Table des matières

Support d'exécution de documents multimédia

Nabil Layaïda, Cécile Roisin [a] and Loay Sabry-Ismail [b] [1]

Résumé : Ce chapitre décrit les fonctions nécessaires à l'exécution de documents multimédia. Dans une première partie, les principaux modèles utilisés pour spécifier des présentations multimédia sont analysés. Puis, après avoir identifié les caractéristiques des présentations multimédia, nous décrivons les principes de conception d'une machine d'exécution multimédia au travers de son architecture générale. Nous détaillons ensuite les composants qui réalisent les fonctions d'ordonnancement et de contrôle de l'exécution. Nous illustrons ces différents composants au travers de leur réalisation dans le système prototype Madeus.

Mots-clés : Multimédia, Support de présentation, Synchronisation, Navigation.

1. Introduction

Le qualificatif de *multimédia* se rencontre dès lors qu'une application traite des informations non seulement textuelles mais de nature variée comme du son, des images, des vidéos. Dans le cas de l'échange d'information entre utilisateurs, l'unité d'échange est appelée *document* et les applications de traitement de documents multimédia doivent assurer la restitution des informations qui ne sont plus seulement textuelles. On trouve ainsi des documents caractérisés par une dynamique beaucoup plus importante grâce à l'intégration d'informations textuelles, d'images mais aussi du son, de la vidéo, des images animées, des éléments contrôlés par des programmes externes, etc. C'est bien évidemment pour cette dernière catégorie de documents que les problèmes de support à leur présentation se posent avec le plus d'acuité. En effet, les données comme

[1] (a) Projet Opéra - Inria Rhône-Alpes, (b) Université Ain Shams - Le Caire

l'audio et la vidéo imposent des contraintes de synchronisation qui sont difficiles à assurer dans un environnement comme l'internet où les ressources disponibles sont très variables.

Ainsi, les documents *multimédia* sont non seulement caractérisés par des contenus de nature diverse mais aussi par l'organisation temporelle de leurs composants. Dans ce chapitre, les unités d'information considérées comme atomiques sont appelées *objets média* et la description de l'enchaînement des objets dans le temps est appelée *scénario temporel.* Cette organisation temporelle est à la base des besoins que doivent assurer le support de présentation. Nous verrons qu'une prise en compte intelligente de cette information lors de l'exécution du document peut aider à améliorer la qualité de la présentation Aussi nous consacrons la première partie de ce chapitre à l'étude des modes d'expression des scénarios temporels, la seconde étant présente les différentes fonctions nécessaires à la réalisation d'un support d'exécution de documents multimédia.

2. Expression du temps dans les documents multimédia

2.1. *Besoins d'expression des informations temporelles*

L'introduction de la dimension temporelle dans les documents conduit nécessairement à de nouveaux besoins d'expression : la durée des objets, leur placement temporel et leur synchronisation. Ces informations temporelles doivent s'intégrer dans l'ensemble des informations attachées aux documents. Par exemple, pour spécifier le déplacement d'un objet sur l'écran, il est nécessaire d'exprimer une information spatiale pour la trajectoire, comme les positions initiale et finale correspondant au déplacement, ainsi qu'une information temporelle comme la durée du déplacement et sa synchronisation avec d'autres objets.

La définition des structures temporelles d'un document consiste à spécifier les schémas de synchronisation entre les objets et les éléments composites du document.

2.2. *Typologie des approches de spécification*

Différents modes de composition temporelle ont été expérimentés, depuis la spécification par placement absolu ou *timeline* (spécification des dates de début et de fin des objets), l'utilisation de langages de programmation ou d'opérateurs de composition jusqu'à l'utilisation d'algèbres de relations. Ces différents modes de composition peuvent être classés selon trois approches :

1. Les techniques opérationnelles ou impératives : axe de temps absolu (Director [MAC 99], HyTime [ISO 97]), programmation par scripts (Lingo dans Director, MHEG [MEY 95]).
2. Les techniques déclaratives par opérateurs temporels qui correspondent à une spécification d'une structure d'exécution à base d'arbres (CMIFed [ROS 93] et SMIL [SMI 98]).
3. Les techniques déclaratives par relations temporelles, Isis [KIM 95], Madeus [JOU 98], dans lesquelles l'auteur déclare les placements temporels souhaités sans donner toutes les informations temporelles attachées aux objets. La plupart de ces techniques s'appuient sur l'algèbre d'Allen [ALL 83] pour la spécification des contraintes temporelles.

Nous décrivons brièvement dans la section suivante le mode de spécification de la première approche au travers du format standard MHEG. La section 2.4 est consacrée au standard W3C Smil qui fait partie de la deuxième classe d'approches. Nous présentons finalement les principes des approches de spécification par contraintes en section 2.5.

2.3. *Étude du format MHEG*

Le but du groupe MHEG [MEY 95] (*Multimedia and Hypermedia Information Coding Experts Group*) est la définition et la standardisation d'un format portable de représentation multimédia pour faciliter l'échange d'objets entre sites. Ceci a abouti à un modèle objet constitué d'une hiérarchie de classes *MHEG object* comprenant les classes *Content* (qui permettent de décrire le contenu d'objets média), *Script* (pour définir des comportements sous forme de programmes), *Link* et *Composite*, ces deux derniers étant décrits ci-dessous.

Les *MHEG links* ne sont pas les liens hypermédia (hyperliens) conventionnels mais permettent de spécifier les relations spatiales, temporelles et conditionnelles entre les objets média ainsi que les actions effectuées par ou sur ces objets. Un *MHEG link* consiste en une condition (*LinkCondition*) et un effet (*LinkEffect*) qui est une liste d'actions à exécuter lorsque la *LinkCondition* est vérifiée. Les *MHEG links* avec les deux primitives temporelles binaires : *serial* (qui définit une exécution en séquence entre des objets média) et *parallel* (indiquant une exécution simultanée des objets média) sont utilisés pour spécifier un scénario multimédia. En ce sens, le modèle d'exécution de MHEG est fondé sur la programmation par événements.

Les *objets composites* sont utilisés comme un container pour un groupe de *MHEG objects* formant ainsi un objet complexe. Ils permettent aussi la synchronisation entre des parties d'une présentation multimédia à l'aide de primitives de composition spatiale et temporelle.

```
(Composite:InfoScene
<autres attributs ici...>
group-items:
(bitmap: BgndInfo
content-hook: #bitmapHook
original-box-size: (320 240)
original-position: (0 0)
content-data: referenced-content: "InfoBngd"
)
(text:
content-hook: #textHook
original-box-size: (280 20)
original-position: (40 50)
content-data: included-content: "1. STAR ..."
)
links:
(link: Link1
event-source: InfoScene
event-type: #UserInput
link-effect: action: transition-to: InfoStar
)
<suite de la spécification ... >
)
```

Figure 1. *Exemple de description d'une scène en MHEG*

MHEG est à l'origine de quelques prototypes d'application multimédia comme les projets GLUE, GLASS [GEY 97] et MAJA [BIT 96] de DeTeBerkom. Ainsi, le projet MAJA consiste en une *Java Applet* dont le rôle est de présenter des applications MHEG en reliant, au moyen de la technologie de communication CORBA, un client de présentation MHEG à un moteur MHEG sur un site serveur. Le principe de fonctionnement est le suivant. GLASS a pour objectif de réaliser un système qui peut offrir différents services comme la vidéo à la demande ou bien la télévision interactive. La présentation au sein de ce système est fondée sur la norme MHEG, comme par exemple la spécification de la scène Fig. 1 : la scène se compose d'une image du fond avec plusieurs objets de type texte formant les options d'un menu. À chaque texte, est associé un objet *MHEG Link* qui décrit son comportement vis à vis de l'interaction d'utilisateur.

Comme pour les langages de scripts classiques, les principes de spécification de MHEG lui permettent d'offrir un fort pouvoir d'expression (variété des objets de base traités, composition temporelle, navigation, etc.). Cependant, leur point faible vient de leurs capacités limitées à répondre aux besoins

des auteurs : les compétences en programmation requises pour les utiliser restreignent leur emploi aux seuls informaticiens ; il est de plus très difficile de percevoir l'enchaînement temporel des objets par la simple lecture d'un script car la composition temporelle est dispersée dans le script ; enfin, comme pour tout langage de programmation, la mise à jour d'un scénario est une tâche souvent délicate du fait des erreurs qu'il est facile d'y introduire. Les besoins pour les auteurs sont cependant un peu mieux couverts : les possibilités de structuration et de modularité sont fournies par l'approche objet et la notion d'objets composites.

