande portabilité du document créé. Dans les langages de programmation, on peut distinguer les langages compilés (C, C++, Pascal) des langages interprétés (Prolog, Java) :
  - Les langages compilés nécessitent une phase de compilation du programme pour toutes les plates-formes si l'on désire rendre les documents accessibles aux lecteurs. L'avantage de cette approche réside dans le fait que le document produit est adapté pour une présentation sur la plate-forme cible.
  - Les langages interprétés, quant à eux, peuvent être exécutés sur tous les systèmes d'exploitation qui possèdent une machine virtuelle de ce langage permettant ainsi au programmeur d'être déchargé de la gestion de la portabilité du code écrit. Parmi les langages interprétés Java offre aussi une autre possibilité intéressante : la définition *d'applets*. Une *applet* permet à un programme Java d'être intégré dans une page Web, le rendant ainsi accessible.

Les langages les plus adaptés à cette tâche, de par la diversité des plates-formes de conception et de restitution sont les langages interprétés. Nous allons présenter un peu plus complètement le langage Java qui est un des langages permettant la conception de documents multimédias.

En effet, Java est un langage à objets développé au début des années 90, il intègre complètement la manipulation des médias (JMF [JMF00]), des protocoles réseau et de la sécurité. Il fournit aussi un ensemble de bibliothèques évoluées (Java 2D[Java2D00], Java 3D[Java3D00]) pour les manipulations des médias ou d'objets en deux et trois dimensions.

Nous pouvons voir dans l'exemple de la Figure II-5 le document de présentation de l'INRIA. Nous avons écrit deux versions de ce document. Dans le premier cas (scène1_absolue), nous avons utilisé une spécification absolue, nécessitant une description plus longue et moins facilement modifiable. Dans le deuxième cas (scène1_relative), nous avons utilisé une spécification par relation, plus souple lors de l'édition. Notons que cette spécification nécessite l'écriture d'un formateur temporel et spatial pour calculer le placement réel des médias.

Dans un programme de description d'un document multimédia, l'auteur doit gérer explicitement la manipulation de l'environnement de présentation du document en plus du placement temporel et spatial de son document. Par exemple, il doit gérer la création d'une fenêtre dans laquelle le document va être joué. De plus, la facilité de réutilisation et de manipulation du document lors des opérations d'édition dépend énormément de la façon dont le programme a été conçu.

```
public class DocumentINRIA

void CreerMenu() {
...
}

void Scene1_absolue() {

// le constructeur de Media text prend 5 paramètres : le texte à afficher,
// les coordonnées spatiales de l'objet, l'instant de début de l'objet,
// et l'instant de fin.

MediaText T1 = new MediaText("Institut",120,60,0,50);

//Valeur, X, Y, Début, Fin

MediaText T2 = new MediaText("National",180,80,10,50);
MediaText T3 = new MediaText("de Recherche",240,100,20,50);
MediaText T4 = new MediaText("en Informatique",300,120,30,50);
MediaText T5 = new MediaText("et Automatique",360,140,40,50);
T1.Start(); T2.Start();T3.Start(); T4.Start();T5.Start ();
```

```
}

void Scene1_relative() {

// le constructeur de Media text prend 5 paramètres : le texte à afficher,
// les coordonnées spatiales de l'objet, l'instant de début de l'objet,
// et l'instant de fin.

MediaText T1 = new MediaText("Institut",120,60,0,50);

//Valeur, X, Y, Début, Fin

MediaText T2 = new MediaText("National");

// les autres attributs ne sont pas spécifié

MediaText T3 = new MediaText("de Recherche");
MediaText T4 = new MediaText("en Informatique");
MediaText T5 = new MediaText("et Automatique");

DécaléSpatialement(T1,T2,20);
DécaléSpatialement(T2,T3,20);
DécaléSpatialement(T3,T4,20);
DécaléSpatialement(T4,T5,20);

// Dans ce cas, le programmeur doit implémenter un formateur temporel
// et spatial pour calculer les attributs sur tous les médias.

T1.Start(); T2.Start();T3.Start(); T4.Start();T5.Start ();

}

static public void main(String argV) {

CreerFenetreGraphique();
CreerMenu();
Scene1_relative(); // ou Scene1_absolue();

}
```

Figure II-5 : Spécification du document INRIA en Java

L'avantage majeur de ce type d'approche est que l'auteur peut décrire n'importe quel document multimédia, puisqu'il a à sa disposition le pouvoir d'expression d'un langage de programmation.

Les inconvénients des langages de programmation sont :

- Absence de facilité de spécification, l'auteur doit programmer tous les comportements qu'il désire dans le document. Il n'existe pas, à l'heure actuelle, de boîte à outils avec de tels comportements (par exemple avec un ensemble d'opérateurs prédéfinis, comme la mise en séquence d'une liste d'objets).
- L'absence de visualisation globale du document et notamment de sa structure, et de sa dimension temporelle.
- L'inaccessibilité de l'approche à des nons-informaticiens. Cependant nous présenterons les travaux réalisés pour permettre un accès plus large à la programmation dans la section 4.2.

3.3 L'approche événementielle

Les approches dites événementielles permettent de définir des règles du type : événements/ conditions/ actions. La sémantique associée à ces règles est : si l'événement est déclenché et que les conditions sont vérifiées alors les actions sont effectuées. Les événements peuvent être associés à la fin, au début d'un objet, au changement de comportement (changement d'un des attributs caractérisant l'objet : couleur, position,), ou encore à une action utilisateur (clic sur un objet).

Le langage de ce type le plus connu est MHEG où la description du comportement spatial et temporel se fait de cette manière. Par exemple, la description de la dimension temporelle se fait au moyen d'objets composites, de primitives de composition (qui permettent de donner un comportement temporel simple à tous les objets d'un composite) et de liens. Les liens ne sont pas de simples liens hypertextes, mais permettent de

spécifier des événements temporels et spatiaux. Un lien est constitué d'une partie condition (*LinkCondition*) et d'une partie action (*LinkEffect*) qui correspond à une liste d'actions à effectuer sur une liste d'objets. Pour illustrer de manière plus précise cette approche, nous allons présenter MHML qui est une variante de MHEG dont la syntaxe a été mise sous forme XML. Nous ferons aussi le parallèle avec le modèle IMD [Vazirgianis98] qui permet de manipuler l'historique de lecture du document.

Les événements MHML

Le langage MHML [MHML98] est un langage événementiel qui utilise une syntaxe XML [XML99].

Un document MHML se compose de trois parties :

- Une partie *Area*, qui permet de définir des zones réactives pour les médias.
- Une partie *Body* qui permet de définir des groupes d'objets dans le document (opérateur *DIV*). Cette notion de groupe est relativement faible et n'est utilisée de manière directe que pour associer des comportements (qui sont définis dans la partie *Behavior*). Sur chaque objet élémentaire du document, l'auteur peut définir un ensemble d'attributs permettant par exemple de définir le placement spatial de l'objet, le style,
- Une partie *Behavior* qui permet de définir le comportement du document.

Ce comportement est composé d'un ensemble de comportements élémentaires. Un comportement élémentaire est défini par :

- **Un événement**, qui sera déclenché par le système ou par un objet du document. Un événement peut être associé à la position d'un objet, à son état d'activité, et enfin à la sélection par le lecteur d'un objet dans le document.
- **Une condition,** qui permet d'évaluer un certain nombre de paramètres sur le système ou sur des objets du document. Les conditions couvrent le même éventail de valeur que les événements, c'est-à-dire que l'utilisateur a la possibilité de tester la position d'un objet, s'il est sélectionné ou pas, s'il est en train d'être exécuté ou non. Il peut aussi combiner les conditions via les opérateurs *AND* et *OR*.
- **Une liste d'actions**, qui sera à réaliser si l'événement est déclenché et si les conditions d'application des actions sont vérifiées. Dans la liste des actions que l'utilisateur peut décrire, on trouve le déclenchement et l'arrêt d'un objet ainsi que le changement d'attributs spatiaux.

La partie *Behavior* comporte un événement particulier, le *Start_Link* qui permet de définir les objets qui sont lancés au début de la présentation.

