The University of Osaka
Institutional Knowledge Archive

Title	分散型マルチメディアシステムの構築に関する研究
Author(s)	藤川, 和利
Citation	大阪大学, 1993, 博士論文
Version Type	VoR
URL	https://doi.org/10.11501/3066006
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

The University of Osaka

分散型マルチメディアシステムの構築に関する研究

藤川 和利

1993 年 2 月

①

分散型マルチメディアシステムの構築に関する研究

藤川 和利

1993年2月

内容梗概

本論文では，ワークステーションがローカル・エリア・ネットワーク (LAN) に接続された分散環境におけるマルチメディアシステムのための簡潔で拡張性のあるドキュメントの表現モデルとそれを実現するためのシステムアーキテクチャを提案する．

まず，提案するドキュメントの表現モデルは，オブジェクト指向の概念を導入し，マルチメディア情報とその情報に対する操作・手続きをカプセル化してひとまとまりのオブジェクトとして扱うことにより，映像やテキストなどのメディアの相違を抽象化する．また，ハイパーテキストモデルを採り入れており，マルチメディア情報であるオブジェクトをノードとして捉え，オブジェクト間にリンクを付与することで素材の表示順序などの情報間の関連を表すことができるため，マルチメディア情報間の関係を簡潔に表現することが可能となる．この実現のために，拡張したリンク構造を提案している．さらに，映像に映し出される人物などや複雑な図形の一部分に対して，リンクを付与することをも可能にしている．

次に，分散型マルチメディアシステムのアーキテクチャを示す．ここでは，それぞれの種類のマルチメディア情報を扱うための複数のクラスプロセスと，オブジェクト間のリンク情報を管理するためのサーバプロセスがメッセージをやりとりすることでマルチメディアドキュメントを表示する．クラスプロセスはオブジェクトとして表される個々のマルチメディア情報の表示や編集等を実現する．サーバプロセスは，ドキュメントを入力するためのユーザインタフェース部，リンク情報を管理・実行するリンク管理部，ドキュメントを含めたマルチメディア情報全体を管理するデータベース部からなる．サーバプロセスの構築には，商用のオブジェクト指向データベースシステムを用いそれぞれのプロセスにおけるオブジェクトの管理および交換を行なわせるが，クラスプロセスにおけるメソッド実行部は新たに実現している．

最後に，分散型マルチメディアシステムにおいて重要なマルチメディア情報間の同期のモデル化および実現について述べる．分散環境における同期の問題点として複数プロセスによって表示させた場合の同期のずれ，および，計算機の負荷の変動による実時間性のずれをとりあげ，それらを解決するためのマルチメディア情報間の同期モデルを提案し，解決する方法を示している．マルチメディア情報の表示において，単に実時間性を優先した表示方法を実現するのみでなく，実時間性よりも

むしろ複数のメディア間の同期を優先した表示方法も提案し，その有用性を確かめている．

UNIX ベースのワークステーション上でメディアの種類として，映像，アニメーション，音声，音楽，図形，テキストを扱うことのできるプロトタイプシステム “Harmony” を構築し，これらの間の同期が実現可能であることを確認した．

関連発表論文

1. 学術論文誌，国際会議録

[1-1] 藤川和利, 下條真司, 松浦敏雄, 西尾章治郎, 宮原秀夫, " オブジェクト指向に基づくハイパーメディアシステム Harmony の構築," 電子情報通信学会論文誌 (D-I), Vol.J75-D-I, No.11, pp.1015-1024 (1992-11).

[1-2] 藤川和利, 下條真司, 松浦敏雄, 西尾章治郎, 宮原秀夫, " 分散型ハイパーメディアシステム Harmony における情報間同期機構の実現," 電子情報通信学会論文誌 (D-I) (投稿中).

[1-3] Shinji Shimojo, Toshio Matsuura, Kazutoshi Fujikawa, Shojiro Nishio, and Hideo Miyahara, "A New Hyperobject System Harmony: Its Design and Implementation," International Conference on Multimedia Informaiton Systems, McGraw-Hill, pp.243–257, Singapore (Jan. 1991).

[1-4] Kazutoshi Fujikawa, Shinji Shimojo, Toshio Matsuura, Shojiro Nishio, and Hideo Miyahara, "Multimedia Presentation System Harmony with Temporal and Active Media," Proceedings of the Summer 1991 USENIX Conference, pp.75–93, Nashville, TN. (June 1991).

[1-5] Shinji Shimojo, Toshio Matsuura, Kazutoshi Fujikawa, Shojiro Nishio, and Hideo Miyahara, "Architectural Issues on Multimedia Presentation System Harmony," Proceedings of the IFIP TC8/WG8.1 Working Conference the Object Oriented Approach in Information Systems, North-Holland, pp.381-402 (Oct. 1991).

2. 研究会会議録，口頭発表

[2-1] 下條真司, 藤川和利, 宮原秀夫, "分散オブジェクト指向システム O^2NE," 情報処理学会, マルチメディア通信と分散処理研究会, SIGDSP41-4 (1989-03).

[2-2] 藤川和利, 下條真司, 松浦敏雄, 香取啓志, 小島増生, "ワークステーション上での映像編集システムの試作," 日本 UNIX ユーザ会, 第 14 回 UNIX シンポジウム予稿集, pp.67–75 (1989-11).

[2-3] 藤川和利, 梶本雅人, 有吉勇介, 下條真司, 松浦敏雄, 西尾章治郎, 宮原秀夫, "マルチメディアプレゼンテーションシステム Harmony," 電子情報通信学会, ヒューマンコミュニケーション研究会, 電子情報通信学会技術研究報告, HC90-7 〜14 (1990-07).

[2-4] 藤川和利, 有吉勇介, 下條真司, 松浦敏雄, 西尾章治郎, 宮原秀夫, "協調型プレゼンテーション作成支援システム HarmonyII," 日本ソフトウェア科学会, 第 10 回ソフトウェア研究会, SW-92-10 (1992-04).

[2-5] 藤川和利, 田島孝一, 有吉勇介, 下條真司, 松浦敏雄, "実時間性を考慮した分散型マルチメディアシステムの構築," 日本 UNIX ユーザ会, 第 20 回 UNIX シンポジウム予稿集, pp.161–172 (1992-10).

[2-6] 藤川和利, 東浩, 下條真司, 宮原秀夫, "分散型オブジェクト指向システムにおけるインスタンスの実現," 情報処理学会, 第 37 回全国大会, 2Y-2, pp.601–602 (1988-09).

[2-7] 藤川和利, 下條真司, 松浦敏雄, 西尾章治郎, 宮原秀夫, " オブジェクト指向に基づくハイパーメディアシステム Harmony," システム制御情報学会, 第 34 回システム制御情報学会研究発表講演会 (1990-05).