Notons enfin que malgré les efforts importants effectués pour établir les standards de ce type (HyTime, MHEG ainsi que PREMO), les réalisations qui s'y réfèrent sont peu nombreuses. Deux raisons principales à cela : d'une part leur état non encore stabilisé (sauf pour HyTime), et d'autre part leur complexité surtout dans le cas de HyTime et PREMO. Néanmoins, le standard MHEG-5 a été adopté dans les spécifications DAVIC (*Digital Audio VIsual Council*) comme format pour les services de la télévision numérique.

2.4. *Étude du langage Smil*

SMIL (*Synchronized Multimedia Integration Language*) [SMI 98] est un langage déclaratif développé par le W3C dont le but est de spécifier des documents multimédia sur le Web et de les jouer par des navigateurs SMIL. Un tel navigateur peut être soit un système de présentation isolé qui peut être configuré suivant les besoins d'une communauté d'utilisateurs, soit intégré dans un navigateur Web existant.

SMIL est un format d'intégration, c'est-à-dire qu'il ne décrit pas le contenu des objets média faisant partie d'une présentation multimédia, mais plutôt leur composition temporelle et spatiale ainsi que les hyperliens qui lient ces objets. En effet, un auteur d'un document SMIL peut :

- Décrire le comportement temporel de la présentation. SMIL utilise les opérateurs *seq* et *par* pour spécifier qu'un ensemble d'objets est joué respectivement en séquence et en parallèle. La durée d'un objet peut être spécifiée explicitement. De plus, la date de début ou de fin d'un objet peut être spécifiée par un délai par rapport à la date de début ou de fin d'un autre objet. Enfin, dans le cas de la mise en parallèle, des attributs de synchronisation sont définis pour exprimer la terminaison de la composition parallèle. Elle peut par exemple être définie par la terminaison du plus court objet.
- Décrire le placement des objets média sur l'écran pendant la présentation. Sur la fenêtre principale de présentation, SMIL spécifie des régions

```
<smil>
<head>
  <layout type="text/smil-basic">
    <region id="title"
          left="4" top="4" width="47" height="22"/>
    <region id="image"
          ...
  </layout>
</head>
<body>
   <par id="A" endsync="last">
    <seq id="Seq_P">
         <audio id="P" dur="20.0 s"
           src="http://www.inria.fr/past.au"/>
         <text id="Name" region="title" dur="5.0 s"
           src="http://www.inria.fr/text.html"/>
         <img id="Hello" region="image"
           src="http://www.inria.fr/hello.gif" dur="10.0 s" />
       </seq>
       <video id="Visit" region="video"
           begin="40s"
           src="http://www.inria.fr/visit.mpg"/>
       <a id = "H1" href="#Next" show="replace">
          <img id="Button" region="xx"
           src="http://www.inria.fr/button.gif" />
       </a>
    </seq>
    <par id="Next"> <!-- Next part of the scenario -->
     .....
   </par>
</body>
</smil>
```

Figure 2. *Un exemple de scénario spécifié dans le langage SMIL*

dont la position et la taille sont exprimées soit en valeur absolue, soit en pourcentage de la taille de la fenêtre principale,

– Associer des hyperliens aux objets média. La désignation de la destination d'un lien est effectuée en termes d'adresse URI.

L'exemple de la Fig. 2 permet d'illustrer les différentes possibilités de spécification de SMIL.

Les apports principaux de SMIL se situent au niveau de la portabilité du langage de spécification des documents multimédia et de la spécification de la navigation. En effet, dans SMIL, les ancres des liens hypermédia peuvent être définies activables pendant des intervalles de temps et/ou sur des sous-régions de la fenêtre où l'objet média est affiché.

Une nouvelle version de cette norme, appelée SMIL Boston [SMI 99], est en cours de définition par le groupe SYMM du W3C. Sa principale caractéristique est de suivre une approche modulaire qui permet de mieux intégrer les autres standards du W3C : par exemple, l'utilisation du standard de dessin vectoriel (Scalable Vector Graphics) avec le module de synchronisation de SMIL permet d'obtenir des animations graphiques. De même l'utilisation de XPointer et XLL permet d'enrichir le modèle de liens temporels. Par ailleurs, le modèle temporel de SMIL est étendu, notamment sur les possibilités d'interaction et la synchronisation événementielle.

2.5. *Langages à base de contraintes*

Différents travaux théoriques, qui proviennent notamment du domaine de la planification en robotique dans lequel le problème de la synchronisation temporelle est crucial, ont permis de formaliser l'information temporelle des objets ainsi que des relations entre eux. Ainsi, tout objet multimédia peut être manipulé à travers trois informations temporelles principales (dont l'une est redondante) :

- Son instant de début.
- Sa durée de présentation.
- Son instant de fin.

Il existe ainsi deux façons de représenter le déroulement d'un scénario : à travers les changements qui surviennent, comme *Lorsque la vidéo est terminée, le texte commence à s'afficher et la bande son démarre*, ou au contraire en reliant globalement les activités entre elles comme *le texte et la bande son démarrent en même temps et suivent la vidéo.* Ceci débouche sur deux types de représentation :

- Une représentation fondée sur les *instants*, en particulier les instants de début et de fin des objets. Trois relations sont possibles : avant, après ou pendant.
- Une représentation fondée sur les *intervalles* correspondant à la durée des objets. Allen a étudié les relations possibles entre deux intervalles et a

```
<composite>
    <media type="texte" name="TxtMessage" duration"3 5 7"
                        source="http://info.com/hello.html" />
    <media type="video" name="V" duration"10 15 20"
                        source="http://film.com/hello.mpg" />
    <media type="audio" name="Comment" duration"10 15 20"
                        source="http://speech.com/hello.au" />
    <relations>
       <before Interval1="TxtMessage" Interval2="V" />
       <equal Interval1="V" Interval2="Comment" />
    </relations>
</composite>
```

Figure 3. *Exemple de description par contraintes*

montré qu'elles se réduisaient à 13 relations, soit de nature parallèle (pendant, au_démarrage, à_la_fin, en_même_temps, en_chevauchement), soit de nature séquentielle (avant, à_la_suite).

Dans le cas des documents multimédia, des modèles de composition temporelle fondés sur les intervalles ont été proposés (Isis [KIM 95], Tiempo [WIR 97] et Madeus [JOU 98]). Le jeu de relations d'Allen a été étendu, notamment pour exprimer la *causalité*, comme lorsque l'occurrence d'un instant de fin d'un élément provoque l'arrêt d'un autre, même si ce dernier n'a pas restitué tout son contenu à l'utilisateur [DUD 95]. On parle alors de *relations d'interruption.*

Ainsi, avec cette approche, l'auteur spécifie ce qu'il souhaite obtenir comme placement temporel (par exemple, qu'une vidéo soit présentée après un texte et en même temps qu'un commentaire sonore, cf. Fig. 3) sans avoir à spécifier toutes les informations temporelles attachées à ces objets (instants de début et de fin, durée). Un scénario est donc défini par un ensemble de *contraintes* comprenant des *relations temporelles* (avant, pendant, ...) entre les objets et une spécification de la *durée souhaitée de chaque objet* sous forme d'un intervalle de valeurs possibles.

À partir de cette liste de contraintes, le système peut vérifier la cohérence du document : l'existence d'au moins une solution (pour chaque objet, une date de début et une durée) qui respecte toutes les contraintes ; il peut ensuite produire statiquement et/ou dynamiquement une (des) solution(s) de placement temporel à partir des spécifications : par analogie aux systèmes d'édition conventionnels, on parle de *formatage temporel.* Le document formaté peut alors être présenté à l'auteur/ lecteur. Ainsi, un système de présentation peut tirer parti de la spécification par contraintes pour mettre en œuvre les fonctions de présentation adaptables au contexte de présentation.

2.6. *Synthèse : de la spécification à la présentation*

La présentation d'un document multimédia consiste à activer (afficher / formater / ...) les objets média en respectant les spécifications temporelles, spatiales, d'activation, etc. fournies par l'auteur. Pour cela, les spécifications sont le plus souvent transformées dans une *représentation intermédiaire* (interne) : arbre ou graphe temporel, code impératif (par exemple en code Java). Cette représentation intermédiaire est à la base des capacités de la machine de présentation (réactivité, prédiction, adaptation, etc.). En effet, selon la représentation utilisée, trois approches de gestion des événements d'une présentation multimédia peuvent être identifiées : réactive, prédictive ou hybride.