Dans l'exemple de la Figure II-6 nous présentons en MHML le même exemple de document que celui de la Figure II-3.

```
<?xml version="1.0"?><!DOCTYPE MHML PUBLIC "mhml.dtd" "mhml.dtd" >

<MHML>
<BODY>

<DIV y="0" x="0" width="640" height="480" id="Document"> <TXT y="0" x="0" width="100"
height="100" id="INRIA"/> INRIA </TXT>
       <DIV y="0" x="0" width="640" height="480" id="Groupe">
              <TXT y="180" x="80" id="INRIA0"/> Institut </TXT>
              <TXT y="240" x="100" id="INRIA1"/> National </TXT>
              <TXT y="300" x="120" id="INRIA2"/> de Recherche </TXT>
              <TXT y="360" x="140" id="INRIA3"/> en Informatique</TXT>
              <TXT y="420" x="160" id="INRIA4"/> et Automatique </TXT>
       </DIV>
</DIV>
</BODY>
<BEHAVIORS>
<START_LINK>
       <ACTION>
```

```
            <RUN propagation="true" target="Document"/>
      </ACTION>
</START_LINK>

<LINK div="Document">
      <RUN_EVENT target="Document" value="true"/>
      <ACTION>
            <RUN propagation="true" target="INRIA0"/>
      </ACTION>
</LINK>

<-- Evénement exemple -->
<-- Définition d'un événement sur le composite groupe
Cet événement est un événement temporel qui sera déclenché  au début du document (time equal
O). L'action associée à cet événement sera une action de présentation (run), qui aura pour
effet de démarrer l'objet INRIA0. La valeur propagation = true, signifie que si l'objet
INRIA0 est un objet composite, cet événement sera automatiquement propagé à ses fils. -->

<LINK div="Groupe">
<TIME_EVENT target="INRIA0" operator="equal" value="0" />
      <ACTION>
            <RUN propagation="true" target="INRIA1"/>
      </ACTION>
</LINK>
<-- fin de l'événement exemple -->

<LINK div="Groupe">
      <TIME_EVENT target="INRIA0" value="10" operator="equal"/>
      <ACTION>
      <RUN propagation="true" target="INRIA2"/>
      </ACTION>
      </LINK>
<LINK div="Groupe">
      <TIME_EVENT target="INRIA0" value="20" operator="equal"/>
      <ACTION>
      <RUN propagation="true" target="INRIA3"/>
      </ACTION>
      </LINK>



<LINK div="Groupe">
      <TIME_EVENT target="INRIA0" value="30" operator="equal"/>
      <ACTION>
      <RUN propagation="true" target="INRIA4"/>
      </ACTION>
      </LINK>
<LINK div="Groupe">
      <TIME_EVENT target="INRIA0" value="40" operator="equal"/>
      <ACTION>
      <RUN propagation="true" target="INRIA5"/>
      </ACTION>
      </LINK>
<LINK div="Document">
      <TIME_EVENT target="INRIA" value="50" operator="equal"/>
      <ACTION>
      <STOP propagation="true" target="INRIA"/>
      <STOP propagation="true" target="INRIA0"/>
      <STOP propagation="true" target="INRIA1"/>
      <STOP propagation="true" target="INRIA2"/>
      <STOP propagation="true" target="INRIA3"/>
      <STOP propagation="true" target="INRIA4"/>
      </ACTION>
      </LINK>
</BEHAVIORS>
</MHML>
```

Figure II-6 : Spécification événementielle du document INRIA en MHML

On peut noter que le modèle IMD proposé par Varzigianis [Vazirgianis98] proche de MHML permet en plus de tester l'historique des événements. Un des intérêts est de pouvoir ainsi faire évoluer le document en

fonction des actions précédentes de l'utilisateur. Cela se fait essentiellement par une extension de la partie condition sur les événements. Ces conditions sont une combinaison d'expressions booléennes et d'opérateurs sur les conditions de déclenchement des événements. Les opérateurs sont :

- Any (k,e_1, .., e_n) : vrai si au moins k événements parmi les n événements e_1, .., e_n ont été déclenchés.
- Seq(e_1, .., e_n) : vrai si les événements ont été déclenchés en séquence.
- Times (n, e_1) : vrai si l'événement e_1 est arrivé n fois.
- In(e_1 ,Int) : vrai si l'événement e_1 est intervenu dans l'intervalle Int.
- Not(e_1 ,Int) : vrai si l'événement e_1 n'est pas intervenu dans l'intervalle Int.
- S_CON(e_1, .., e_n) : vrai si les événements e_1, .., e_n sont arrivés en séquence.

Synthèse sur l'approche événementielle

Les langages événementiels ont un pouvoir d'expression très important et permettent de décrire tous les types de documents.

La difficulté majeure de cette approche réside dans la maîtrise du comportement du document. En effet, avoir une vue globale du comportement est relativement difficile à cause de l'aspect imprédictif de certains événements et de par le grand nombre d'événements qu'il faut spécifier pour décrire le comportement de ce document.

Aujourd'hui peu de systèmes auteur permettent d'avoir une édition évoluée de tels langages, on se ramène souvent à de la programmation classique (voir Lingo de Director). Les facilités de spécifications de l'auteur sont relativement faibles. Nous verrons dans le Chapitre VI, avec la présentation de MHML-Editor, ce qu'il est possible de faire pour aider l'auteur avec ce type de langage en lui offrant notamment, un aperçu des différents comportements possibles de son document.

3.4 L'approche relationnelle

Les approches relationnelles visent à faciliter l'écriture des documents multimédias en permettant à l'auteur de spécifier des relations entre les objets, par exemple :

- Enchaînement temporel de type séquence : les objets sont joués les uns après les autres.
- Position spatiale définie par des placements relatifs de type centrage, alignement,

L'auteur ne spécifie donc pas de manière absolue la position spatiale et temporelle de tous les objets, c'est le système qui, à partir de toutes les informations qu'il possède, calcule ce positionnement (étape de formatage). Les relations introduites peuvent être flexibles ou non.

Ce type de relation permet de définir un ensemble de placements possibles, les relations non flexibles permettent elles de définir des placements relatifs fixes. Par exemple, la relation *during* permet de spécifier qu'un objet A sera pendant un objet B, sans pour autant spécifier explicitement où se situent ces deux objets l'un par rapport à l'autre.

Le fait d'introduire des relations flexibles permet de modéliser des documents plus facilement adaptables.

Ces différents langages illustrant cette approche (CMIFed [Hardman93], ISIS [Kim95], TIEMPO [Wirag95]) permettent plus facilement à l'auteur de modifier le document : en effet, lors de l'ajout ou du retrait d'un objet, c'est le système qui s'occupe de calculer le nouveau placement de tous les objets. On peut noter dans ces approches deux catégories : la première basée sur des arbres d'opérateurs (SMIL, CMIF), permet à l'utilisateur de définir par exemple un arbre temporel où les noeuds représentent des relations et les feuilles des médias. Nous illustrerons cette approche par la présentation de SMIL (section 3.4.1). La deuxième catégorie d'approche, permet à l'auteur de définir des groupes d'objets ainsi que de définir des relations binaires entre les objets d'un même groupe (ISIS, TIEMPO, MADEUS [Layaïda97]). Nous présenterons Madeus (section 3.4.2) pour illustrer cette approche.

3.4.1. Les arbres d'opérateurs : SMIL

Nous allons illustrer cette approche à l'aide du langage SMIL [SMIL98].

SMIL est une recommandation du World Wide Web Consortium (W3C), sa version courante est la 1.0. Sa syntaxe est décrite par une DTD XML. La version 2.0 de SMIL est prévue pour février 2001 [SMIL2-00].

L'objectif de SMIL 1.0 est de faciliter l'intégration de médias de différents types en fournissant un moyen de spécifier à la fois les dimensions spatiale et temporelle du document.

Nous allons rapidement en présenter les principales caractéristiques : un fichier SMIL 1.0 (Figure II-7) est composé d'un élément *head* et d'un élément *body*. La partie *head*, qui correspond à l'en-tête du fichier permet de spécifier les informations non temporelles du document. Par exemple, le positionnement spatial des objets se fait au moyen de régions. Une *région* définit une zone d'affichage à l'écran. Cette zone d'écran peut être partagée par plusieurs objets multimédias. Dans l'exemple de la Figure II-7, la *région* "droite" est partagée par deux objets (une image et un texte) décrits dans la partie *body*.

La partie *body*, qui correspond au corps du document, contient la description de l'ensemble des objets manipulés dans le document, la description de leur organisation temporelle et la définition des liens hypertexte contenus dans le document.

```
<?xml version="1.0"?>

<smil>
<head>
<layout>
       <region id="région1" top="180" left="80" width="100" height="40" />
       <region id="région2" top="240" left="100" width="100" height="40" />
       <region id="région3" top="300" left="120" width="100" height="40" />
       <region id="région4" top="360" left="140" width="100" height="40" />
       <region id="région5" top="420" left="160" width="100" height="40" />
</layout>
</head>

<body>
<switch id="Contentenu adaptatif">
       <par dur="50" id="Document" system-bitrate="115200">
              <text src="/home/Texte1.html" id="T1" region="region1"/>
              <text src="/home/Texte2.html" id="T2" begin="10.0"
              region="region2"/>
              <text src="/home/Texte3.html" id="T3" begin="20.0"
              region="region3"/>
              <text src="/home/Texte4.html" id="T4" begin="30.0"
              region="region4"/>
              <text src="/home/Texte5.html" id="T5" begin="40.0"
              region="region5"/>
       </par>
       <par dur="50" id="Document" system-bitrate="144000">
              <sound src="/home/MusicINRIA.mp3" >
              <text src="/home/Texte1.html" id="T1" region="region1"/>
              <text src="/home/Texte2.html" id="T2" begin="10.0"
              region="region2"/>
              <text src="/home/Texte3.html" id="T3" begin="20.0"
              region="region3"/>
              <text src="/home/Texte4.html" id="T4" begin="30.0"
              region="region4"/>
              <text src="/home/Texte5.html" id="T5" begin="40.0"
              region="region5"/>
       </par>
</switch>
</body>
</smil>
```

Figure II-7 : Spécification relationnelle du document INRIA en SMIL

L'organisation de la partie *body* des documents SMIL est basée sur une structure hiérarchique dont les noeuds sont des opérateurs temporels et les feuilles des médias. L'opérateur temporel associé à un noeud décrit la façon dont s'enchaînent temporellement les fils du noeud. Les opérateurs peuvent être soit l'opérateur *par*, soit l'opérateur *seq*. L'opérateur *par* permet de décrire une exécution en parallèle des fils, l'opérateur *seq* permet de décrire une exécution en séquence des fils. Dans l'exemple de la Figure II-7 on peut voir un exemple d'utilisation de ces deux opérateurs.