[2-8] 表武史, 有吉勇介, 藤川和利, 下條真司, 江澤義典, 宮原秀夫, " ポータブルマルチメディアプレゼンテーションシステム Harmony のアーキテクチャ," 情報処理学会, 第 45 回全国大会, 1R-3, pp.4-103–4-104 (1992-10).

目次

1 序論 **1**

1.1 分散型マルチメディアシステム 3

1.2 関連する研究および問題点 4

2 マルチメディア情報のドキュメントモデル **8**

2.1 モデル化における問題点 8

2.2 ハイパーオブジェクトモデル 9

2.2.1 リンク 11

2.2.2 メディアインフォメーションオブジェクト 11

2.2.3 メディアオブジェクト 12

2.2.4 サブオブジェクト 14

2.2.5 シナリオ 14

2.3 評価 15

2.3.1 情報の種類と粒度 15

2.3.2 リンクの意味記述と時間記述 15

3 分散型マルチメディアシステムのアーキテクチャ **17**

3.1 システム構築の基本方針 17

3.2 Harmony のシステムアーキテクチャ 19

3.2.1 オブジェクト指向データベース Harmony/DB 19

3.2.2 リンクマネージャ Harmony/LM 21

3.2.3 ユーザインタフェース Harmony/UI 21

3.3 マルチメディアドキュメントの記述 22

3.3.1 関連するドキュメント記述モデル 22

3.3.2 シナリオの記述 23

3.4 システムの実装 27
3.5 マルチメディア情報の管理 33
3.5.1 クラス定義 33
3.5.2 オブジェクト管理 35
3.6 分散型システムのオブジェクト指向データベースによる実現 39
3.6.1 オブジェクトの実行 42
3.7 クラスライブラリの設計 44
3.7.1 アプリケーションプログラム・インタフェース 44
3.8 評価 44
3.8.1 拡張性と移植性 44
3.8.2 情報の管理機構 46
3.8.3 プロトタイプシステムの評価 46

4 分散型マルチメディアシステムにおける情報間の同期 49
4.1 同期実現の基本方針 49
4.2 マルチメディア情報の同期モデル 50
4.3 関連した同期モデル 51
4.4 同期機構の実現 53
4.4.1 ハイパーオブジェクトモデルの拡張 53
4.4.2 開始点同期の実現 54
4.4.3 実時間連続同期への対応 58
4.4.4 適応型連続同期への対応 58

5 結論 62

謝辞 64

参考文献 64

第 1 章

序論

近年の技術革新によりワークステーションにビデオボードやデジタル・シグナル・プロセッサ (DSP) などが搭載され，X ウィンドウ [31] に代表されるようなウィンドウシステム上でマルチメディア情報を手軽に扱えるようになってきている．また，JPEG[37] や MPEG[22] などのデータ圧縮方式を用いることにより通常のハードディスク上に映像情報を蓄積することができるようになり，ローカル・エリア・ネットワーク (LAN) に接続されたワークステーション間でネットワークを介してマルチメディア情報のやりとりを行なうことが可能となってきている．このような状況下では，マルチメデイア情報を用いた CAI(Computer Aided Instruction) システムや DTP(Desk Top Presentation) システム，ネットワークを利用した電子メール，電子ニュース，会議システム等で大きな効果をあげることが期待できる．そこで，本論文では，さまざまなマルチメディアシステムを分散環境上で実現するために，簡潔で拡張性のあるドキュメントモデルを提案し，そのモデルに基づいたアプリケーションのアーキテクチャを示す．また，マルチメディアを扱う上で重要な情報間の同期のモデル化，および，その実現方法についても提案する．さらに，マルチメディアシステムの一例としてマルチメディアプレゼンテーションシステムを試作することで本論文で提案するモデルやアーキテクチャの有用性を示す．

第 2章では，さまざまなマルチメディア情報を統一的かつ簡潔に扱うことができるように，オブジェクト指向の概念をハイパーテキストモデル上に展開した “ハイパーオブジェクトモデル” を提案する．ハイパーオブジェクトモデルでは，マルチメディア情報の各々をオブジェクトとして扱い，ハイパーテキストモデルに基づき，オブジェクト間にメッセージ交換で実現されるリンクを付与することでオブジェク

ト間の表示の順序・同期といった関連を表現することでマルチメディアドキュメントを構成することができる．また，テキスト中の一単語や映像に現れる人物，図形の一部分をもオブジェクトとして統一的に扱うことができる．さらに，ハイパーオブジェクトモデルでは，リンク情報をマルチメディア情報から独立して存在させることで，複数の利用者が共通のマルチメディア情報を用いて，ハイパーテキストモデルによる独自のマルチメディアドキュメントを作成することを可能にしている．

第3章に，分散環境上で実現されるマルチメディアシステムのアーキテクチャを提案する．オブジェクトとして表現される構造の異なる種々のマルチメディア情報をまとめて管理し，新たなメディアの情報を容易に加えて拡張することができるように，商用のオブジェクト指向データベースシステムを用いる．現在までに開発されている商用のオブジェクト指向データベースシステムでは，データベースカーネルにオブジェクトのメソッド実行やメッセージ交換をする機能が提供されていないため，オブジェクトの実行機構を実現するプロセスを用意する．ここでは，メディアの種類によって表示方法等が異なるため，メディアの種類ごとにプロセスを実現し，これらのプロセス間でメッセージ交換を行なうことによって，マルチメディア情報を表示する．このように，複数のプロセスで実現することにより，マルチメディアシステムに要求される複数のマルチメディア情報を同時に表示するといった処理の並列性が実現される．

第4章に，時系列メディアを扱う上で重要な機能である情報間の同期方法について述べる．ここでは，複数のUNIXプロセスが相互に通信する分散型マルチメディアシステムでのマルチメディア情報の表示における重要な同期として，実時間同期，適応型連続同期，開始点同期の3つを実現する方法を示している．開始点同期は，マルチメディア情報が時間通りに表示されるように，あらかじめマルチメディアドキュメントの構成を解析し，情報の表示開始・終了時刻を決定して，対応するプロセスにそれらの時間を知らせることで実現する．実時間同期は，これらのプロセスに定期的に表示の進み具合を監視する機能を組み込み，表示が遅れていれば，その時間まで早送りことで実現する．適応型連続同期では，計算機の負荷増大などによる表示の遅れを検出し，プロセス間でメッセージをやりとりすることで全体の表示速度を遅らせ異なるメディア間の意味的な同期を保つ．さらに，この機能を利用者からの対話的操作によるプレゼンテーション進行の変化にも対応させている．