2.6.1. *L'approche réactive*

La machine de présentation utilise une représentation du scénario qui a la forme d'une liste de *condition(s)-action(s)*. La *condition* est une expression booléenne fonction des attributs d'objets média et des événements générés pendant la présentation, et l'*action* constitue l'action de présentation à effectuer si la condition correspondante est validée. Pendant la présentation, l'ordonnanceur fonctionne comme suit : il reçoit des événements générés par les objets média et l'interaction de l'utilisateur, et chaque fois qu'une condition est validée, il lance des *commandes* pour exécuter l'action correspondante. Il faut noter que l'ordonnanceur, dans cette approche, n'a pas de vue globale du scénario. Cette approche est adoptée par les systèmes réalisant le standard MHEG. Ses principaux avantages sont d'une part sa simplicité de mise en œuvre et d'autre part son adéquation aux situations intrinsèquement réactives.

2.6.2. *L'approche prédictive*

Dans le cas de l'*approche prédictive*, l'ordonnanceur peut savoir comment la présentation d'un scénario doit se comporter dans le futur, et c'est l'ordonnanceur qui génère les événements. En effet, l'ordonnanceur a une vue globale du scénario grâce à l'utilisation d'une structure interne plus riche : graphe temporel [JOU 98] ou encore liste d'événements triés selon leurs estampilles temporelles [FLI 95].

- Structure linéaire temporelle : liste d'événements triés selon leurs estampilles temporelles auxquels sont attachés les actions de présentation à effectuer.
- Arbre de relation temporelle : arbre dont les nœuds portent les relations alors que les feuilles contiennent les objets média.

– Graphe des relations temporelles : graphe dont les nœuds constituent les instants temporels alors que les arcs constituent les objets média, comme dans le réseau de Pétri d'OCPN [LIT 93], le graphe orienté acyclique d'Isis ou l'hypergraphe de Madeus.

L'avantage de cette approche est la possibilité d'effectuer les actions préliminaires en avance afin par exemple de diminuer le délai et la gigue pendant la présentation multimédia. Par contre, une approche prédictive pure ne peut pas réagir aux événements externes susceptibles de modifier le scénario temporel et donc à l'interactivité des présentations multimédia.

Un exemple très répandu de système de présentation prédictif est l'outil RealPlayer G2 de Real Networks [REA 99] puisque l'exécution s'appuie sur le calcul à l'avance du scénario.

2.6.3. *L'approche hybride*

Afin de diminuer le délai et de réaliser l'interactivité dans les applications multimédia, l'ordonnanceur doit suivre une approche hybride qui intègre les deux approches : réactive et prédictive. C'est cette approche que nous présentons dans la section ci-dessous.

3. Support de présentation

Après avoir brièvement décrit les caractéristiques des présentations multimédia, nous décrivons les principes de conception d'une machine d'exécution multimédia au travers de son architecture générale. Nous détaillons ensuite les composants qui réalisent les fonctions d'ordonnancement et de contrôle de l'exécution.

3.1. *Caractéristiques et besoins de présentation multimédia*

Nous avons identifié cinq caractéristiques dont il est nécessaire de tenir compte lors de la réalisation d'une machine de présentation multimédia. L'importance des différents aspects est plus ou moins grande suivant le type de présentation considéré.

3.1.1. *Hétérogénéité des objets média*

Les objets média qui font partie d'une présentation multimédia ont des caractéristiques hétérogènes comme leur comportement temporel, leur format de codage, leur volume et leur mode de perception par l'utilisateur.

3.1.2. *Organisation des objets média entre eux*

Comme on l'a vu dans la section précédente, une présentation multimédia n'est pas une simple présentation d'objets média indépendants. C'est un processus qui traite un ensemble d'objets organisés logiquement, spatialement et temporellement, et qui doit donc assurer que les spécifications de composition (en particulier temporelle) sont respectées lors de la présentation quelles que soient les conditions extérieures.

3.1.3. *Portabilité des applications multimédia*

C'est une caractéristique qui touche non seulement à la possibilité de jouer des présentations multimédia sur plusieurs plates-formes (code exécutable, cartes périphériques pour les média), mais aussi à la qualité de leur exécution : en effet, le comportement temporel d'une présentation est très sensible à des paramètres comme la vitesse du processeur, la charge de la machine ou encore les accès au réseau et aux périphériques.

3.1.4. *Dynamicité de la présentation multimédia*

Le *processus de présentation d'un document multimédia* est un processus dynamique où, à chaque instant de la présentation, les objets média changent leur état de présentation selon le scénario spécifié et les interactions effectuées par l'utilisateur. La façon selon laquelle les changements d'état doivent avoir lieu est définie par le mode de synchronisation de la présentation multimédia : synchronisation intra-objet, synchronisation inter-objet et synchronisation avec l'environnement.

3.1.5. *Répartition des objets média*

Les objets média faisant partie d'une même application multimédia peuvent être distribués sur différents serveurs : soit sur le même site, soit sur des sites

différents. Du fait de sa grande taille et des aléas du débit des liaisons numériques, l'accès à un objet peut subir un délai plus ou moins important et d'une valeur aléatoire. Par conséquent, la qualité de la présentation de ces objets peut être dégradée à cause du non respect des échéances temporelles.

3.2. *Architecture générale*

Dans cette section, nous présentons l'architecture générale et les fonctions de la machine d'exécution multimédia. À partir des informations du scénario maintenues dans la représentation intermédiaire (cf. section 2.6), la machine d'exécution réalise les fonctions de présentation en tenant compte de l'interaction utilisateur : elle gère la synchronisation entre les objets média, la navigation temporelle intra- et inter-document ainsi que l'indéterminisme de la présentation.

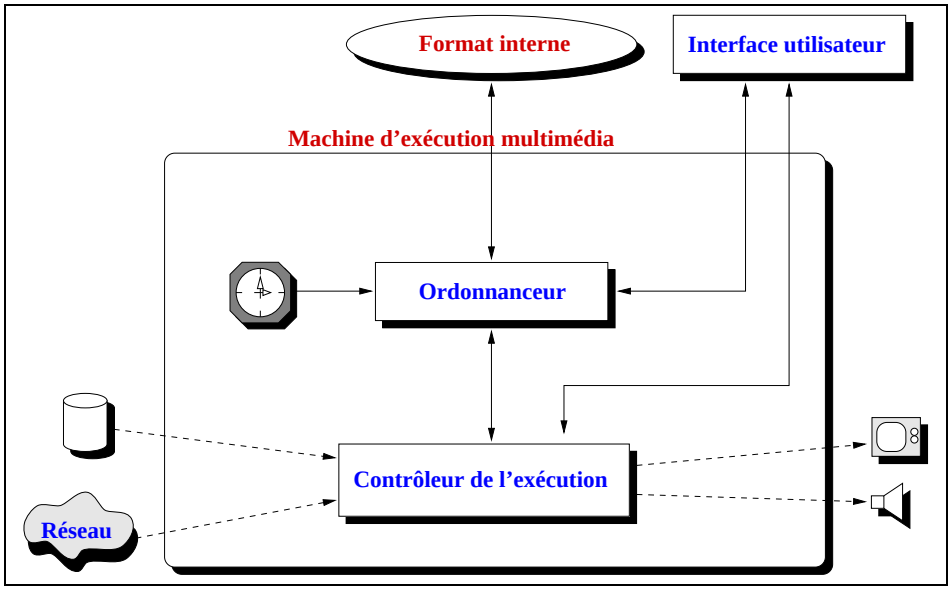

Figure 4. *Architecture générale d'une machine de présentation multimédia*

Sur la Fig. 4, les fonctions à réaliser sont divisées en deux groupes : le premier constitue l'ensemble des fonctions de haut-niveau qui réalisent l'ordonnancement des actions de présentation tandis que le second contrôle et gère les accès aux ressources physiques. Plus précisément, on a :

- Les fonctions d'ordonnancement : synchronisation à gros grain (inter objets), synchronisation fine (inter et intra objets), gestion de la navigation temporelle, gestion de la présentation des objets média.
- Les fonctions de contrôle de l'exécution (en général dépendant de la plate-forme d'exécution) : gestion des accès et des périphériques.

3.3. *Ordonnancement d'une présentation multimédia*

La présentation d'un document multimédia est une séquence d'actions d'exécution appliquées aux dates correspondant à la spécification temporelle donnée par le scénario et qui s'applique sur l'ensemble des objets impliqués dans le document. Ces actions sont les suivantes :

- afficher/cacher un objet visible sur l'écran ou commencer/arrêter la sortie d'un objet audible sur un canal sonore,
- jouer une entité d'un flot composant un objet continu comme par exemple une image d'une séquence vidéo ou un échantillon d'audio. Cette action nécessite des opérations supplémentaires pour accéder à cette entité et, en général, pour la décompresser,
- jouer un effet de style tel que déplacer un objet sur l'écran, changer le niveau de volume d'une audio ou changer périodiquement la couleur d'un texte.