L'auteur peut compléter/modifier l'information déduite des opérateurs en spécifiant des attributs sur les médias. Par exemple, l'attribut *dur* permet de spécifier explicitement la durée d'un objet, alors que par défaut les objets discrets ont une durée infinie. Les attributs *begin* et *end* permettent, eux, de spécifier de façon absolue ou relative (par rapport au début ou à la fin d'autres objets dans le document) le début ou la fin des objets. Le placement de ces attributs permet de définir des synchronisations temporelles plus fines entre les objets.

Exemple a :

```
<par>
       <img src="" id="A" dur="6s">
       <img src="" id="B" dur="4s">
       <img src="" id="C" dur="3s">
</par>
```

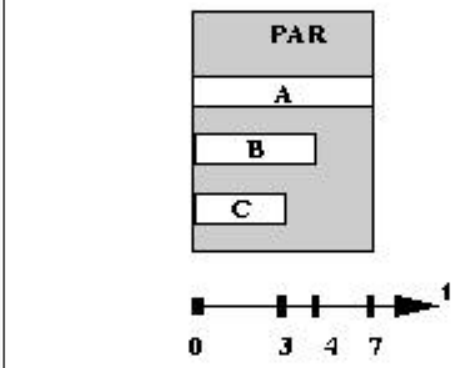


Figure II-8 : L'attribut dur de SMIL

Dans l'exemple de la Figure II-8, on peut voir un exemple de noeud *par* avec trois fils. Sur chacun de ses fils, l'attribut *dur* est spécifié de façon explicite, comme il n'y a pas d'attribut *begin* de spécifié, les trois objets commenceront en même temps. Cela signifie qu'on laisse aux trois objets le placement défini par défaut par le noeud *par*. Les trois objets vont se jouer en parallèle, ils commenceront en même temps, et ils se termineront quand leur durée respective sera écoulée.

Exemple b :

```
<par>
       <img src="" id="A" dur="6s" begin="3s">
       <img src="" id="B" dur="4s">
       <img src="" id="C" dur="3s"
       begin="end(B)">
</par>
```

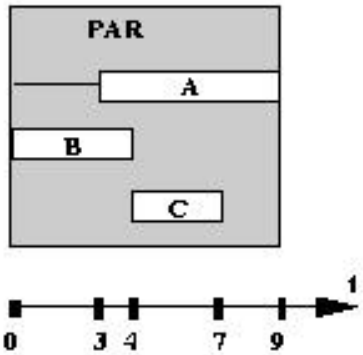


Figure II-9 : Les attributs begin et end de SMIL

Dans l'exemple de la Figure II-9, on peut voir trois manières de spécifier le début d'un objet :

- de façon absolue, il permet de retarder le début effectif de l'objet ;
- de façon implicite, dans ce cas, l'objet démarre en même temps que son ascendant (si c'est un noeud *par*) ;

- de façon relative, en liant le début de l'objet au début ou à la fin d'un autre objet.

Le comportement par défaut a été complètement modifié par la définition de l'attribut *begin* sur les objets A et C. Le début de l'objet A a été retardé de trois secondes par rapport au début du noeud *par* et l'objet C commence à la fin de l'objet B.

L'attribut *endsync* sur l'élément *par*, permet de lier la fin de l'élément composite à la fin d'un des objets qui le compose. Par exemple, la valeur *first* pour l'attribut *endsync* signifie que l'objet *par* se termine avec la fin du premier objet qui le compose. Dans le cas où les fils sont des objets déterministes cette valeur est connue statiquement, dans le cas d'objets indéterministes cette valeur ne sera connue que dynamiquement.

En résumé, la structure d'opérateurs permet à l'auteur de spécifier des comportements temporels basiques. Cependant, pour décrire des comportements temporels plus complexes, l'auteur doit manipuler les attributs *begin* et *end* sur les objets.

On peut noter que le langage SMIL offre un attribut pour permettre un certain type d'adaptation. L'attribut *Switch*, permet de définir une alternative en fonction des conditions de présentation du lecteur. Par exemple, cela permet de définir deux branches au document, une si l'utilisateur utilise un modem, l'autre s'il est sur un réseau local par exemple. Dans chacune des branches, on peut utiliser des médias différents et adaptés au contexte. Cet opérateur permet de définir une adaptation sommaire qui revient en partie à décrire statiquement les différents comportements possibles. On peut voir dans l'exemple de la Figure II-7 l'utilisation de cet élément. Dans le cas où le lecteur a un débit réseau supérieur à 144000 baud/s nous ajoutons un élément audio à la présentation.

Dans l'exemple de la Figure II-7, nous avons défini en SMIL le document INRIA en privilégiant l'utilisation de valeurs d'attribut relatives (nous avons défini que tous les objets finissaient en même temps par le positionnement de l'attribut *dur* sur le noeud Document). En effet, nous aurions pu construire le même document en n'utilisant que des valeurs absolues (positionnement des attributs *dur* sur tous les médias). L'avantage d'une spécification plus relative est qu'elle est plus facilement modifiable. Ainsi si nous voulons insérer un nouvel objet "INRIA", présent tout au long de la scène, nous avons juste à le spécifier (avec l'instant de début désiré) dans l'objet *par*. Nous n'avons alors pas à modifier la durée des objets, celle-ci étant définie de façon relative.

On peut noter aussi que l'approche à base d'arbres d'opérateurs n'est pas vraiment une approche relationnelle pure, puisqu'un objet ne peut être impliqué que dans une seule relation. De ce fait, lorsque l'on désire ajouter une relation, on est souvent amené à restructurer l'arbre d'opérateurs.

La difficulté majeure avec ce genre d'approche est de bien concevoir son document pour permettre des modifications futures. Cependant, d'une part, l'auteur ne peut prévoir a priori toutes les modifications futures qu'il voudra apporter à son document, d'autre part il arrive parfois qu'un bon document pour une modification ne le soit pas pour une autre modification. Pour toutes ces raisons les remises en cause de la structure de l'arbre d'un document SMIL sont fréquentes, et nous savons que restructurer un arbre n'est pas une chose simple [Quint87].

Cependant, un des avantages majeurs de cette approche est qu'elle permet de définir rapidement un comportement temporel simple.

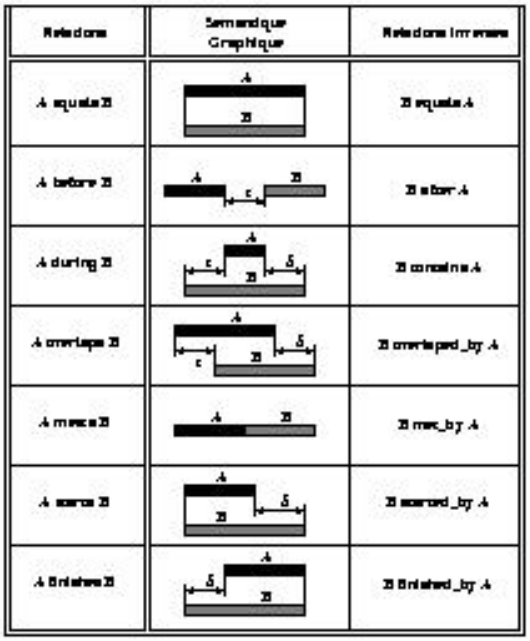
On peut noter pour finir qu'un langage comme SMIL permet de définir des documents intégrant des médias indéterministes tout en gardant certaines synchronisations (attribut *endsync*).

3.4.2 Une approche à base de relations binaires : Madeus

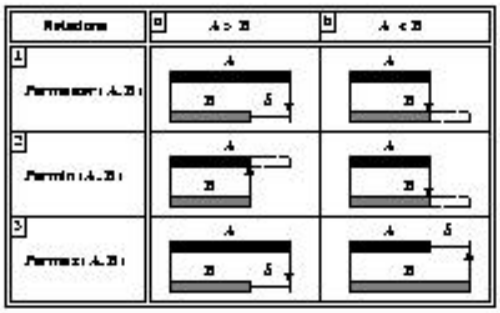
Le langage Madeus [Layaïda97] a été développé au sein du projet Opéra. Les documents Madeus sont composés de 4 sections :

- **La section média** : permet de définir les médias qui seront utilisés dans le document. Ces médias seront référencés dans les sections spatiales et temporelles.