さらに，第5章では，各章を踏まえて，本論文のまとめと今後の課題等を議論する．

1.1 分散型マルチメディアシステム

さまざまなワークステーションで映像や音声といったマルチメディア情報が扱えるようになってきているが，単一のワークステーションですべてのメディアが扱えるものは少ない．また，すべてのメディアが扱える場合であっても，現状ではワークステーションに対する負荷が非常に大きくなり，十分な操作環境を実現することが困難である．しかし，このようなワークステーションがLANに接続された分散環境では，複数のワークステーションが協調して一つのマルチメディアドキュメントを表示したり作成したりすることが考えられる．

「分散型マルチメディアシステム」とは，LANに接続されている複数のワークステーションを利用し，これらのワークステーションが協調しあってネットワーク上のさまざまな資源を用いて，一つのマルチメディアドキュメント内の個々のマルチメディア情報の表示や作成等を行なうシステムを意味する．ここでは，多種のマルチメディア情報が統合化されて構成されるひとまとまりのドキュメントを「マルチメディアドキュメント」と呼ぶ．

分散環境上の多種のメディアを効果的に用いたマルチメディアシステムは，CAIやプレゼンテーション，会議システム等の分野で大きな効果をあげることが期待できる．このようなさまざまな分野において有用可能なマルチメディアシステムをより強力で魅力あるものにするために，以下に挙げる項目を実現することが望まれる．

要件 1[時系列メディア]

映像やアニメーションといった時系列をもったメディアを扱う．

要件 2[部分的情報の扱い]

マルチメディアドキュメントを作成する場合，従来のハイパーテキストシステムのようにテキスト中の単語に対して他のテキスト情報を関連づけることができるのと同様に，映像やアニメーションなど時系列情報の一部分に対しても他のマルチメディア情報を関連づけることができる．

要件 3[メディア間の同期の実現]

映像とそれに関連した音楽を同時に再生するといったような時系列メディアの情報間の同期を扱う．

要件 4[メディア情報の統一的扱い]

すべてのメディアの情報を統一的に扱えるようにし，システムの構築および利用の便宜をはかる．

要件 5[開放型システムの実現]

システムを拡張して新たなメディアを採り入れる場合に，システムを大きく再構築ことなく，変更を最小限にとどめ容易に拡張ができるシステムとして実現する．これを開放型システムと呼ぶ．

1.2 関連する研究および問題点

マルチメディアシステムは，主にデータベースとユーザインタフェースといったサブシステムから構成され，オペレーティングシステムの機能を利用することで実現される．以下にこれらのシステムにおける問題点をあげる．

- オペレーティングシステム

  音楽や映像といった時系列情報を計算機に表示させる場合，リアルタイムに再生することが必要となる．また，複数のメディア情報を同期して再生させることが必要になってくる．たとえば，ある映像のシーンに合わせて，特定の音楽を再生することなどである．個々のメディアをリアルタイムに再生することは，現在 DSP や特殊なグラフィックハードウェアを用いて実現されつつある．ところが，複数のメディア情報を同期して再生することは非常に難しい．例えば，UNIX のようなオペレーティングシステムを用いて，複数のプロセスを同期して動かすことや時間的制約をもったプロセスを実行することは，一般的に容易ではない．オペレーティングシステムにこのような能力をもたせることで，アプリケーションにおける時間的制約等が保証される．

- データベース

  マルチメディアドキュメントを構成するためには，多数の素材となるマルチメディア情報を扱う必要がある．多数の素材の中から，その場に応じて最適の素材を的確に選択するためには，マルチメディア情報を効率良くとり出せるデータベースが必要となる．また，現在のところ，これらのマルチメディア情報をすべて単一のハードディスクの中に格納し，リアルタイムに再生することができず，個々のデータの特性に適した蓄積装置を扱わなければならない．たとえ

ば，映像などは光磁気ディスクに蓄積される．従って，これらメディアによる蓄積装置の違いを吸収する必要がある．さらに，マルチメディア情報の場合，従来のデータベースのように値そのものをとり出すだけでは十分でなく，むしろ値を操作することの方が重要である．関係データベースシステムではデータのみを管理しており，データの操作はプログラムで実現しなければならず，システム構築が複雑になる．例えば，音楽や音声の場合もデータそのものをビット列としてとり出しても意味はなく，それを再生して，初めて意味のある情報を構成する．つまり，利用者がこのようなデータを簡潔に扱うには，データとそれに対する操作をまとめて扱う能力がデータベースに必要となる．これらの機能は従来の関係データベースでは実現できず，新たな概念に基づくデータベースが望まれている．

- ユーザインタフェース
  マルチメディアシステムにおいては，利用者が素材を組み合わせて容易にマルチメディアドキュメントを構成できることが重要である．その際，マルチメディアの特徴を生かし，利用者が個々のメディアの違いを意識せず，できるだけ直接操作可能な環境を提供する必要がある．また，音声・映像など時系列情報を取り扱うためのわかりやすいマルチメディアドキュメントの表現モデルが必要となる．

以上の 3 つのサブシステムに対して，実現可能な現状の技術を探ってみると，オペレーティングシステムに関しては Mach[15] などのようにリアルタイム性や時間的制約のあるジョブスケジューリングを実現可能なものが登場してきた．データベースとしては，複雑な構造を持ったデータを蓄積でき，手続きとデータをカプセル化して蓄積できるオブジェクト指向データベースが注目を集めている [16]．また，マルチメディアに適したユーザインタフェースモデルとしてはハイパーテキストモデルがあり [18, 24]，実際に開発されたシステムとして Intermedia[35, 36]，HyperCard[29]，Guide[26] などがある．さらに，映像や音声など時系列情報を扱うためのユーザインタフェースモデル [25] も提案されている．

Intermedia は，米国の Brown 大学において開発されたマルチメディア対応のハイパーテキストシステムであり，Macintosh 上で稼働している．このシステムで扱うことのできるメディアはテキスト，グラフィック，映像であり，これらのマルチメディア情報をリンクでつなぐことができる．Intermedia でのリンクは任意の情報の

任意の場所に生成することができ，一度リンクが生成されると永続的に存在する．このリンクは，双方向リンクであるので，参照されている情報から参照している情報を逆に辿ることができる．リンクは，マルチメディア情報の一部として扱われるのではなく，web と呼ばれるマルチメディア情報間の関連情報を扱う部分に保存される．これによりデータの独立性を保っている．また，Intermedia では，マルチメディア情報を作成・編集するためのアプリケーションも用意されており，これらのアプリケーションとしては，テキスト・エディタ InterText，グラフィックス・エディタ InterDraw，スキャナから画像を取り込む InterPix などがある．

HyperCard は，Macintosh 上で提供されるハイパーテキストシステムである．HyperCard では，カードと呼ばれる情報の基本単位を組み合わせることでひとまとまりの情報を構成する．カードには，文字列やイメージを表示させることができ，さらに，利用者が対話的に操作を行なうことができるボタンをおくことも可能である．ボタンでは，スクリプトを記述することでさまざまな動作を定義することができる．