C'est l'ordonnanceur qui décide des actions de présentation à effectuer, en utilisant pour cela un ensemble d'informations internes et externes fournies par différentes sources :

- la représentation intermédiaire (format interne) qui donne l'ordonnancement de la présentation d'objets selon le scénario spécifié,
- l'état d'exécution des objets média (actif, affiché, suspendu, terminé, etc.) que contrôle le gestionnaire d'objets,
- l'horloge système qui sert à réaliser effectivement la durée spécifiée pour différents types d'objets comme :
  - les délais temporels associés à certaines relations temporelles ou spécifiés explicitement par l'auteur ;
  - les objets discrets temporisés, par exemple une image qui s'affiche pendant 5 secondes ;
  - les effets de style de présentation, par exemple le déplacement d'un objet sur l'écran avec une vitesse spécifiée.

– l'interface utilisateur qui spécifie la commande de présentation choisie par l'utilisateur.

L'organisation de l'ordonnanceur est décrite dans la Fig. 5. Celui-ci se compose d'un gestionnaire d'événements et de quatre autres gestionnaires : le gestionnaire de synchronisation, le gestionnaire des objets média, le gestionnaire de la navigation temporelle et le gestionnaire de l'indéterminisme. Le gestionnaire d'événements de l'ordonnanceur filtre les événements engendrés par la présentation et les notifie au(x) gestionnaire(s) concerné(s).

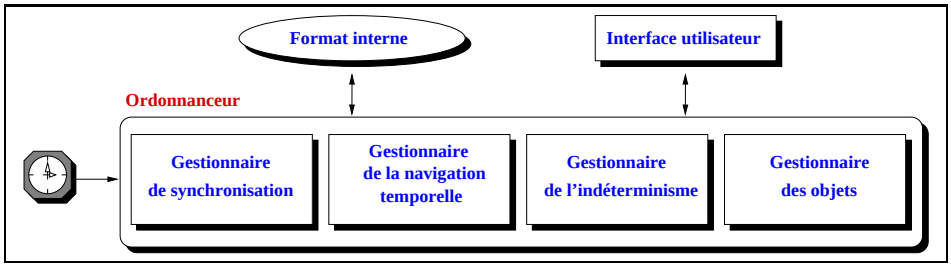

Figure 5. *Structure de l'ordonnanceur*

Ces différents gestionnaires sont décrits ci-dessous. Nous présentons également leur mise en œuvre dans le prototype Madeus.

3.3.1. *Gestionnaire de synchronisation*

Deux types de synchronisation sont à prendre en compte lors d'une présentation multimédia : la *synchronisation à gros grain* qui définit la synchronisation entre les instants de début et/ou de fin des objets telle que définie dans le scénario, et la *synchronisation fine* qui est définie entre les unités composant les objets média synchronisés, intra-objet ou inter-objet, comme par exemple le *lip-sync*.

– **La synchronisation à gros grain** : cette synchronisation s'appuie principalement sur les informations fournies par la représentation intermédiaire (condition -> actions, arbre ou graphe temporel) pour, par exemple, lancer la présentation d'un ensemble d'objets simultanément, terminer la présentation d'un objet dépendant de la terminaison d'un autre, notifier périodiquement les objets liés par une synchronisation fine, etc. Avec une structure d'hypergraphe, le gestionnaire de synchronisation peut avoir une approche plus prédictive puisqu'une telle structure fournit une vue globale du scénario qui permet de contrôler la synchronisation inter-objet de façon à garantir une qualité de service de la présentation

multimédia. Ainsi lorsque le gestionnaire de synchronisation ne peut pas garantir la coïncidence temporelle entre les instants de début d'un ensemble d'objets média à cause de l'insuffisance des ressources système et/ou des ressources réseau, un préchargement des objets média est effectué. Le mode de préchargement dépend du type d'objet : discret ou continu. En effet, un objet média discret est préchargé par une seule requête d'accès, tandis que, pour un objet continu, le préchargement est effectué au niveau des composants de l'objet (comme un groupe d'images d'une vidéo ou un ensemble d'échantillons d'une audio). La technique de préchargement, par conséquent, exige l'allocation de tampons pour stocker les données média préchargées. Ainsi, un problème important, lié à la disponibilité de l'espace mémoire, se pose dans le cas où plusieurs objets média doivent être préchargés simultanément. Il est nécessaire de faire un compromis entre l'avantage de la technique de préchargement et la disponibilité de l'espace mémoire.

- **La synchronisation fine** : la synchronisation fine est définie au niveau d'un seul objet ou entre deux objets qui sont fortement liés, comme par exemple un film qui se compose d'une vidéo et d'une audio.

  Dans le premier cas, il s'agit de minimiser la gigue (*jittering*), c'est-à-dire la *différence entre le temps de présentation réel et le temps de présentation nominal pour chaque unité d'un objet continu.* Pour cela des techniques de préchargement et de saut d'échantillons peuvent être mis en œuvre [KOU 96].

  La synchronisation inter-objets, comme le *lip-sync*, a pour objet de contrôler la dérive (*skewing*) entre des flux continus. Ce type de synchronisation peut être géré soit de façon centralisée, soit de façon distribuée au niveau des objets synchronisés. Dans le premier cas, le gestionnaire s'occupe de signaler aux objets concernés de façon périodique qu'ils doivent s'aligner temporellement. Par contre, dans le deuxième cas, le gestionnaire de synchronisation désigne un objet comme *maître* : c'est lui qui s'occupe de signaler périodiquement aux autres objets, les *esclaves*, qu'ils doivent s'aligner avec lui. Cette période de notification de resynchronisation dépend des seuils de dérive acceptable des différents objets [OWE 98]. L'objet audio est souvent choisi comme un maître parce qu'il ne tolère ni la perte de ses données, ni le changement de son taux de présentation, et exige que les autres objets s'adaptent à son taux effectif de présentation (voir SMIL [SMI 98]).

  Dans Madeus, nous avons choisi le mode de gestion distribuée pour la synchronisation fine afin de minimiser la charge du gestionnaire. De plus, la structure des objets média s'adapte parfaitement avec ce type de gestion, parce que les fonctions qui réalisent la présentation de chaque objet média sont maintenues par l'objet lui-même, grâce à l'approche orientée objet adoptée par le gestionnaire d'objets [SAB 98].

3.3.2. *Gestionnaire de la navigation temporelle*

Le gestionnaire de la navigation temporelle interprète les commandes de navigation lancées par l'utilisateur pour lui permettre de se déplacer temporellement à travers les documents multimédia. Cette forme de navigation est fournie au travers de boutons de contrôle du temps (TAC : Temporal Access Control) au niveau du système de présentation, comme les boutons de pause, reprise, accélération en avant ou en arrière, etc. Un tel gestionnaire est facile à mettre en œuvre pour le contrôle d'un seul média (une vidéo par exemple), mais se révèle plus complexe dans le cas d'informations multimédia structurées et fortement synchronisées. En effet les changements de l'unité temporelle (pour l'accélération) et les sauts doivent être effectués tout en respectant la structure temporelle du document.

Dans Madeus, un nouveau mode de parcours des documents multimédia à travers leur déroulement temporel [SAB 97] est proposé. En s'appuyant sur la structure logique (composites) et la structure temporelle (synchronisation des débuts et fins d'objets), la navigation est à la fois structurale et temporelle : il est possible de dérouler pas à pas un document en accédant directement aux instants significatifs de sa présentation (par exemple, le début de tous les composites d'un niveau donné de la hiérarchie ou le début/fin d'un objet à l'autre).

La réalisation de ce mode de navigation implique d'être capable de calculer l'état d'un document pour tout instant, c'est-à-dire la liste des objets actifs avec leur décalage temporel par rapport à leur instant de début : c'est ce qu'on appelle le *contexte de présentation*. En fonction des actions effectuées par l'utilisateur, les contextes sont calculés, empilés ou dépilés.

3.3.3. *Gestionnaire dynamique de l'indéterminisme*

Dans les sections précédentes, la synchronisation qui est présentée prend en compte uniquement l'aspect déterministe et donc prédictif des objets multimédia : les objets sont lancés avec des durées élaborées à l'avance lors du formatage. Or la prise en compte des retards et du comportement indéterministe des objets multimédia de façon générale est une nécessité absolue. En effet, les systèmes d'exploitation actuels, y compris les systèmes temps-réel, les protocoles réseau et la nature de certains média comme les programmes sont autant de sources d'indéterminisme qu'il faut prendre en compte. Ces comportements incontrôlables peuvent conduire à une désynchronisation des objets présentés et donc à un non respect du scénario spécifié par l'auteur.