- **La section spatiale** : permet de définir les attributs de présentation graphique de l'objet (fonte, couleur, placement, etc.). On peut aussi associer à chaque objet ou groupe d'objets des relations spatiales pour définir un placement relatif entre chaque élément.
- **La section temporelle** : permet de définir la position temporelle de tous les objets du document en plaçant des relations temporelles entre les objets ou entre groupes d'objets. Les relations sont basées sur les relations d'Allen quantifiées (Figure II-10a) augmentées de trois relations supplémentaires exprimant des interruptions (Figure II-10b).
- **La section hypertexte** : permet de définir des liens entre les objets ; ils peuvent être entre documents ou temporels. Cette section est incluse dans la partie temporelle. L'auteur peut aussi définir des liens temporels. Un lien temporel permet d'aller vers un instant donné de la présentation.



a) Les relations d'Allen



b) Les relations causales

Figure II-10 : Les relations temporelles de Madeus

Si nous reprenons l'exemple de notre petite animation présentant l'INRIA (Figure II-11) dans le langage Madeus, on peut voir les trois parties du document : la description des médias, la description de la structure temporelle et spatiale. Dans ces deux structures, le positionnement des objets est défini par des relations entre les objets (comme *Finishes*, *AlignéAGauche*). La principale différence avec SMIL est que l'on a des groupes d'objets et non pas un arbre d'opérateurs. De ce fait si l'on désire changer le placement temporel des objets pour réaliser deux séquences en parallèle, il suffit de changer les relations entre les objets, et l'on n'a pas à changer la structure du document. Une limite du langage réside dans la localité des relations, une relation ne peut porter que sur des objets présents dans le même groupe. Cette limitation permet de conserver une localité à la spécification. L'auteur n'a ainsi pas à comprendre la structure complète de son document pour comprendre le comportement local de ses médias.

```
<Madeus>

<Media>
       <text Id= "O1" value="Institut" >
       <text Id= "O2" value="National" >
       <text Id= "O3" value=" de Recherche " >
       <text Id= "O4" value="en Informatique" >
       <text Id= "O5" value="et Automatique" >
</Media>

<Spatial>
       <Groupe Id="Animation">
              <Zone ID="z1" media="O1" X="60" Y="120">
              <Zone ID="z2" media="O2">
              <Zone ID="z3" media="O3">
              <Zone ID="z4" media="O4" >
              <Zone ID="z5" media="O5" >
       <Relation>
              <AligneAGauche param1="z1" param2="z2" delta="+60">
              <AligneAGauche param1="z2" param2="z3" delta="+60">
              <AligneAGauche param1="z3" param2="z4" delta="+60">
              <AligneAGauche param1="z4" param2="z5" delta="+60">
              <Under param1="z1" param2="z2" delta="+20">
              <Under param1="z2" param2="z3" delta="+20">
              <Under param1="z3" param2="z4" delta="+20">
              <Under param1="z4" param2="z5" delta="+20">
       </Relation>
       </Groupe>
</Spatial>

<Temporal>
       <Groupe Id="AnimationT" dur="50s">
              <Zone ID="T1" X="60" Y="120" media="O1">
              <Zone ID="T2" media="O2">
              <Zone ID="T3" media="O3">
              <Zone ID="T4" media="O4">
              <Zone ID="T5" media="O5">
       <Relation>
              <finishes param1="z1" param2="z2" delay="10s">
              <finishes param1="z2" param2="z3" delay="10s">
              <finishes param1="z3" param2="z4" delay="10s">
              <finishes param1="z4" param2="z5" delay="10s">
       </Relation>
</Groupe>
</Temporel>
</Madeus>
```

Figure II-11 : Spécification relationnelle du document INRIA en Madeus

Le langage Madeus ne permet de décrire que des documents prédictifs avec des liens de navigation. L'introduction des objets indéterministes lève de nouveaux problèmes pour le formatage ([Layaïda97]) non encore résolus de manière satisfaisante à ce jour.

Un des avantages majeurs de l'approche est qu'elle permet à l'auteur de décrire des relations flexibles. Par exemple, cela permet d'utiliser ces informations pour adapter automatiquement le document au contexte de présentation. Nous verrons au cours de la présentation du système Madeus (section 4.3.3), des travaux réalisés dans ce sens.

Un autre avantage est la facilité de modification due en partie aux intervalles de durées définis sur les objets. En effet, cela permet au système d'adapter la durée des objets au cours du processus d'édition (en fonction de l'intervalle donné par l'auteur) évitant ainsi à l'auteur de modifier la durée et le placement de ces objets lors de chaque modification de son document.

3.4.3 Synthèse sur les approches relationnelles

La principale limite dans les approches relationnelles actuelles est le pouvoir d'expression. C'est-à-dire

qu'elles ne permettent pas de spécifier des interactions complexes entre l'utilisateur et le document, ou des enchaînements complexes comme des répétitions conditionnelles. Cependant, si l'on reste dans le cadre de documents prédictifs, l'extension de ces langages à une expressivité plus grande ne pose pas de problème. Dans ce cas, les langages relationnels ont le même pouvoir d'expression que des langages comme MHML (restreint à un comportement prédictif). L'avantage par rapport à des approches événementielles est la présence de relations de haut niveau qui permettent de définir des comportements entre les objets. Cela permet d'avoir une description du comportement plus concise. De plus la sémantique de la structuration du document est plus forte. On peut noter une différence entre l'approche de SMIL où la structuration est une structuration temporelle à base d'arbre d'opérateurs, et celle de Madeus où il existe une structure temporelle indépendante des relations temporelles à base de groupes. La structure à base d'arbre d'opérateurs peut être un avantage pour la description de documents simples [Bétrancourt99], cependant la structure de groupes permet une plus grande souplesse pour les modifications et l'évolution du document.

Un autre avantage de ces approches est la définition des objets dans des intervalles de valeurs. Cela rend la spécification plus facilement adaptable, et permet d'automatiser le processus de formatage. Une proposition est faite par Kim [Song99] pour intégrer cette notion dans SMIL-2.0 après l'avoir intégrée dans MPEG 4.

L'approche relationnelle relativement récente n'a pas encore fait ses preuves, notamment pour des documents complexes. Cependant, du point de vue de l'édition, cette approche semble offrir de plus grands potentiels du fait du bon compromis qu'elle offre entre facilité de spécification (et de réédition) et pouvoir d'expression [Jourdan98c]. Parmi les outils mettant en oeuvre une telle approche, on peut citer Grins[Bulterman00] et Madeus que nous présenterons dans les sections 4.3.2 et 4.3.3. Elle a suscité de nombreux travaux de recherche ([Layaïda97], [Kim95], [Hardman93], [Wirag95]) du fait des nombreux aspects non encore résolus qu'elle soulève, comme le formatage efficace ou dynamique et la cohérence. En effet, à ce jour il n'existe pas de système auteur basé sur ce type d'approche offrant les services d'édition adaptés.

3.5 Synthèse sur les langages de documents multimédias

Nous venons de voir que l'auteur de documents multimédias dispose d'un ensemble de formalismes pour spécifier ses documents. Au cours de cette analyse, nous avons pu voir les différences qu'il existe entre ces approches en terme d'expressivité et d'utilisation :

- Les approches absolues permettent la description de documents sans synchronisation temporelle relative. Cela signifie que dans le cas de médias imprédictifs l'auteur ne peut définir de synchronisation temporelle avec d'autres médias. Elles sont limitées en expressivité, mais sont simples d'utilisation.
- Les approches programmation et événementielle sont très expressives, mais difficiles d'utilisation. Ces approches permettent de décrire les trois classes de documents décrites dans la section 2.1.1.
- Les approches relationnelles offrent le meilleur compromis entre expressivité et simplicité du fait qu'elles permettent de spécifier un large éventail de documents (prédictif avec navigation) de manière relativement simple et intuitive. Parmi ces approches, on peut distinguer deux sous-classes :
    - Les arbres d'opérateurs, qui au profit de la simplicité rendent certaines opérations d'édition plus complexes. Notons que dans SMIL 2.0 il sera intégré de manière explicite les objets indéterministes et les interactions.
    - Les relations dans des groupes, qui offrent une plus grande souplesse de modification. Dans Madeus des travaux pour intégrer l'indéterminisme sont en cours.

On peut noter que l'on peut combiner ces deux approches (voir Chapitre IV). Nous allons maintenant présenter les différents outils d'édition de documents en nous intéressant essentiellement aux formalismes proposés par ceux-ci. Ceci afin de voir comment ils répondent aux besoins définis dans la section 2.

4. Les outils d'édition

L'objectif est d'identifier les fonctions générales offertes par les environnements d'édition pour les confronter aux besoins vus en section 2.2. Cependant, cet exercice présente un certain nombre de limites du fait de la dépendance entre l'outil et le langage de sauvegarde utilisé.