Guide のデータモデルは，複数の guideline と呼ばれるファイルから構成され，ファイルの内部には複数の structure(構造) が存在する．structure の保持する内容は，テキスト，Paint グラフィック，PICT グラフィックのいずれかである．リンクはこの structure 同士に形成することができ，4 種類ある．Button/Replacement リンクは，同じファイル内の structure 間に形成されるもので，ボタンを押すとリンク先のデータが表示される．Note/Definition リンクも同じファイル内の structure 間に形成されるが，これはボタンを押し続けている間だけリンク先の情報が表示される．Reference/Reference Point リンクは異なるファイルの structure 間でも形成できる．Command/Definition リンクは，情報と Genesis という言語のコマンドのリンクを表している．

しかし，従来より開発されてきたハイパーテキストシステムやハイパーメディアシステムといわれるマルチメディアシステムでは，前節で挙げた 5 つの要件を必ずしもすべて実現していない．例えば，上記の Intermedia，HyperCard，Guide では，映像や音声などの時系列をもったメディアを扱うことはできるものの，時系列メディアの情報は主にリンクの宛先として参照され，それにともなって再生されるに過ぎず，映像内部に現れる人物をリンクの始点とする機能が提供されていない．さらに，再生に関しては情報間の同期をさせる機能が考慮されていない．

また，HyperCard の場合，XCMD や XFCN といった外部プログラムを実行するための機能を用いて，HyperCard の外部に存在する新たなメディアを扱うアプリケーションを使用することができるが，システム自身を直接拡張することはできず，その新たなメディアをリンクの始点として用いることは不可能である．情報間の関連を表す場合，ボタンにスクリプトを記述することで実現されるため，利用者が簡単な操作でリンクを付与するといったことができない．さらに，Intermedia では，統一されたユーザインタフェースで情報を扱うことが可能であるが，情報の管理をメディアの種類ごとに行なっているため，異なるメディアに対する情報の検索等を統一的に行なうことができない．

以上のように，従来のマルチメディアシステムでは，1.1で述べた 5 要件を必ずしもすべて実現していない．そこで，本研究では，このような既存のマルチメディアシステムにおける問題点を考慮しながら，1.1の 5 項目の要件を満たすようなマルチメディアシステムを実現するためのマルチメディアドキュメントのモデル化とマルチメディアシステムのアーキテクチャを提案する．特に，マルチメディアドキュメントのモデル化においては，時系列メディアの情報を柔軟に扱うために，

- 時系列メディア間の時間的関係
- 時系列メディアの振舞いに対応した情報間の関連

を充分にしかも簡潔に表現できるように考慮した．また，マルチメディアシステムのためのアーキテクチャとして

- システムの拡張性
- 操作の簡潔性

を充分に考慮したシステム構築を目指した．

第 2 章

マルチメディア情報のドキュメントモデル

本章では，前章で挙げた高度なマルチメディアシステムを構築するうえで重要な5要件をふまえた上で，マルチメディア情報に適したドキュメントモデルを提案する．

2.1 モデル化における問題点

マルチメディア情報のモデル化に着目した場合，1.1で挙げた5要件は，以下の3つにまとめることができる．

(1) 異なる種類のマルチメディア情報を統一的に扱う．
これは，要件1,4に対応するもので，マルチメディア情報をそのままの形で利用者に提供した場合，メディアの種類によってデータ構造や操作方法が大きく異なるため，利用者は個々のメディア扱うための知識を必要とし，複雑な操作を行なわなければならない．

(2) 情報間の関連を表現する．
この場合は，要件3に対応するものであり，さまざまな種類のマルチメディア情報を組み合わせて，一つのドキュメントを形成するためには，これらの情報間の関連を簡潔に表すことが必要である．

(3) 部分的な情報を扱う．
これは，要件2に対応しており，映像などの時系列情報では，内容が時間とともに変化するため，高度なマルチメディアドキュメントを形成するには，部分的に関連を表すことが必要となる．

(1) に対処して，テキストや図形情報に加えて映像やアニメーション等の時系列メディアを含めた種々の情報を統一的に扱うために，オブジェクト指向の概念 [23] を導入する．これにより，マルチメディア情報を一様なオブジェクトとして扱うことができる．また，時系列メディアの情報の再生や停止といった振舞いをオブジェクトのメソッドとして捉えカプセル化することができる．さらに，(3) に対応するため，テキスト情報内の単語や映像情報に現れる人物等の情報の一部分をもオブジェクトとして扱えることを可能にする．(2) でマルチメディア情報を組み合わせて，一つのマルチメディアドキュメントを表現するために，ハイパーテキストモデル [29] を採り入れ，マルチメディア情報を表すオブジェクトをハイパーテキストモデルのノードとして捉え，リンクを付与することでマルチメディアドキュメントを構成する．このような方法によって，1.1の要件 1,2 に対処している．また，オブジェクトという単一の枠組で扱うことは，要件 4 を実現するための基盤となる．

映像や音声といった時系列メディアを扱う場合，情報間の表示順序や同期といった時間関係を表現する必要がある (要件 3)．このため，従来のハイパーテキストモデルのリンクの概念を拡張する．拡張したリンクでは，始点のオブジェクトから宛先のオブジェクトに対して，情報の表示開始や終了を行なうメソッドを起動するために送られるメッセージとそのメッセージ送信の開始時間を記述する．また，情報間の同期を実現するために，一つの始点のオブジェクトから同時に複数のリンクをたどることを可能する．

以下の節では，これらの対応に基づいたマルチメディア情報を扱うためのドキュメントモデルであるハイパーオブジェクトモデルの概念について詳しく述べる．

2.2 ハイパーオブジェクトモデル

本研究では，1.1の要件 1,2,3,4 を実現するために，ハイパーテキストモデルとオブジェクト指向の概念を組み合わせた“ハイパーオブジェクトモデル”を提案している (図 2.1)．

ハイパーオブジェクトモデルでは，マルチメディア情報を表すオブジェクト群にリンクを付与することで情報間の関連を表現し，ハイパーメディアドキュメントを構成することができる．このハイパーメディアドキュメントを“シナリオ”と呼ぶ．