Il est donc nécessaire d'effectuer un traitement pendant la présentation du

document multimédia. Ce traitement comporte deux phases : la phase d'observation et la phase de recouvrement. La phase d'observation permet de relever les valeurs effectives des intervalles incontrôlables, par exemple la durée d'un objet incontrôlable (comme un bouton d'interaction) est observée lors de sa terminaison. Ces valeurs sont ensuite utilisées pour ajuster le scénario dynamiquement de façon à respecter au mieux les contraintes temporelles décrites dans le graphe d'exécution. Cet ajustement se traduit par la re-synchronisation des objets en cours de présentation et des objets futurs qui se trouvent en retard ou en avance les uns par rapport aux autres. Pour cela, la durée de certains délais, celle de certains objets ou éventuellement leur vitesse nominale sont modifiées dans les limites prévues par le scénario (ces limites sont identifiées sous la forme de valeurs de *flexibilité* des objets).

Dans la mise en œuvre, l'algorithme de compensation et de réajustement dynamique est une sous-composante de l'ordonnanceur qui prend en charge ces événements particuliers : les événements de fin des intervalles incontrôlables. Dans la conception du modèle temporel, l'indéterminisme a été intégré comme composante fondamentale afin de décrire de façon plus réaliste le comportement temporel d'une présentation (les fluctuations de durée constituent un phénomène courant).

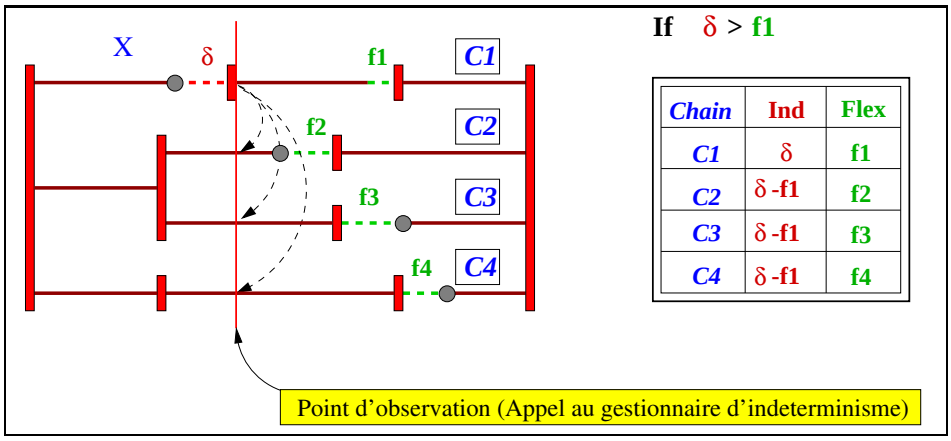

Figure 6. *Gestion prédictive de l'indéterminisme*

La Fig. 6 illustre l'application de cet algorithme pour l'observation et le recouvrement d'une désynchronisation produite par un objet incontrôlable. Lorsque l'événement de fin d'un objet incontrôlable a lieu, le délai imprévisible ou retard (noté δ) est en priorité pris en compte sur la chaîne correspondante (une chaîne représente un ensemble d'objets média et/ou de délais en séquence temporelle). Si la flexibilité (notée f1) de la chaîne depuis cet instant jusqu'à la fin de la chaîne est suffisante, le retard est reporté sur cette chaîne (ici C1) :

le reste de la chaîne est raccourcie de δ. Dans le cas contraire, le retard est transmis aux chaînes concurrentes (C2, C3 et C4) qui sont réajustées afin de permettre à la chaîne C1 de rattraper ce délai : elles sont rallongées de δ. L'algorithme est décrit de façon plus détaillée dans [SAB 99].

L'avantage de cette technique est de permettre de ré-aligner temporellement le scénario temporel de façon non bloquante. Notons que les délais imprévisibles sont compensés en priorité au moyen des objets discrets. Ces objets particuliers comme le texte et les délais agissent dans le graphe comme des *élastiques* qui amortissent les retards imprévisibles.

Lorsque ces ajustements prédictifs ne sont plus suffisants, il est nécessaire de mettre en œuvre une *stratégie de réparation* qui tend à minimiser les désynchronisations issues des comportements incontrôlables [LAY 00]. Plutôt que de laisser ces désynchronisations se propager, l'algorithme identifie les objets concernés et leur applique un décalage temporel permettant de les resynchroniser plus rapidement. La Fig. 7 illustre le résultat obtenu grâce à l'intégration de l'approche par compensation et de l'approche par réparation.

Dans la 7 b represente l'exécution d'un scénario sans formatage dynamique ni réparation.On constate que la période de désynchronisation provoquée par le retard de la terminaison de l'objet X subsiste jusqu'à la fin de la présentation. Dans la 7 c, l'algorithme de formatage dynamique a été appliqué : la période de désynchronisation a été réduite en tirant partie de la flexibilité des objets B, F et K. Enfin, dans la 7 d, l'algorithme de réparation a permis de réduire encore la désynchronisation en décalant le lancement de G pour assurer sa synchronisation avec E. La désynchronisation est représentée en deux dimensions dans les courbes de la 7 : leur durée ainsi que leur intensité (évaluée en nombre d'objets désynchronisés).

3.3.4. *Gestionnaire des objets média*

Le gestionnaire d'objets a pour rôle de créer et de contrôler les objets média nécessaires à la présentation du document. Cette gestion concerne tous les traitements qui sont indépendants du contexte d'exécution proprement dit (périphériques utilisés, espace de stockage local ou distant). Ce sont les gestionnaires d'accès et des périphériques qui contrôlent ces traitements spécifiques (voir section 3.4).

Les natures différentes et hétérogènes des objets média amènent à les classifier en différentes catégories. L'approche objet paraît la mieux adaptée à ce type de classification. En effet, l'approche objet permet d'encapsuler les objets média dans des classes contenant d'une part des données représentant les

attributs des objets média et d'autre part des méthodes représentant leurs comportements.

La classification s'appuie sur l'analyse des caractéristiques des médias. Ainsi, un objet média de base peut être : visible, audible ou opaque. Les objets média opaques sont les objets sur lesquels nous n'avons pas de vrai contrôle (plug-ins et applets). De plus, ces objets média peuvent être continus ou discrets (un clip vidéo est continu alors qu'une image est discrète). Enfin, la propriété d'interactivité des objets (les boutons par exemple) est également considérée.

Le résultat est une hiérarchie de classes qui prend en compte différents aspects des objets média comme celle utilisée dans le système Madeus représentée sur la Fig. 8.

À toute classe d'objet média, on peut associer un graphe de transitions qui décrit le comportement de ses objets en fonction des événements qui leur sont adressés et des actions effectuées ou subies. La Fig. 9 montre le graphe de transitions retenu pour les objets média dans Madeus.

La classe *Interval* (cf. Fig. 8) implémente le graphe de transitions des objets média : l'état courant d'un objet est mémorisé dans l'attribut *state* de la classe *Interval* et le comportement d'un objet correspondant à chaque état du graphe de transitions est réalisé par une méthode appropriée (*mapObject*, *playObject*, etc.) appelée par l'objet de type *StateHandler*. Les différents états d'un graphe de transitions sont :

- **Unmapped** : si l'objet n'est pas exécuté et qu'aucune ressource de présentation (fenêtre d'affichage, canal audio, module de décompression, etc.) ne lui est allouée ou s'il est terminé et que toutes ses ressources de présentation sont libérées.
- **Mapped** : l'objet est prêt à être exécuté. Le flot de données qu'il doit interpréter est prêt et les ressources de présentation sont allouées.
- **Active** : l'objet média est en cours d'exécution.
- **Suspended** : l'objet média est suspendu pour des raisons externes (un lecteur a stoppé l'objet) ou internes (pour une synchronisation : l'objet est bloqué jusqu'à l'occurrence d'un événement donné).
- **Terminated** : l'exécution de l'objet média s'est terminée mais ses ressources de présentation sont encore allouées.
- **Stored** : un objet média atteint cet état lorsque, par exemple, un hyperlien vers un autre document est activé. A ce moment, l'exécution de l'objet est interrompue et il passe dans l'état *Stored*. Les informations sur la présentation et l'exécution de l'objet média sont alors stockées (e.g. numéro de l'image courante pour un clip vidéo). Ces informations peuvent être réutilisées pour réactiver l'objet à partir du moment exact de sa suspension.

3.4. *Contrôleur de l'exécution d'une présentation multimédia*

Ce composant se charge d'effectuer les fonctions de présentation au niveau du système d'exploitation et du système de communication. En effet, il gère :

- la création des processus qui se chargent de la présentation des différents objets média,
- la communication entre ces processus et l'ordonnanceur,
- le contrôle des périphériques de présentation,
- et l'accès aux objets média locaux et distants.