Les outils d'édition de documents multimédias sont issus de plusieurs origines, les principales sont listées ci-dessous en fonction de la nature des données manipulées :

- *Les documents textuels* : comme Thot[Quint99] ou Word qui permettent d'introduire tous les types de médias dans leurs documents (sans synchronisation temporelle entre eux) ainsi que la définition de synchronisations spatiales simples entre ces médias (alignement, centrage).
- *Les documents hypertextes* : avec l'avènement d'internet, des outils accompagnent l'émergence de standards (DHTML[DHTML98], flash[Macromedia-flash00], SMIL), c'est le cas par exemple de RealNetworks [Real98] et d'outils comme SmilWizard [Real99].
- *Les bases de données* : qui se tournent de plus en plus vers des outils plus complets, outils qui permettent aujourd'hui de construire des requêtes complexes et de les visualiser sous une forme multimédia [Martin99].
- *Les documents multimédias*: depuis l'avènement de ce type de documents, des outils ont permis de les éditer et ont évolué avec les besoins des auteurs, c'est le cas par exemple de Director et de Toolbook [Asymetrix99] qui offrent des outils complets.
- *Les langages de programmation* : qui offrent de plus en plus de facilités pour manipuler les médias et l'émergence de boîtes à outils complexes, c'est par exemple le cas de Java et des JMF.

À partir de ces différentes origines nous avons choisi de classer les environnements en trois grandes catégories que nous présenterons ensuite :

- Les approches à base de schémas, où les auteurs ne font que remplir des documents prédéfinis. Cette catégorie regroupe les outils issus des documents structurés qui ont eut le souci de simplifier au maximum la tâche de l'auteur (section 4.1).
- Les approches à base de programmation, où l'auteur spécifie un document comme une application à part entière (section 4.2).
- Les approches classiques, issues de l'édition des documents textuels ou multimédias, où l'auteur définit de bout en bout son document avec l'aide du système (section 4.3).

4.1 Les environnements auteur à base de modèles prédéfinis

La première approche que nous allons présenter est celle qui utilise des schémas prédéfinis. L'idée générale de ces outils est de fournir à l'auteur des types de présentations prédéfinies, comme, par exemple, une succession d'images avec un fond sonore à partir duquel il n'a plus qu'à instancier les médias. L'intérêt de ces approches est de simplifier au maximum l'édition de documents multimédias en fournissant des modèles prédéfinis pour les principales classes de documents, en évitant ainsi à l'auteur d'avoir à manipuler un document multimédia dans toute sa complexité.

4.1.1 PowerPoint

Le premier exemple, sans doute le système le plus connu, combine une approche classique d'édition et une édition à base de schéma. Pour la création de transparents, le système offre à l'auteur un ensemble de transparents prédéfinis (Figure II-12). L'auteur n'a alors plus qu'à remplir les différents champs de son transparent. Le système permet à l'auteur de modifier la présentation de son transparent à souhait, en enlevant ou en rajoutant certains champs. Le schéma temporel entre les différents transparents reste élémentaire (séquence), cependant l'auteur peut obtenir un schéma temporel plus évolué à l'intérieur de ses transparents, en décrivant des comportements temporels simples (séquence, animation). Pour cela, il dispose d'un ensemble

d'animations prédéfinies, ainsi que de la possibilité de décrire l'ordre d'apparition des objets.

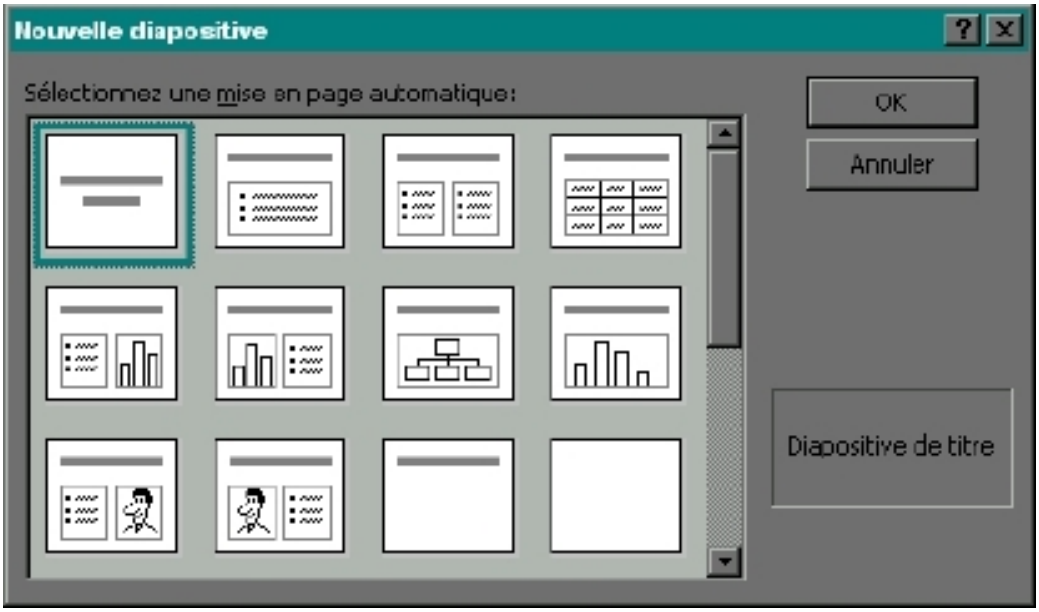


Figure II-12 : Le jeu de transparents prédéfini de PowerPoint

4.1.2 Smil Wizard

Le deuxième exemple d'outil à base de schémas prédéfinis est SMIL Wizard [Real99] (Figure II-13). L'objectif de ce système est d'offrir un moyen simple d'écrire des documents SMIL pour des personnes n'ayant aucune connaissance du langage. Le système propose un ensemble de présentations prédéfinies : Karaoké, succession de transparents, présentation de photographies,
L'auteur choisit le type de présentation (Figure II-13a) qu'il désire, et ensuite il spécifie les informations sur la localisation des médias (Figure II-13b).
Le système génère ensuite un document SMIL présentant les informations comme l'auteur l'a souhaité.
L'inconvénient majeur de ce type d'environnement est qu'il ne permet pas de définir ses propres présentations ou de sortir légèrement de la présentation standard proposée par le système d'édition. Un autre inconvénient est la qualité du fichier produit. Le système produit lors de la génération automatique un fichier utilisant peu les possibilités du langage, notamment pour la structuration de l'information (le fichier produit est constitué d'un élément *par*, et tous les objets sont placés de manière absolue grâce à des attributs *begin*), de ce fait, il est difficilement rééditable par un auteur.

Figure II-13 : Les vues d'édition de Smil Wizard

4.1.3 RealSlideShow

Le dernier environnement que nous présentons est RealSlideShow [Real98], il permet de générer des présentations sous forme de transparents. Chaque transparent peut contenir une image et du texte. L'auteur peut choisir la durée de chacun de ces transparents, et de mettre un son pendant la présentation des transparents.

On peut voir, dans la Figure II-14 les deux vues de RealSlideShow :

- la *vue temporelle* (Figure II-14a) qui permet de définir les transitions entre les transparents et la durée de chacun des transparents ;
- *la vue région* (Figure II-14b) qui permet de définir un agencement spatial pour les objets.

Comme pour le système précédent, l'auteur est guidé lors de la conception de sa présentation, mais dans ce cas-ci, il peut modifier certains paramètres de la présentation, comme l'agencement spatial des objets.

Figure II-14 : Les vues d'édition de RealSlideShow

Le langage de sortie de RealSlideShow est un langage propriétaire qui ne peut être lu que par les outils de RealNetworks. C'est à dire que les médias utilisés sont dans un format propriétaire.

4.1.4. Synthèse sur les environnements à base de modèles de présentation prédéfinis

Les environnements d'édition à base de modèles prédéfinis permettent à l'auteur d'écrire facilement des documents multimédias. Du point de vue de l'édition, le système offre très peu de possibilités à l'auteur pour spécifier un placement qui n'est pas prédéfini. Les besoins en visualisation et en aide sont moins importants que dans le cadre d'un environnement classique : l'auteur ne peut pas ici spécifier de documents incohérents du fait des restrictions et des contraintes de manipulations permises à l'auteur. L'édition devient minimale. Le système d'édition réduit volontairement le pouvoir d'expression du langage sous-jacent pour simplifier la tâche de l'auteur. Dans le cas de PowerPoint, la seule possibilité de spécification d'un comportement non défini est d'utiliser un formalisme de placement absolu. Dans le cas des deux derniers outils, les possibilités de l'auteur sont tellement réduites qu'il n'a que peu de moyen pour modifier son document une fois le processus d'édition fini. En effet, le document de sauvegarde ne contient aucune des informations qui ont permis l'édition du document.