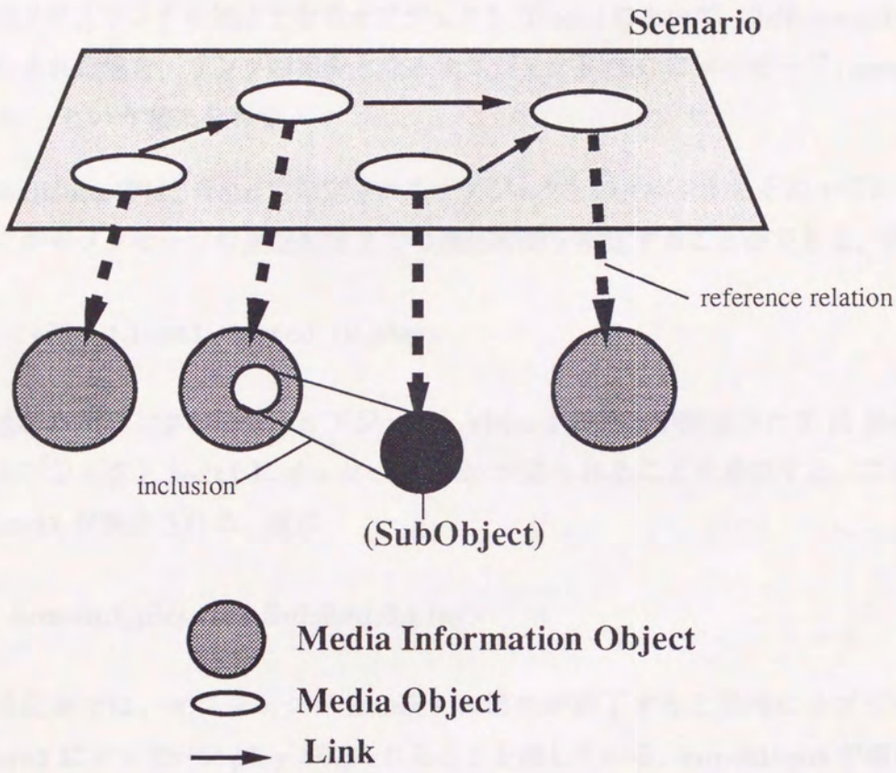

図 2.1: ハイパーオブジェクトモデル

以下に，ハイパーオブジェクトモデルの各要素の特徴について述べる．

2.2.1 リンク

ハイパーオブジェクトモデルにおけるリンクの記述は，

<from,to,conditions,message>

で表現され,「リンクの始点となるオブジェクト (from) において，条件 (conditions) が満たされた場合，リンクの宛先となるオブジェクト (to) にメッセージ (message) を送る.」という意味をもつ．

conditions では，from で指定されるオブジェクトのイベントとそのイベントが発生してからメッセージの送信開始までの遅延時間を指定することができる．例えば，

<video1,text1,started:10,play>

と記述されたリンクの場合，オブジェクト video1 の再生が開始されて 10 秒経過するとオブジェクト text1 にメッセージ play が送られることを意味する．これにより，text1 が表示される．次に，

<music1,picture1,finished:0,play>

という記述では，オブジェクト music1 の再生が終了すると同時にオブジェクト picture1 にメッセージ play が送られることを表している．conditions が満たされるリンクが複数ある場合，それらのリンクは同時にたどられることを意味する．

リンクをこのようにモデル化することにより，要件 3 を実現している．

2.2.2 メディアインフォメーションオブジェクト

メディアインフォメーションオブジェクトは，マルチメディア情報そのものを表しており，要件 1 を実現している．メディアインフォメーションオブジェクトは，後述の複数のメディアオブジェクトから参照することが可能であり，複数の利用者がマルチメディア情報を共有することができる．

2.2.3 メディアオブジェクト

メディアオブジェクトは，ハイパーテキストモデルのノードとしての役割を果たし，ノード情報としてメディアインフォメーションオブジェクトを図2.1のように参照する．メディアオブジェクトを以下に述べるように実現することにより，要件4に対処するための基盤となる．

メディアオブジェクトは，外部からメッセージを受けとり，それにしたがって，情報の表示開始や終了を行ない，表示の開始や終了といった状態変化を知らせるイベントを発生する．ハイパーオブジェクトモデルにおいて，ハイパーテキストモデルで任意のオブジェクトどうしを関連づけることができるように，メディアオブジェクトでは，リンクによりメッセージ交換の情報が与えられる．さらに，メディアオブジェクトでは，メッセージなどのインタフェースを限定することで，任意のオブジェクト間にリンクを容易に付与することを可能にしている．

メッセージの伝達

ノードであるメディアオブジェクトは，動的にメッセージの宛先を変更することを可能にするため，直接特定のオブジェクトにメッセージを送るのではなく“target”と呼ばれる仮想的なオブジェクトにメッセージを送るように記述する．実際にリンクがたどられた場合，この仮想オブジェクト target は，リンクによって指定されているメディアオブジェクトに置き換えられ，そのオブジェクトにメッセージが送られる．

インタフェースの共通化

メディアオブジェクトでは，受理可能なメッセージ (表2.1) と発生するイベントの種類 (表2.2) を限定している．このようにメディアオブジェクトのインタフェースを共通にすることにより，任意のオブジェクト間にリンクを容易に付与することができる．

表 2.1: メッセージの種類

メッセージ	動作
play	オブジェクトの情報の表示を開始する.
stop	オブジェクトの情報の表示を終了する.
pause	オブジェクトの情報の表示を一時停止する.
continue	一時停止されたオブジェクトの情報の表示を再開する.

表 2.2: イベントの種類

イベント	意味
started	オブジェクトの情報の表示が開始されたことを表す.
finished	オブジェクトの情報の表示が終了したことを表す.
selected	オブジェクトが利用者に選択されたことを表す.

2.2.4 サブオブジェクト

従来のマルチメディア／ハイパーメディアシステムでテキスト中のある単語や図形情報の一部分に対して，他のマルチメディア情報をリンクによって関連づけることが可能である．ハイパーオブジェクトモデルでは，この機能を映像やアニメーションにも適用し，映像内部に現れる人物や物体，アニメーションの個々のキャラクタに対しても，リンクの始点や宛先として扱うことができるように，これらの部分的な情報をもオブジェクトとして捉えることを可能にしている．ハイパーオブジェクトモデルでは，このような他のオブジェクトの内部に存在するオブジェクトを“サブオブジェクト”と呼ぶ．サブオブジェクトは，図2.1のように元のメディアインフォメーションオブジェクトの一部を構成する情報のみからなる別のメディアインフォメーションオブジェクトとして実現されており，このため通常のメディアインフォメーションオブジェクト同様にシナリオにおけるノードとなる．これにより，サブオブジェクトがクリックされるとリンクをたどるといったハイパーメディアドキュメントを構築できる．

サブオブジェクトは，元のメディアインフォメーションオブジェクトが消去されると自動的に消去される．サブオブジェクト自身がサブオブジェクトを含むことも可能である．

これにより，1.1で挙げた要件2の部分情報の一般化をメディアの種類に関係なく対処することができる．

2.2.5 シナリオ

シナリオは，リンクとリンク内に指定されるメディアオブジェクトの集合から構成されるマルチメディアドキュメントである．ハイパーオブジェクトモデルでは，シナリオをマルチメディア情報から独立させることにより，複数の利用者がマルチメディア情報を共有しながら独自のハイパーメディアドキュメントを構成することが可能である．