L'organisation de ce composant est décrite dans la Fig. 10. Celui-ci se compose d'un gestionnaire d'accès, d'un gestionnaire de traitement et d'un gestionnaire de périphériques que nous décrivons ci-dessous.

3.4.1. *Gestionnaire d'accès*

Le gestionnaire d'accès est la partie de l'application qui est responsable du chargement des objets média et, à ce titre, il joue un rôle important dans la performance du système de présentation. Le temps qui s'écoule entre le lancement d'une requête de chargement et le chargement effectif (*round trip*) doit être le plus court possible.

Un objet média peut être stocké sur un serveur local ou sur un serveur distant. Afin de limiter ces délais, en particulier pour les objets média continus (comme l'audio et la vidéo), ils doivent être chargés et présentés au fur et à mesure sous forme d'un flot limité (en cas de clips enregistrés) ou infini (en cas de diffusion en direct). Un problème directement lié à ce processus cyclique consiste d'une part à assurer la disponibilité des données et d'autre part au respect des contraintes temps réel liées à leur traitement (pour éviter des effets de gigue, cf. section 3.3). Cela illustre le lien étroit qui existe entre les flots de données et les flots d'exécution (traitements) dans un système multimédia. Par conséquent, il est indispensable de pouvoir synchroniser ces deux flots. Un choix important est alors de déterminer le lieu où doit s'opérer cette synchronisation : du côté client ou du côté serveur.

La Fig. 11 illustre l'opération d'accès aux objets média qu'ils soient locaux ou distants. À partir d'une instance d'un objet média créé par le gestionnaire d'objets, l'opération d'accès aux données commence par l'envoi d'une demande adressée au gestionnaire d'accès. Ensuite, ce gestionnaire localise les données média (résolution du nom du serveur) et identifie le protocole à partir de l'URI fourni. Il crée alors un tampon de données qui stocke provisoirement les données

récupérées. La taille de ce tampon est déterminée soit par le gestionnaire de synchronisation si la technique de préchargement est appliquée, soit par l'objet média lui-même si une stratégie spécifique lui est appliquée. Par exemple, certains plugins, comme mpegTV, prennent en charge la gestion de leur tampon. Dans ce cas, le gestionnaire d'accès se limite à transmettre les données au plugin dès leur réception.

Les besoins d'une application multimédia en terme d'accès distant peuvent être subdivisés en deux catégories. La première catégorie concerne les objets média dont la présentation ne dépend pas d'un flot continu de données externe (comme les objets média discrets et le fichier de spécification du document). Les images stockées sur des serveurs distants forment un cas particulier de cette catégorie. En effet, elles ne nécessitent qu'un seul accès distant mais leur traitement peut être effectué de façon progressive (comme le cas du format PNG. La deuxième catégorie correspond aux média continus dont le traitement, et donc le transfert de leur contenu, est soumis à des contraintes temps réel. Des protocoles spécifiques ont été spécifiés pour permettre de transmettre ces médias continus. Ils sont décrits dans la section ci-dessous.

3.4.2. *Protocole d'accès distant pour médias continus*

RTSP [SCH 98] est le protocole temps réel conçu pour transférer et contrôler un ou plusieurs flux synchronisés d'objets média continus comme la vidéo et l'audio. Le protocole est fondé sur le modèle client-serveur. Il permet le contrôle de la présentation d'un objet média en agissant sur le flot de données transmis entre le client et le serveur. Autrement dit, les commandes de contrôle de la présentation d'un objet média (comme démarrer, stopper, faire une pause, etc.) sont traduites sous forme de requêtes d'accès RTSP envoyées par le client et exécutées par le serveur.

RTSP est un protocole avec états qui utilise la notion de session. Cette caractéristique est importante pour effectuer le transfert de média continus. En effet, la présentation d'objets de type média continus est caractérisée par une évolution entre différents états de présentation, comme les états Unmapped, Mapped, Active, Terminated et Suspended (voir les états d'exécution des objets en 9). L'évolution des transferts de données effectués au moyen du protocole RTSP est décrit par une machine d'états associée au client et au serveur. Le serveur change d'état quand il reçoit une requête du client, et le client change d'état quand il envoie une requête au serveur.

Pour chaque transfert de données, le serveur RTSP maintient une session qui est désignée par un identificateur unique. Pendant une session, le client RTSP peut ouvrir et fermer plusieurs connexions de transport avec le serveur RTSP

en utilisant l'identificateur de cette session et en gardant la correspondance entre l'état courant du côté client et celui du côté serveur. Ces connexions sont établies afin d'échanger les requêtes RTSP entre le client et le serveur. Ces requêtes peuvent être des requêtes d'initialisation de session, de transfert de données et de contrôle de ces flots de données.

Les différentes requêtes RTSP sont définies à travers un ensemble de méthodes. Les principales méthodes sont :

- **DESCRIBE**. Elle est utilisée par le client pour récupérer la description d'une présentation ou d'un objet média, identifié par un URI, sur un serveur RTSP.
- **SETUP**. Elle est utilisée par le client pour spécifier au serveur les paramètres de transport du flot média, comme par exemple le type de protocole de transport (RTP), le mode de transport (point à point ou multipoint) et le numéro du port de communication.
- **PLAY**. Elle signale au serveur qu'il peut commencer à envoyer les données via le mécanisme spécifié dans la méthode SETUP. Elle permet également de jouer un ou plusieurs sous-intervalles de la durée d'un objet média, par exemple jouer une audio à partir de la 10ème seconde jusqu'à la 20ème seconde.
- **PAUSE**. Elle est utilisée par le client pour demander au serveur d'interrompre temporairement le transfert du flux média. Le client peut reprendre la transmission du flux en envoyant la requête PLAY au serveur.
- **TEARDOWN**. Elle est utilisée par le client pour demander au serveur d'arrêter définitivement le transfert du flux média.

La notion de session dans RTSP est très importante pour le transfert de données sous contraintes temporelles. Car, en plus du transfert de données proprement dit, RTSP maintient une horloge qui est associée à chaque objet média transmis. C'est cette horloge qui permet de déterminer, à chaque instant, si une portion de donnée à transmettre a déjà dépassé son échéance. Dans ce cas, il est inutile de l'envoyer au client qui la rejetterait à sa réception. RTSP permet donc la mise en oeuvre de stratégies de filtrage des données du côté du serveur et donc à la source.

Au niveau du protocole, RTSP permet de surveiller la qualité de la transmission à travers RTCP. Pour cela, le client émet périodiquement à destination du serveur (boucle de contrôle ou feed-back) des rapports sur la qualité de la transmission, contenant le pourcentage et le nombre de paquets perdus, le délai de transmission, la gigue, etc. Le serveur reçoit ces rapports et apprécie s'il y a congestion. Dans ce cas, la source réduit alors le débit émis (contrôle de flux). La réduction de débit se fait par filtrage des données au niveau des fichiers

d'audio et de vidéo, par exemple, en éliminant des images dans le cas de la vidéo. Dans le gestionnaire d'accès, la diminution du débit, et donc l'adaptation qui en résulte, a lieu chaque fois qu'une résynchronisation avec le serveur est nécessaire.

Un serveur RTSP doit donc avoir une connaissance du format pour fournir de bonnes stratégies d'adaptation de la qualité. Et même dans ces conditions, un client peut avoir une stratégie différente qui n'est pas disponible au sein du serveur. C'est cet aspect qui constitue la principale limitation de la gestion de la qualité de service avec le protocole RTSP.

3.4.3. *Gestionnaire de traitement*

Le but du gestionnaire de traitement est de fournir les mécanismes permettant l'exécution de différents types et formats d'objets média.

Afin d'être en mesure de supporter différents formats, ce gestionnaire s'appuie sur un modèle à objet tel que décrit en section (voir figures Fig. 8 et Fig. 9) dans lequel les principales abstractions définies sont les types et les méthodes correspondant au comportement générique des objets média (par exemple *PlayObject*, *PauseObject*, etc.).

Les différents types d'objets média nécessitent des traitements variés lors de leur présentation (comme par exemple la décompression des données, l'allocation d'une table de couleurs, etc.) qui correspondent à autant de codes binaires différents liés (dynamiquement ou non) aux objets média correspondants.

Ce gestionnaire assure également le traitement des événements. En effet, au niveau le plus bas du système, tous les changements d'état des objets ainsi que les événements externes se traduisent par l'échange d'un ensemble d'*événements*. L'occurrence de chaque événement implique le déclenchement d'un traitement particulier au sein du système de présentation. La gestion d'événements est donc un élément central dans la mise en œuvre du gestionnaire de présentation. Dans Madeus par exemple, le modèle de communication s'appuie sur des événements asynchrones car les entités actives du système de présentation (ordonnanceur, objets média, accès aux flots de données) ont des contraintes temps réel fortes.