De plus, les documents produits sont de qualité relativement pauvre, et peuvent être difficilement réutilisés dans un autre contexte. L'utilisation de modèles logiques de documents multimédias (structuration du document indépendamment de sa présentation), de par l'expérience acquise dans l'édition de documents textuels semble une voie à explorer, voie qui à ce jour est complètement inexploitée par les outils actuels qui se contentent d'offrir des modèles de présentation.

4.2 Les environnements de programmation

Nous avons vu dans la section précédente que les langages de programmation permettaient à l'auteur un grand pouvoir d'expression ; la contrepartie est liée aux compétences en informatique nécessaires :

- Problèmes d'inaccessibilité aux non-informaticiens.
- Problèmes de mise au point, l'auteur doit utiliser des débogueurs classiques, le contexte d'utilisation (c'est-à-dire l'écriture de documents multimédias) n'est pas exploité. Par exemple, on pourrait imaginer des modules spécialisés pour aider l'auteur dans ses spécifications de synchronisations spatiales et temporelles.
- Absence de WYSIWYG, c'est-à-dire qu'avant de visualiser son document, l'auteur doit le compiler.

Cependant des techniques existent pour réduire ces difficultés, on peut citer, par exemple, la programmation

visuelle (section 4.2.1) et la programmation par démonstration (section 4.2.2).

4.2.1 Les langages de programmation visuelle

Des travaux ont été réalisés pour faciliter l'accès à la programmation à des non-informaticiens. L'objectif de la programmation visuelle est de fournir une interface pour décrire de façon simple et plus conviviale des programmes. On peut citer comme exemples d'environnements Venn [DelBimbo96] et dans le cadre des documents multimédias Icon Author [AimTech96]. Dans l'exemple de la Figure II-15, on peut voir un exemple de spécification de document. L'auteur a défini une boucle avec une succession de calculs puis un affichage d'objets qui sera interrompue par un clic de l'utilisateur. L'auteur édite son document au travers de palettes d'opérateurs, et pour chaque opérateur, il a la possibilité de définir un certain nombre de paramètres.

Cependant on peut noter qu'en facilitant la tâche de l'auteur, ces méthodes limitent l'expressivité du langage et le pouvoir de l'auteur. En effet, on peut difficilement spécifier de gros programmes et/ou des programmes complexes par ce biais du fait des difficultés de perception liées à l'affichage et au volume d'information que l'on peut visualiser.

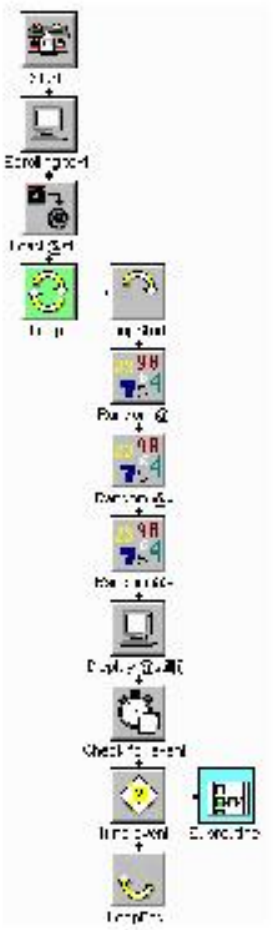


Figure II-15 : Logigramme d'un programme réalisé avec IconAuthor

Les limitations des environnements de programmation visuelle sont du même ordre que pour les programmes classiques :

- Difficulté d'écrire de gros programmes/documents. L'auteur a du mal à avoir une perception globale du comportement de son document. De plus, peu de techniques ont été développées pour améliorer l'affichage et permettre une meilleure visualisation des informations.
- Difficulté de description de comportements complexes. La description d'un gros document entraîne une complexité visuelle difficilement manipulable pour un auteur.
- Difficulté de spécification de document réactif. Il est difficile pour un auteur de manipuler un

document contenant de nombreuses branches, de par la complexité visuelle de la représentation.

4.2.2 La programmation par démonstration

L'objectif de la programmation par démonstration est de faciliter la spécification de comportements complexes ou répétitifs. Par exemple, l'auteur exécute une fois la tâche, et le système automatise la tâche par le biais de macros. L'utilisateur, quand il aura besoin d'exécuter une autre fois la tâche, fera appel à la macro. Dans le cadre de Toolbook [Asymetrix99] l'utilisateur peut montrer un déplacement d'objet à l'écran à l'aide de la souris, et le système écrit pour l'auteur le code correspondant dans les langages de scripts de Toolbook. Actuellement, la programmation par démonstration vise donc à aider l'auteur dans l'écriture de scripts. Elle pourrait être, par exemple, utilisée pour l'aider à spécifier le comportement temporel de son document. Par exemple, dans Director, pour spécifier un déplacement complexe, l'auteur peut dessiner à l'aide de fonctions prédéfinies le déplacement de son objet, le système en déduira le placement de l'objet réel à chaque instant. Ces techniques fournissent des idées pour la phase de visualisation ainsi que sur les modes d'action pour l'édition. Ces méthodes sont souvent utilisées en complément dans des outils plus complets (Toolbook).

4.2.3. Synthèse sur les environnements de programmation

Les environnements de programmation sont encore réservés aux informaticiens. L'utilisation d'outils utilisant des métaphores graphiques ou des méthodes de programmation par démonstration, facilite l'accès des non-informaticiens à de tels outils. Cependant, pour atteindre cet objectif, ces outils réduisent le pouvoir d'expression de l'auteur. De plus, un autre inconvénient de ces approches est qu'il n'existe pas d'aide pour l'auteur de documents multimédias (librairies, modèles, classes,).

Cependant, on peut imaginer des améliorations possibles de ces techniques en fournissant à l'auteur un ensemble de classes permettant de définir des fragments de documents. De plus, on pourrait imaginer une extension des techniques de programmation par démonstration, de manière à permettre au système de déduire des relations à la place de simplement noter des propriétés absolues sur les médias. Ces extensions peuvent s'inspirer des techniques développées dans Garnet [Sannella92] qui reposent sur des mécanismes à base de contraintes pour reconnaître des propriétés. Cela faciliterait la relecture des programmes produits et rendrait le document généré plus flexible.

4.3 Les environnements auteur complets

La troisième catégorie de systèmes auteur laisse plus de latitude à l'auteur, tout en offrant des moyens de l'aider dans la tâche complexe qu'est la création de documents. Ces outils sont dits complets, car ils permettent à l'auteur de spécifier toutes les caractéristiques du document à toutes les étapes du processus d'édition. Ces outils mélangent le plus souvent les formalismes d'édition, pour permettre différents moyens de spécification.

4.3.1 Director

Director est un des premiers outils dédié à l'édition de documents multimédias. Il est apparu à la fin des années 80 et ne cesse de changer pour intégrer les évolutions des documents et améliorer la phase d'édition.

La phase d'édition dans Director est basée sur quatre étapes :

- *Création d'un objet de base* : cet objet peut être une image, un texte, un son.
- *Placement temporel d'un objet* : il s'effectue sur une vue temporelle (Figure II-16) où l'auteur place de façon absolue l'objet. C'est-à-dire qu'il spécifie explicitement l'instant de début de son objet et la durée de celui-ci. Ce type de placement ne permet pas de définir des synchronisations évoluées entre les objets ni de faire des placements relatifs d'un objet par rapport à un autre.

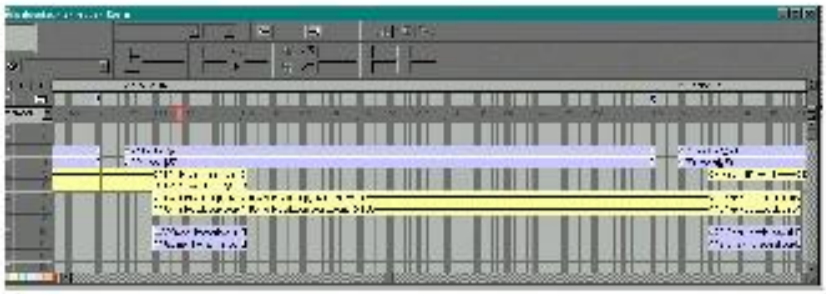


Figure II-16 : Vue temporelle de Director

- *Placement spatial d'un objet* : le placement spatial s'effectue de façon absolue dans la fenêtre de présentation (Figure II-17).