2.3 評価

本節では，ハイパーオブジェクトモデルと既存のハイパーメディアシステムのドキュメントモデルと比較し，1.1で述べた5要件の達成度や有効性等について論じる．

2.3.1 情報の種類と粒度

Intermedia[35, 36] や HyperCard[26] といったハイパーメディアシステムにおいても，映像や音声などの時系列メディアを扱え，1.1の要件1を満たしているが，これらのシステムでは，時系列メディアの情報が参照された場合，単純にその情報を再生するのみである．

ハイパーオブジェクトモデルでは，2.2.4に述べたサブオブジェクトの概念をすべてのメディアに適用し，映像情報やアニメーション情報の一部分をも一つの独立した情報の単位として扱う機構を提供している．このように1.1の要件2を実現することで，図形情報やテキストと同様に映像やアニメーションなどの時系列情報をリンクの宛先だけでなく始点としても設定することができ，より細かな情報の組み合わせが可能となる．

2.3.2 リンクの意味記述と時間記述

従来のハイパーメディアシステムでは，リンクは情報間の参照関係のみを表すものである．映像や音楽といった時系列メディアを扱う場合，複数の情報を組み合わせて効果的に表示することが望まれる．しかし，Intermedia のような単純なリンクでは，情報間の時間関係を表現することが不可能である．また，再生の一時停止や途中終了といった情報の動作を指定することもできない．

HyperCard では，HyperTalk と呼ばれるスクリプト記述言語が提供されており，これを利用することでさまざまな動作を記述することが可能であるが，HyperTalk の場合，言語仕様が複雑であり，利用者はその記述を直接行なわなければならないため，使いこなすのが容易ではない．

ハイパーオブジェクトモデルにおけるリンクは，2.2.1で述べたように，リンクをたどる時の条件やメッセージを指定することでリンクに意味を与えており，上述のよ

うな時系列情報に特有の動作を特別なコマンドやスクリプトを用いることなく，リンクにより容易に指定することができる．これにより，ハイパーオブジェクトモデルでは，リンクという単一の概念でオブジェクトどうしの単純な参照関係以外にさまざまな関係を表すことが可能である．また，リンクの記述の簡潔性を実現するために，2.2.3で述べたようにノードとなるメディアオブジェクトのインタフェースを限定している．

現在, ハイパーメディアに関しては, HyTime, MHEG[27] の二つの標準化作業が行なわれているが, その中でもハイパーオブジェクトモデルと同様に複数のメディア間の同期等が考えられている. 特に, MHEG においては, メディア間の同期を (1) 連接同期, (2) 並行同期, (3) 基本同期, (4) 条件付同期, (5) 循環同期, (6) 連鎖同期の 6 つに分けて考えている.

ハイパーオブジェクトモデルにおいては, (1)〜(3) までを, リンクの **condition** フィールドを拡張することで実現している. (4) も, **condition** フィールドを拡張することで対処でき, (5), (6) に関してはオブジェクトのグループ化などの概念を導入することで実現できると考えている. ハイパーオブジェクトモデルでは, 1.1の要件 3 を含むこれらすべての概念をリンクとオブジェクトのみで表せるため, 記述の統一性, 簡潔性が得られる. これにより，オブジェクト間の関係記述においても，1.1の要件 4 を実現している.

第 3 章

分散型マルチメディアシステムのアーキテクチャ

本章では，前章で述べたハイパーオブジェクトモデルをもとに，1.1で挙げた5要件を満足するようなシステムのアーキテクチャを提案する．本研究では，ハイパーオブジェクトモデルと本章で提案するアーキテクチャの有効性を示すために，分散型マルチメディアシステムのアプリケーションとしてプレゼンテーションシステム“Harmony”を実現している．

3.1 システム構築の基本方針

マルチメディアシステムを構築に関して，1.1の5要件を分析すると以下のような項目にまとめることができる．

(1) 操作の統一性および簡潔性

これは，要件4に対応するもので，ハイパーオブジェクトモデルでは，データ構造や操作方法が異なるマルチメディア情報を一様なオブジェクトとして扱っている．ここでは，ハイパーオブジェクトモデルの利点を損なわずにシステム構築することが望まれる．

(2) 拡張性の提供

要件5より，マルチメディアシステムにおいて，新たなメディアを加える場合に，システムを大幅に変更することなく容易に新しいメディアを扱えるように

拡張できなければならない．

さらに，複数の利用者がマルチメディアシステムを用いる場合には，以下のようなことが必要になる．

(3) マルチメディア情報の共有
CAI システムや会議システムといったアプリケーションでは，複数の利用者で情報を共有しながら，個々の利用者ごとに説明のテキストなどを付け加えていくことが望まれる．

(3) のマルチメディア情報の共有を実現するために，単一のデータベースですべての情報を管理することが考えられる．このようにすれば，メディアの種類に関わらず，すべて情報に対する検索等の操作を一括しておこなうことが可能となる．しかし，個々のマルチメディア情報は，互いにデータ構造が大きく異なるため，構造の柔軟性に乏しい関係データベースのような従来のデータベースを用いて，マルチメディア情報を一括して扱うことは，(1) の操作の統一性を実現することが困難であり，好ましくない．一方，近年，研究開発が著しいオブジェクト指向データベース [16] では，複雑な構造をもつデータを扱うことができ，異なるデータ構造をもつ複数のマルチメディア情報に対応したオブジェクトの一様な管理を可能にすることができる．また，オブジェクト指向データベースのデータ構造の変更や追加に対する柔軟性を利用することで，(2) の拡張性を実現するための基盤となる．さらに，オブジェクトの表示や作成・編集といったメソッドを扱うプロセスをメディアの種類ごとに用意し，このようなデータベースを中心としたサーバ・クライアント型のマルチプロセスシステムとして構築することで，新たなメディアを追加する場合，対応するプロセスのみを構築するだけでシステムを拡張できる．

(3) において，マルチメディアドキュメントの素材となるマルチメディア情報を共有しながら，個々の利用者ごとに異なったドキュメントを構成するためには，マルチメディア情報を表すオブジェクトとそれらの関係を表現するリンクを別々に管理する必要がある．

(1) の操作の統一性・簡潔性を実現するには，ユーザインタフェースにおいてハイパーオブジェクトモデルの利点を損なわず，簡潔な操作環境を提供しなければならない．このような基本方針によって，1.1の要件 4,5 に対応することができる．

以上のことを考慮に入れるとマルチメディアシステムには次のような3つの機能要素が必要となる[29](図3.1).