3.4.4. *Gestionnaire de périphériques*

La restitution des données à l'utilisateur à travers les périphériques de la machine est effectuée par deux gestionnaires principaux : le gestionnaire graphique et le gestionnaire audio. Ces gestionnaires fonctionnent comme des serveurs qui

reçoivent des demandes de création ou de suppression de modules destination attachés à chaque objet de base. Ils permettent aussi de prendre en compte les relations graphiques comme le recouvrement d'objet sur l'écran ainsi que sur le périphérique audio (mixage, musique de fond, etc.).

4. Conclusion

Dans ce chapitre, nous avons d'abord étudié les langages de spécification de scénarios temporels pour les documents multimédia à travers les trois approches prédominantes : les approches opérationnelles à travers le langage MHEG, les approches déclaratives au moyen de SMIL ainsi que les approches à base de contraintes. Nous avons ensuite étudié le support de présentation que nécessitent ces langages tant au niveau de l'architecture qu'au niveau des fonctionnalités. Les nouvelles évolutions du web montrent que la tendance est à l'accroissement de la complexité des systèmes de présentation. Il existe en effet de plus en plus de formats et de fonctionnalités à intégrer, comme par exemple, HTML, CSS et les dessins vectoriels qui sont devenus au fil des années très populaires. Les documents multimédia structurés deviennent ainsi un surensemble des documents structurés traditionnels et donc extrêmement plus complexes à mettre en œuvre.

Le schéma général d'architecture de présentation multimédia décrit ici doit donc évoluer pour prendre en compte les besoins qui émergent notamment du fait de la prolifération de appareils mobiles. En effet, ces appareils mobiles ont des capacités graphiques et une puissance de calcul nettement inférieures aux machines de type PC traditionnels. Ainsi, les besoins d'adaptation et des modes de représentation qui en découlent doivent pouvoir être pris en compte au niveau des systèmes de présentation, ce qui ouvre le champ à un grand nombre de travaux qui touchent aux différents composants identifiés dans la section précédente : formatage spatial, gestion de la synchronisation et de la navigation, gestion de la qualité de service, protocoles réseau et gestion des contenus.

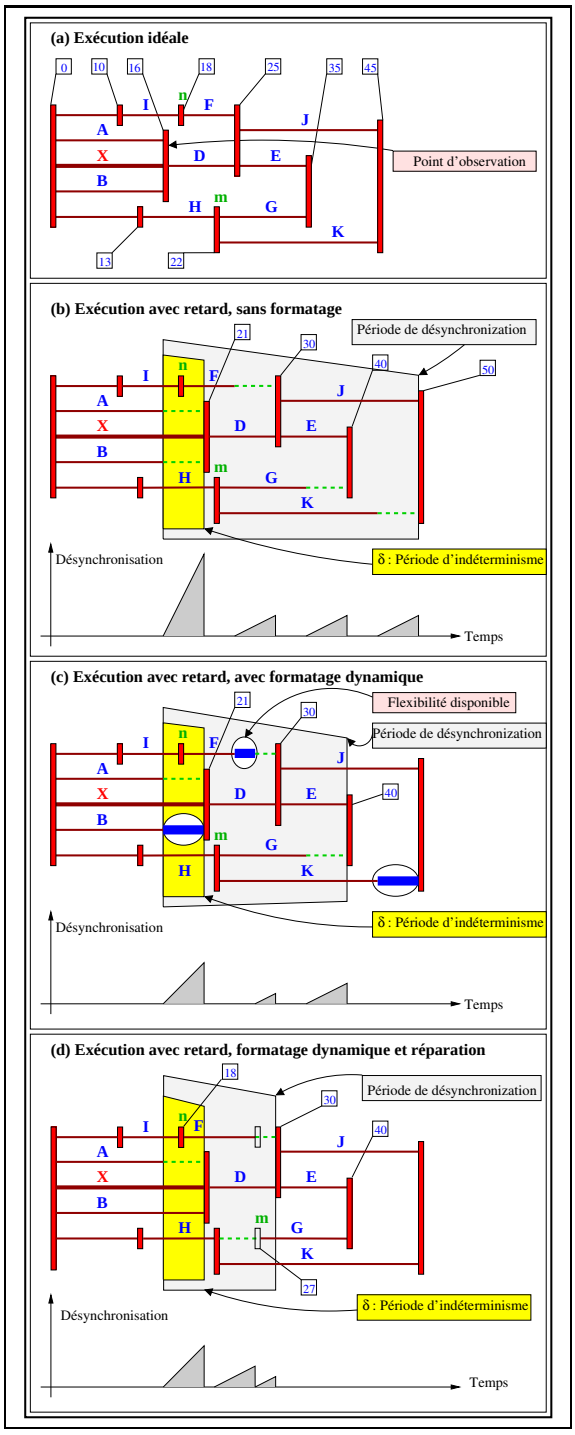

Figure 7. *Gestion de l'indéterminisme par l'intégration de techniques de compensation et de réparation*