Figure II-17 : : Vue de présentation de Director

- *Description fine de la synchronisation temporelle* et des événements liés aux interactions des utilisateurs : cette description se fait au moyen d'un langage de programmation (Lingo [Macromedia-Lingo00]) dont on peut voir un exemple dans la Figure II-18. Dans cet exemple, on peut voir un comportement défini pour garder une scène à l'écran. Les scripts sont par exemple utilisés pour définir un bouton qui permettra de déclencher le début d'une vidéo. L'utilisateur peut associer un comportement à plusieurs types d'événements : clic utilisateur, changement de comportement d'un objet. L'utilisation de scripts augmente la complexité de la spécification dans Director. En effet, même si le langage de scripts est relativement simple, une utilisation évoluée nécessite des compétences informatiques.

```
--------------------------------------------------------------------------------
-- Attente
-- Ce script permet de garder affiché la scène courante, pendant Howlong secondes.
--------------------------------------------------------------------------------

property howLong, currentTime --déclaration des variables
  on prepareFrame -- événement associé à la création de la scène
  currentTime = the timer
end

on exitFrame --événement associé à la fin de la scène
  repeat while the timer < currentTime + howlong
    nothing
  end repeat
end

-- mise à jour des propriétés et initialisation du vérificateur de propriétés.
on getPropertyDescriptionList
  propertyDescriptionList = [ ¬
  #howLong: [ #comment: "durée d'attente (1/60e seconde):"
  #format: #integer,
  #default: 10
  return propertyDescriptionList
end

on getBehaviorDescription
  return "garde la scène présente à l'écran en fonction de la durée spécifiée."
End
```

Figure II-18 : Exemple de script dans Director

À tout moment de l'édition, l'auteur peut visualiser les objets présents dans son document (Figure II-19). De plus lors des opérations d'édition, le système donne des informations complémentaires à l'auteur : par exemple dans la Figure II-17, lorsque l'auteur manipule un objet dans la vue de présentation, le système affiche des informations : coordonnées absolues de l'objet ainsi qu'un conseil pour réaliser une opération.

Figure II-19 : Vue objet de Director

Director est un environnement d'édition complet, qui, à partir des informations auteur, génère un fichier compilé, qui servira de support à l'exécution du document. Ce fichier contient des informations spécifiques au système d'exploitation pour lequel il est destiné. Cette approche limite donc la portabilité du document produit. Ce qui dans le cas de Director limite la diffusion du document, étant donné qu'il n'existe pas de machine de présentation sur tous les systèmes d'exploitation. On peut noter que l'environnement ne fournit aucun moyen d'accéder aux informations liées aux périphériques de présentation. Il est donc impossible à l'auteur de faire des documents adaptables sans intervention du lecteur.

Par rapport aux différents formalismes présentés précédemment, Director offre donc à la fois une édition via un formalisme absolu et via un formalisme à base d'un langage de programmation. Cette double combinaison est intéressante du point de vue de l'expressivité, néanmoins dans les deux cas, elle rend difficile la maintenance du document au cours du cycle d'édition.

Du point de vue du cycle d'édition, l'aspect important dans Director est la présence d'une vue temporelle qui permet à l'auteur d'avoir une idée des enchaînements de son document.

4.3.2 Grins

Grins est un éditeur de documents SMIL qui est issu des travaux menés par l'équipe du CWI d'Amsterdam autour de CMIFed [Hardman93]. Il est en cours de commercialisation par la société Oratrix. Grins permet à l'auteur de spécifier de manière indépendante les informations spatiales et temporelles.

Cet éditeur est composé de quatre vues :

- Une vue temporelle hiérarchique (Figure II-20a) qui permet de spécifier la structure temporelle du document : cette vue permet de représenter la hiérarchie temporelle sous forme de boîtes englobantes, l'axe vertical servant à représenter le placement temporel des objets à l'intérieur d'un noeud : en séquence (les objets sont affichés les uns en dessous des autres) ou en parallèle (les objets sont affichés les uns à côté des autres).
- Une vue des régions (Figure II-20b) pour définir le placement spatial des objets.
- Une vue temporelle (Figure II-21) qui est générée à partir des informations de la vue temporelle hiérarchique et des attributs que l'auteur a pu placer sur les différents médias.

- Une vue de présentation pour exécuter le document.

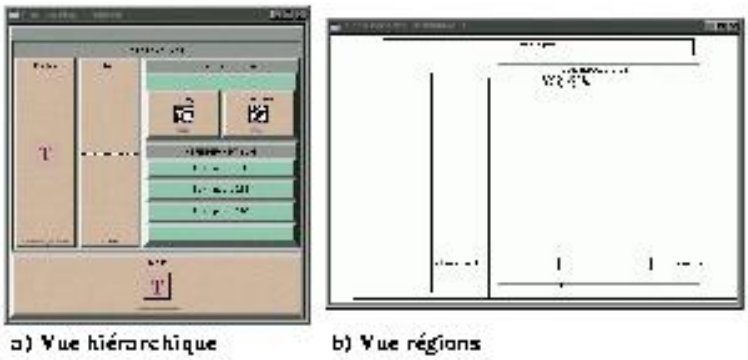


Figure II-20 : Grins : vue hiérarchique et vue des régions

L'édition au sein de Grins se fait essentiellement en 3 étapes :

- Spécification du placement temporel et spatial dans les vues adaptées.
- Spécification d'attributs à l'aide de formulaires.
- Visualisation de la dimension temporelle dans la vue temporelle et exécution du document. L'auteur peut de plus, positionner des arcs de synchronisation entre les objets.

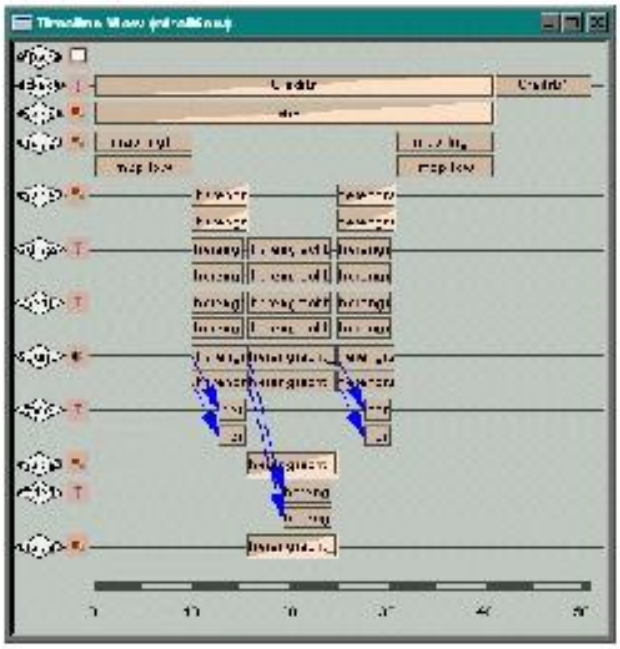


Figure II-21 : Grins, vue temporelle

Grins permet de spécifier un document de façon relative, avec la possibilité de définir des événements entre les objets (dans la limite de ce que permet SMIL 1.0).

Par exemple, l'édition temporelle qui est basée sur une structure d'arbre d'opérateurs se fait dans une vue hiérarchique, le placement temporel résultant sera ensuite visualisé dans une vue temporelle qui représente sur des axes horizontaux le placement des objets. L'auteur peut à ce moment raffiner sa spécification à l'aide d'arcs de synchronisation. Ils permettent de lier le début ou la fin d'un objet au début ou à la fin d'un autre objet (c'est équivalent au placement d'attribut *begin* et *end* relatif).

Du point de vue de l'auteur, on ne peut pas manipuler la vue temporelle pour ajuster directement le placement des objets en modifiant ainsi interactivement les arcs de synchronisation et la valeur des attributs *begin* et *end*. Notons que Grins n'offre pas de service d'aide au formatage, l'auteur doit donc spécifier explicitement les durées sur les objets.

On peut aussi remarquer que le formalisme d'édition de Grins est complètement lié à celui du langage SMIL, ce qui par exemple, pour la dimension spatiale, limite beaucoup les facilités d'édition de l'auteur qui est, dans

ce cas, obligé de placer les objets de manière absolue.

Les deux aspects importants de l'édition offert par Grins sont la visualisation de la structure d'opérateurs combinée avec une visualisation des informations temporelles dans la vue hiérarchique, ainsi que l'édition des attributs spatiaux par manipulation directe.

4.3.3 Madeus-97

Madeus, éditeur développé au sein du projet Opéra, permet d'éditer des documents multimédias spécifiés à l'aide de relations spatiales et temporelles entre les objets. La première version de Madeus (en 1997) a été réalisée par Nabil Layaïda et Loay Sabry au cours de leurs thèses [Layaïda97, Sabry99]. Depuis, ce prototype ne cesse d'évoluer et nous en présenterons une version récente qui inclue les travaux de cette thèse dans le chapitre VI.

Nous allons présenter dans cette section la version de Madeus au début de cette thèse. L'environnement auteur se compose essentiellement d'une vue présentation (Figure II-22 a) et de palettes (Figure II-22 b et Figure II-22 c), qui permettent à l'auteur de spécifier des relations temporelles ou spatiales entre les objets du document, après les avoir sélectionnés dans la vue de présentation. La vue de présentation est à la fois une vue dans laquelle l'auteur peut jouer son document et aussi une vue d'édition puisque l'auteur peut définir le placement des objets en déplaçant ceux-ci directement à l'écran.