- マルチメディア情報の格納・管理を行なうためのデータベース機能
- マルチメディア情報間の関連を扱うリンク管理機能
- これら2つのシステムを統一した環境で扱うことのできるユーザインタフェース機能

次の節では，本節の基本方針に基づいたマルチメディアシステムの一例として構築しているプレゼンテーションシステム "Harmony" のアーキテクチャについて詳しく述べる.

3.2 Harmony のシステムアーキテクチャ

マルチメディアプレゼンテーションシステム Harmony では，マルチメディアシステムに必要なデータベースシステム，リンク管理システム，ユーザインタフェースシステムをそれぞれ，**Harmony/DB**，**Harmony/LM**，**Harmony/UI** と呼んでいる．以下に，Harmony の実システムの核となるこれらの要素においてハイパーオブジェクトモデルを有効に扱うための実現方法について詳しく述べる.

3.2.1 オブジェクト指向データベース Harmony/DB

Harmony/DB は，オブジェクトの登録・削除というようなストレージ管理機構と，オブジェクトの活性化やオブジェクト間のメッセージ交換等のオブジェクト実行機構を実現している.

オブジェクトの定義

データ構造やそれを扱うメソッドは，メディアの種類によって異なるため，映像やアニメーションといったメディアの種類をクラスとして捉えている．データ構造やメソッドといったクラス情報の定義は，Smalltalk-80 と同様のシンタックスを用いて行なう (3.5参照).

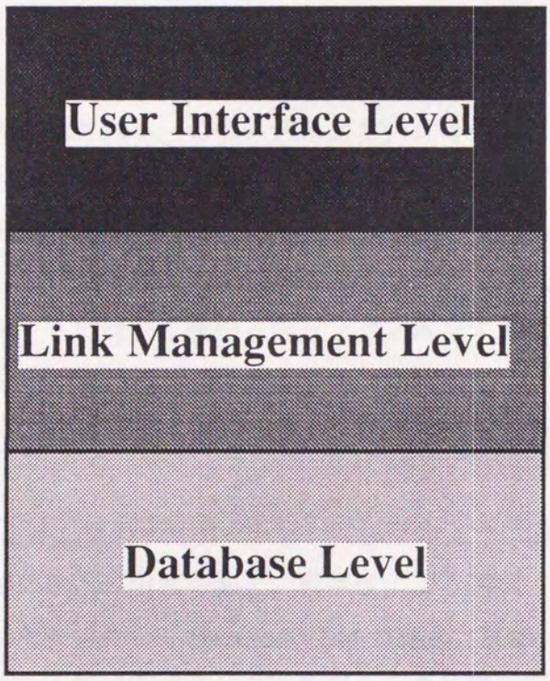

図 3.1: マルチメディアシステムの基本構成

3.2.2 リンクマネージャ Harmony/LM

Harmony/LM は，メディアインフォメーションオブジェクト間のリンクの生成・削除・ナビゲーションに関する機能を実現している．

以下では，リンクの生成とメッセージ伝送によるリンクのナビゲーションについて述べる．

リンクの生成

利用者が Harmony/UI を介して，メディアインフォメーションオブジェクト間にリンクを付与する操作を行なうことにより，シナリオにメディアオブジェクトとリンクオブジェクトが追加される．このメディアオブジェクトは，対応するメディアインフォメーションオブジェクトを参照している．リンクの記述は，始点となるメディアオブジェクト内に追加される．メディアインフォメーションオブジェクトに直接リンクを保持させないことにより，複数のシナリオで共通したメディアインフォメーションオブジェクトに対して異なったリンクを付与することができる．

リンクのナビゲーション

Harmony におけるリンクのナビゲーションは以下のように実現される．メディアオブジェクトで情報の表示の開始や終了，利用者による選択といったイベントが起こると，それをメッセージとして Harmony/LM に送る．Harmony/LM では到着すると即座にそのイベントに対応するリンクを探し，あればリンクの記述で指定されている宛先のオブジェクトを仮想オブジェクト **target** に設定し，リンクで指定されているメッセージを送る．これによってリンクのナビゲーションが行なわれたことになる．

3.2.3 ユーザインタフェース Harmony/UI

Harmony/DB，Harmony/LM 上でのオブジェクトやリンクの生成・削除といった操作は，Harmony/UI を介して行なう．Harmony/UI では，1.1の要件 4 を満たすために，統一した操作環境下ですべてのオブジェクトを容易に扱うことができる．例

えば，映像を表示する場合，データベースより望むオブジェクトを検索し，取り出したオブジェクトに対して play メッセージを送るだけでよい．

3.3 マルチメディアドキュメントの記述

ハイパーオブジェクトモデルの一つのシナリオは，マルチメディアドキュメントの素材となるオブジェクトを組み合わせて，それらの間にリンクを付与することによって構成される．

3.3.1 関連するドキュメント記述モデル

既存のマルチメディアシステム，特に表示に重点がおかれるプレゼンテーションシステムにおけるドキュメント記述モデルは大きく二つに分けられる．一つはハイパーテキストのようなリンクによるモデルであり，もう一つはキーフレームに基づくモデルである．

ハイパーテキスト型の記述として，Macintosh の HyperCard を用いる方法を考えてみる．この場合，ボタンなどのオブジェクトのメソッドとして，シナリオの進行を記述することになる．メソッドに自由な記述が許されるため，利用者と対話的にシナリオ進行の展開を変えることができ，オブジェクト毎の時間的な制御も可能である．しかし，汎用性が高過ぎるため利用者の記述する部分が多く，シナリオ進行はそれぞれのオブジェクトの中のメソッドとして記述することになり，全体の流れが掴みにくい．

ハイパーオブジェクトモデルでは，映像や音声といった時系列をもった情報を扱う．時系列を持ったマルチメディア情報のプレゼンテーションシステムの一例として，アニメーションを主体としたものでは，Macro Mind 社の Director[41] がある．Director では，キーフレームアニメーションの考え方を採り入れ，各フレームにおけるキャスト（登場する図系，人物など）の動きを指定することにより，プレゼンテーションを記述する．しかし，この方法では，シナリオの進行が連続的に変わる時は良いが，利用者と対話的にシナリオ進行の展開を決める場合や，各キャストの条件によって展開を変える場合などの記述が行ないにくい．

3.3.2 シナリオの記述

Harmony ではハイパーテキスト型ドキュメントモデルを用いているが，全体の流れを掴み易いグラフィカルな表現を用いることができるように，Harmony におけるドキュメント記述は，二つのレベルに分けられている．各オブジェクトのリンクを直接指定するレベル（リンクレベル）とオブジェクトを表示する時間的な順序のみを指定するレベル（シナリオレベル）である．シナリオレベルはいわばリンクレベルに対するマクロ記述に相当する．