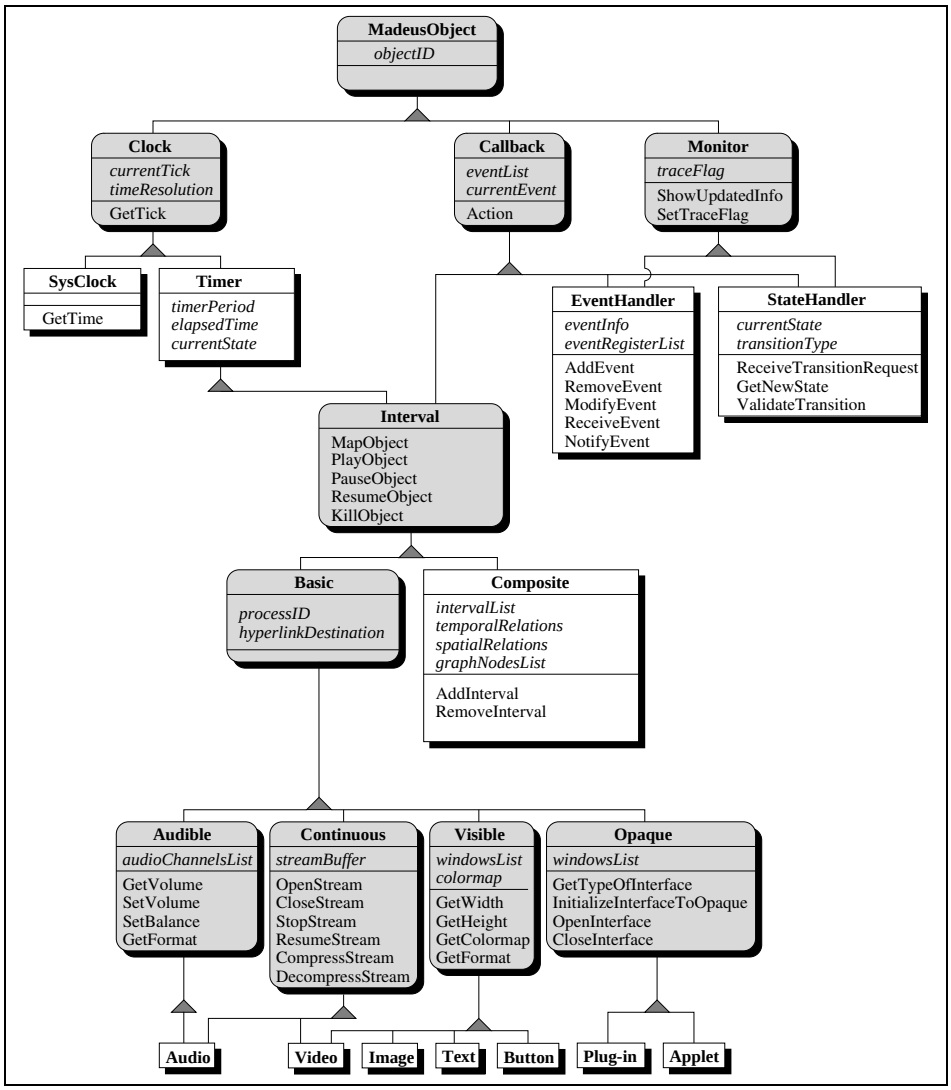

Figure 8. *Hiérarchie des classes Madeus*

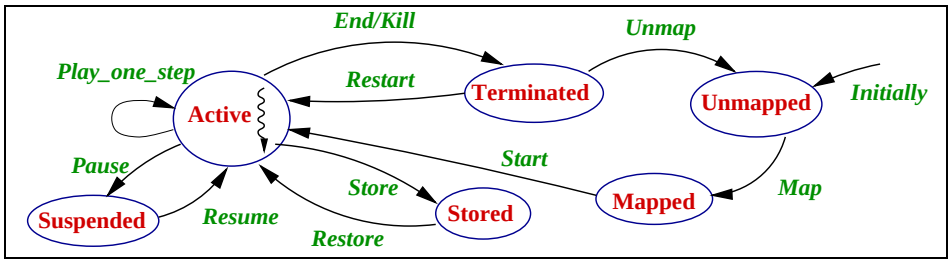

Figure 9. *Graphe de transitions des objets de type Interval dans Madeus*

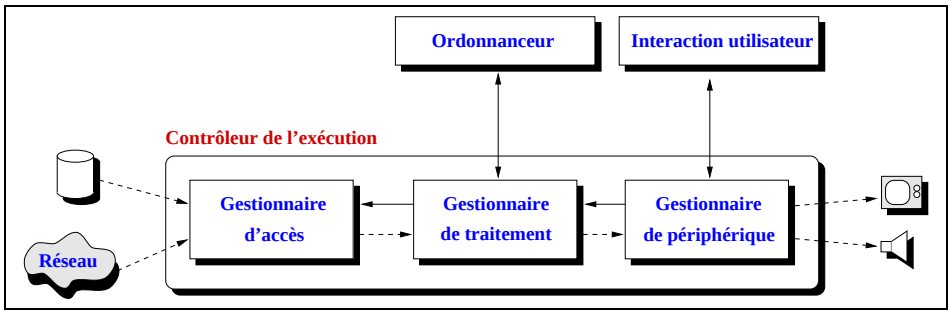

Figure 10. *Contrôleur de l'exécution*

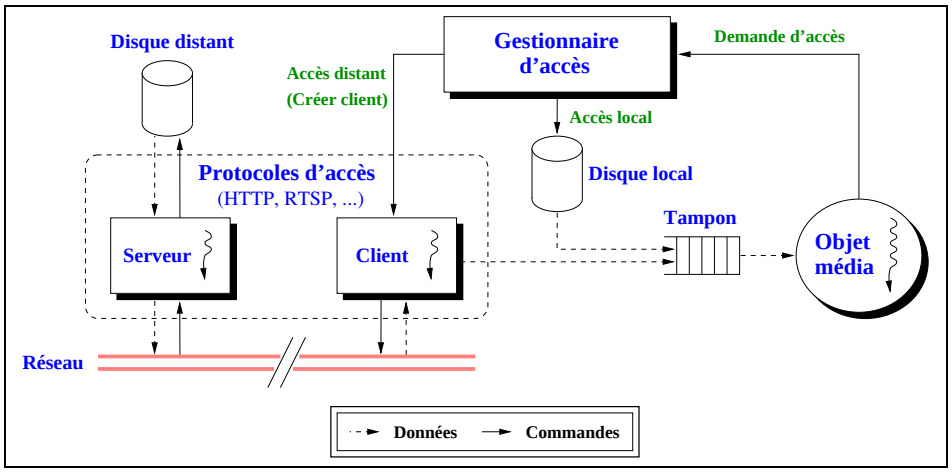

Figure 11. *Gestionnaire d'accès*

15 Septembre 1999

Références

[ALL 83] J. F. Allen, "Maintaining Knowledge about Temporal intervals", *CACM,* vol. 26 , num. 11, pp. 832-843, 1983.

[BIT 96] Bitzer H.W., Gran C. et Hofrichter K., "MAJA: MHEG Applications Using JAVA Applets", *In proceedings of Real Time Multimedia and the World Wide Web, W3C Workshop (RTMW'96),* Sophia-Antipolis, France, octobre 1996.

[BUC 92] C. Buchanam, P. T. Zellweger, "Specifying Temporal Behavior in Hypermedia Documents ", *Proc. of the ACM Conf. on Hypertext,* pp. 262-271, décembre 1992.

[DUD 95] A. Duda, C. Keramane, "Structured Temporal Composition of Multimedia Data", *Proc. of IEEE International Workshop on Multi-Media Data Base Management Systems,* Blue Mountain Lake, NY, août 1995.

[FLI 95] Flinn S. et Kellog S., "Coordinating Heterogeneous Time-Based Media Between Independant Applications", *In proceedings of the ACM Multimedia'95,* pp. 435-444, ACM Press, California, USA, novembre 1995.

[GEY 97] Geyer L., Baentsch M., Baum L., Molter G., Rothkugel S. et Sturm P., "MHEG in Java – Integrating a Multimedia Standard into the Web", *Poster session in the 6th International World Wide Web Conference,* California, USA, avril 1997.

[ISO 97] ISO/IEC JTC1/SC18/WG8 N1920, *Information Technology: Hypermedia/ Time-based Structuring Language (HyTime), Second edition,* ISO/IEC, août 1997. http://www.ornl.gov/sgml/wg8/docs/n1920/html/n1920.html.

[JOU 98] M. Jourdan, N. Layaïda, C. Roisin, L. Sabry-Ismail, L. Tardif, "Madeus, an Authoring Environment for Interactive Multimedia Documents", *ACM Multimedia'98,* pp. 267-272, ACM, Bristol (UK), septembre 1998.

[KIM 95] M. Y. Kim, J. Song, "Multimedia Documents with Elastic Time", *ACM Multimedia'95,* pp. 143-154, San Francisco, novembre 1995.

[KOU 96] I. Kouvelas, V. Hardman, A. Watson, "Lip Synchronisation for use over the Internet: Analysis and Implementation", *Proceedings, IEEE GLOBECOM'96,* novembre 1996.

[LAY 97] N. Layaïda., *Madeus : système d'édition et de présentation de documents structurés multimédia,* Thèse de doctorat, Université Joseph Fourier, Grenoble, 12 juin 1997.

[LAY 00] N. Layaïda, L. Sabry-Ismail, C. Roisin, *Dealing with uncertain durations in synchronized multimedia presentations,* Multimedia Tools and Applications Journal, Kluwer Academic Publishers, 2000.

[LIT 93] T. D. C. Little, A. Ghafoor, "Interval-Based Conceptual Models for Time-Dependent Multimedia Data", *IEEE Transactions on Knowledge and Data Engineering (Special Issue : Multimedia Information Systems),* vol. 5, num. 4, pp. 551-563, août 1993.

[MAC 99] Macromedia, *Flash and Director,* http://www.macromedia.com/software/.

[MEY 95] T. Meyer-Boudnik, W. Eeffelsberg, "MHEG Explained", *IEEE Multimedia Magazine,* vol. 2, num. 1, pp. 26-38, 1995.

[OWE 98] P. Dwezarski P., M. Diaz et C. Chassot, "A Time-Efficient Architecture for Multimedia Applications", *IEEE Journal on Selected Areas in Communications,* vol. Vol. 16 , num. No. 3, pp. 383-396, avril 1998.

[REA 99] Real Networks, *RealPlayer G2,* http://www.real.com/g2/index.html.

[ROS 93] G. van Rossum, J. Jansen, K. Mullender, D. Bulterman, "CMIFed: a presentation Environment for Portable Hypermedia Documents", *Proc. of the ACM Multimedia Conf.,* California, 1993.

[SAB 97] L. Sabry-Ismail, C. Roisin, N. Layaïda, "Navigation in Structured Multimedia Documents using Presentation Context", *Hypertextes et Hypermédias,* Hermès, ed., Paris, septembre 1997.

[SAB 99] L. Sabry-Ismail, *Schéma d'exécution pour les documents multimédia distribués,* Thèse de doctorat, Université Joseph Fourier, Grenoble, 25 janvier 1999.

[SAB 98] L. Sabry-Ismail et R. Guetari, "Le modèle objet Madeus", *L'objet : logiciel, bases de données, réseaux (Les Représentations par Objets en Conception), Editions Hermes,* vol. Volume 4 , num. Numéro 2, juin 1998.

[SCH 98] H. Schulzrinne, A. Rao, R. Lanphier. *Real Time Streaming Protocol (RTSP)*, RFC2326, IETF Internet Draft, avril 1998.

[SMI 98] W3C Recommendation, *Synchronized Multimedia Integration Language (SMIL) 1.0 Specification,* http://www.w3.org/TR/REC-smil, 15-June 1998.

[SMI 99] W3C Working Draft, *Synchronized Multimedia Integration Language (SMIL) Boston Specification,* http://www.w3.org/TR/smil-boston, 20-August 1999.

[WIR 97] S. Wirag, "Modeling of Adaptable Multimedia Documents", *Interactive Distributed Multimedia Systems and Telecommunication Services; 4th International Workshop, IDMS'97,* pp. 420-429, Darmstadt, septembre 1997.

Index