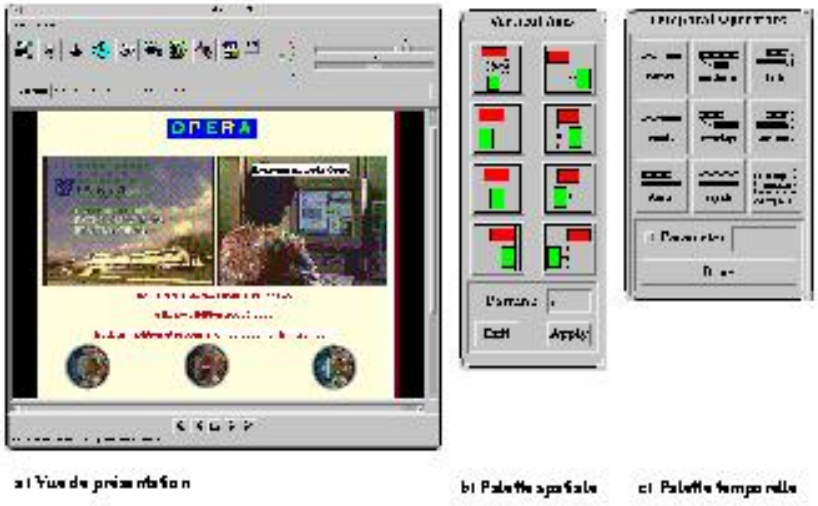


Figure II-22 : Les différentes vues de l'environnement Madeus-97

L'objectif était d'intégrer le WYSIWYG le plus possible tout au long du processus d'édition. L'environnement auteur de Madeus est une des premières réalisations d'un environnement d'édition /présentation intégré.

Dans la version Madeus-97, les principales lacunes du point de vue de l'édition sont :

- Absence de vue temporelle. La perception de l'enchaînement temporel se fait de manière locale par la vue de présentation.
- Absence d'un module de formatage temporel complet qui permettrait de calculer automatiquement le placement et la durée des objets temporels.
- Impossibilité d'exprimer des structures de contrôle avec de l'inderminisme.

Nous verrons que nous apporterons par la suite une réponse aux deux premiers points.

4.4 Synthèse sur les environnements d'édition

Dans cette section, nous avons pu voir qu'il existait aujourd'hui de nombreux outils auteur de documents multimédias. Cependant ces outils ne permettent pas de fournir un bon compromis entre : pouvoir d'expression et simplicité d'édition, même si un outil comme Grins offre des services d'édition avec des services de visualisation d'informations temporelles qui semblent intéressants et prometteurs. Parmi les

systèmes d'édition, l'auteur a aujourd'hui le choix entre :

- Des outils à base de modèles prédéfinis qui sont relativement simples d'utilisation et qui permettent à l'auteur d'écrire des documents proches de modèles, ou d'exemples prédéfinis. Cependant, ces outils autorisent difficilement un auteur à faire ce qu'il désire indépendamment des exemples proposés.
- Des outils de plus haut niveau, basés sur un formalisme proche du langage de sauvegarde, comme Grins et Madeus. Ces outils doivent cependant améliorer les aides qu'ils fournissent à l'auteur pour l'édition.
- Des outils mélangeant plusieurs formalismes de spécification en fonction des besoins de l'auteur. C'est le cas par exemple, de Director qui offre un formalisme absolu et un formalisme de programmation pour répondre aux besoins de l'auteur. Cette approche semble la plus prometteuse, mais un choix judicieux sur les formalismes proposés doit être fait.

5 Conclusion

5.1 Bilan des approches de spécification et d'édition

On s'aperçoit aujourd'hui qu'il existe un lien très fort entre le formalisme d'édition et les facilités offertes à l'auteur. L'auteur a le choix entre :

- Spécifier son document de manière absolue à l'aide d'une interface de type vue temporelle. L'expressivité de ce genre d'approche est relativement faible. L'auteur ne peut que placer dans le temps les objets et ne peut pas définir de relations entre eux. Par contre, les interfaces fournies sont relativement simples d'utilisation (SmilWizard, RealSlideShow, Powerpoint).
- Spécifier son document à l'aide d'un langage de programmation. Ce choix offre une très grande expressivité à l'auteur, par contre l'interface revient à des environnements de programmation classique, qui sont difficilement exploitables pour des non-informaticiens (C++/ Java).
- Spécifier son document à base d'approches relationnelles. Dans ces approches, l'objectif est d'offrir le meilleur compromis entre l'expressivité et la simplicité d'édition puisque l'auteur définit des relations entre les objets. Le système d'édition doit en contrepartie fournir des mécanismes de visualisation plus complexes pour aider l'auteur à manipuler les relations et à comprendre et maîtriser l'espace de solutions.

De l'étude des sections précédentes, on peut en conclure le récapitulatif de la Figure II-23. La partie édition des formalismes a été obtenue en se basant sur un environnement (fictif ou réel), manipulant directement les concepts de l'approche. Par exemple une édition basée sur une vue hiérarchique, dans le cas d'arbre d'opérateurs, ou une édition via une vue temporelle pour les approches absolues,

De cette figure, on peut noter les éléments suivants :

- *Formalismes* :
  - absolu : ce formalisme offre une édition pauvre mais élémentaire.
  - programmation : ce formalisme très riche n'offre que peu de possibilités d'aide dans le cadre de l'édition. La vie du document est liée au coût d'écriture.
  - événementiel : il offre une forte expressivité mais la spécification rend difficile la maintenance du document.
  - relationnel : il offre une forte expressivité mais nécessite une aide à la visualisation et au formatage de la part du système auteur.
- *Outils*:
  - Director : il utilise les formalismes absolu et de programmation. De ce fait il cumule les avantages et les inconvénients de ces deux approches.
  - SmilWizard : simple d'utilisation, cependant l'auteur n'a aucun contrôle sur ce qu'il produit.

- Grins : basé sur un arbre d'opérateurs temporels (celui de SMIL), cet outil est très dépendant du modèle sous-jacent. L'auteur devra, lors de chaque modification de la présentation de son document, changer sa structure de document du fait que celle-ci est liée fortement à la présentation.
- Madeus-97 : basé sur une approche relationnelle, cet outil était une première étape dans la réalisation d'un environnement auteur complet. Il a de ce fait des lacunes dans l'aide à la visualisation de l'auteur et l'automatisation de certaines tâches.

Les parties grisées sont les parties qui nous semblent les plus significatives.

Besoins	**Approches**							
	Formalismes				**Outils**			
	Absolu	Progr.	Evén.	Relation	Director	Smil Wizard	Grins	Madeus-97
Expressivité								
Médias	+	++	++	++	++	+	+	+
Interactions	-	++	++	+	++	-	+	+
Simplicité								
Instructions	-	++	++	+	++	-	+	-
Structure	-	+	+	++	-	-	+	++
Adaptation	-	++	++	++	-	-	+	++
Modification	-	-	-	++	-	-	-	+
Edition								
Visualisation	++	-	-	+	+	+	+	+
Fonction d'édition	-	-	+	++	-	+	+	+
Présentation	+	+	+	++	+	+	+	+
Cohérence	++	-	-	++	++	++	+	++
Cycle d'édition	-	-	-	++	-	-	+	++
Vie du document	-	-	+	++	-	-	-	-

Figure II-23 : Récapitulatif de la satisfaction des différents besoins

5.2 Choix d'un formalisme d'édition

Aujourd'hui, tous les formalismes de spécification de documents multimédias permettent de définir des langages avec à peu près le même pouvoir d'expression si l'on considère que tous les objets sont déterministes.

On peut en effet, spécifier tous les documents sous une forme absolue avec des liens. Pour cela, on explicite toutes les exécutions possibles, exécutions qu'on relie par des liens. On peut noter que de tels documents contiennent plusieurs présentations possibles suivant les liens parcourus. De ce fait le critère de pouvoir d'expression est peu discriminant dans la réalisation d'un environnement auteur si l'on considère des

documents indéterministes avec navigation. L'aspect important est donc la facilité de description du comportement temporel et spatial.

Au cours de cette thèse, nous nous intéresserons essentiellement aux documents déterministes avec de la navigation (section 2.1.1). Au cours des chapitres suivants nous étudierons l'édition et les services que peut fournir un système d'édition pour aider l'auteur dans ce contexte. Nous ferons à la fin de la thèse des propositions pour généraliser les travaux réalisés aux documents contenant de l'indéterminisme.

5.3 Bilan des besoins non satisfaits

Parmi les besoins peu satisfaits à l'heure actuelle, ceux qui nous semblent les plus importants dans un processus d'édition complet, sont la facilité de modification du document et la visualisation des informations contenues dans le document. Les approches relationnelles semblent offrir aujourd'hui les meilleures capacités pour répondre à ces besoins, de par la flexibilité intrinsèque de ces approches. Cependant les outils actuels n'offrent pas de facilité d'édition pour exploiter cette souplesse. Par exemple, aucun système auteur n'offre une édition par manipulation directe avec maintien des relations.

De plus, les outils actuels sont souvent des éditeurs de langage sans être réellement des outils auteur, car ils n'offrent que peu d'aide à l'auteur et qu'ils n'automatisent que peu de tâches (formatage, détection des incohérences).