リンクレベルでは，ハイパーオブジェクトモデルのリンクの記述を直接用いてドキュメントを記述する．リンクレベルの記述はある意味で非常に強力なアセンブラのようなもので，あらゆる記述が行なえるが，実際にアプリケーションを構築する場合にすべてを指定すると利用者の操作が繁雑であるという問題がある．そのため，ある程度自由度を犠牲にしても，簡単に記述できるレベルが必要となる．

Harmony では，利用者に提供するメディアオブジェクトのイベントやメッセージの組み合わせ方を限定したインタフェース (シナリオレベル) を提供している．シナリオレベルの記述は，リンクレベルの記述に対するいわばマクロに相当する．シナリオレベルの記述は，相当するリンクレベルの記述に展開される．また，利用者がシナリオ全体の構成を容易に把握することができるように，シナリオを図 3.2のようなグラフ構造で表すことができる．

シナリオレベルでは，利用者はマルチメディア情報の時間的な順序のみを記述し，イベントやメッセージの記述はほとんど自動的に行なわれる．シナリオレベルにおける記述として，逐次進行，並列進行，選択進行の 3 種類が用意されている．これらの進行を組合わせることで容易にマルチメディアドキュメントの記述が行なえる．進行はメディアオブジェクト間のリンクとして表される．以下にこれらの進行について述べる．進行をグラフ化した図では，長方形はメディアオブジェクトを表し，実線の矢印はリンクを表す．

逐次進行 (serial)

映像や音声などの時系列情報は，ある時間が経つと情報の表示や再生が終了する．「逐次進行」とは，あるメディアオブジェクトの表示が終了した時に，利用者の操作とは関係なく自動的にリンクを辿り，次のメディアオブジェクトを表示する進行で

ある．

逐次進行の例として，メディアオブジェクト A が終了すると，続いてメディアオブジェクト B の表示を開始する場合を図 3.3に示す．対応するリンクレベルの記述は以下のようになる．

< A, B, finished, play >

並列進行 (parallel)

「並列進行」とは，複数のメディアオブジェクトの表示開始を同時に行なう進行である．並列進行は，図 3.4のように表される．ここでは，すでにメディアオブジェクト A とメディアオブジェクト B が逐次進行で関連づけられており，メディアオブジェクト B に対して並列進行するものとしてメディアオブジェクト C を指定している．このとき，次のようなリンクが記述される．

< A, B, finished, play >
< A, C, finished, play >

分岐進行 (branch)

マルチメディアドキュメントを表示する場合，利用者の対話的操作により選択的に表示を進めていく場合がある．選択的な進行を実現するために，提供されているのが「分岐進行」である．分岐進行を用いると，テキスト中のある単語や映像に現れてくる人物などを選ぶことによってシナリオが分岐して進行していくような記述を指定することができる．分岐進行は，常にサブオブジェクトによって引き起こされる．図 3.5に分岐進行の例を示す．メディアオブジェクト B は，メディアオブジェクト A のサブオブジェクトである．ここで，メディアオブジェクト B を選択するとメディアオブジェクト C が表示される．この場合は，以下のようなリンクが生成される．

< B, C, selected, play >

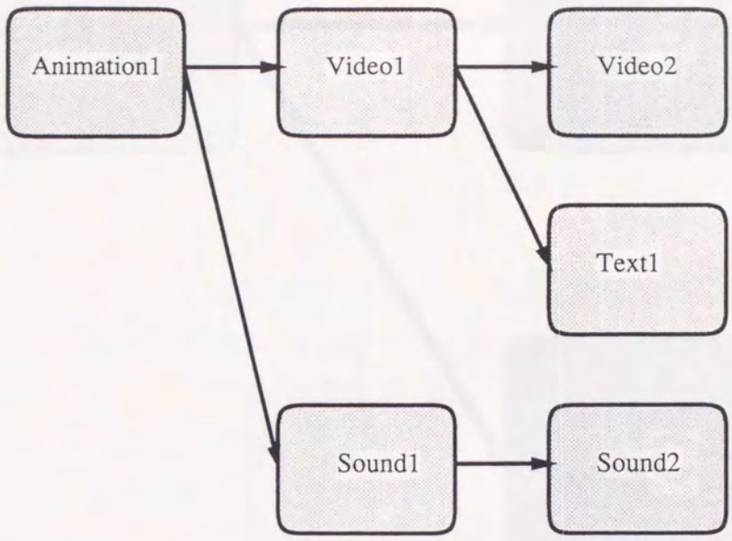

図 3.2: シナリオのグラフィカルな表現

図 3.3: 逐次進行

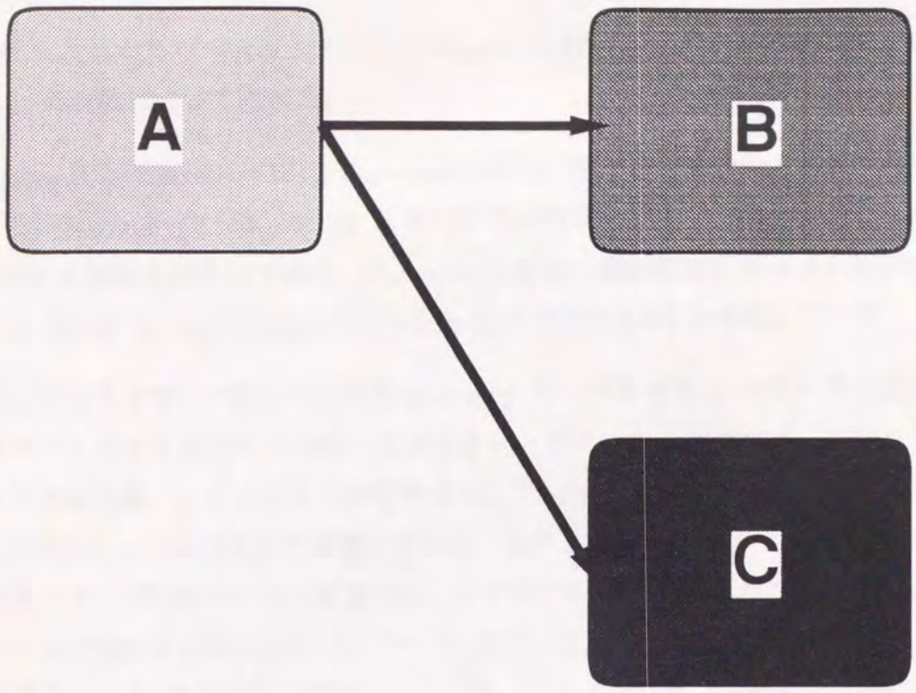

図 3.4: 並列進行

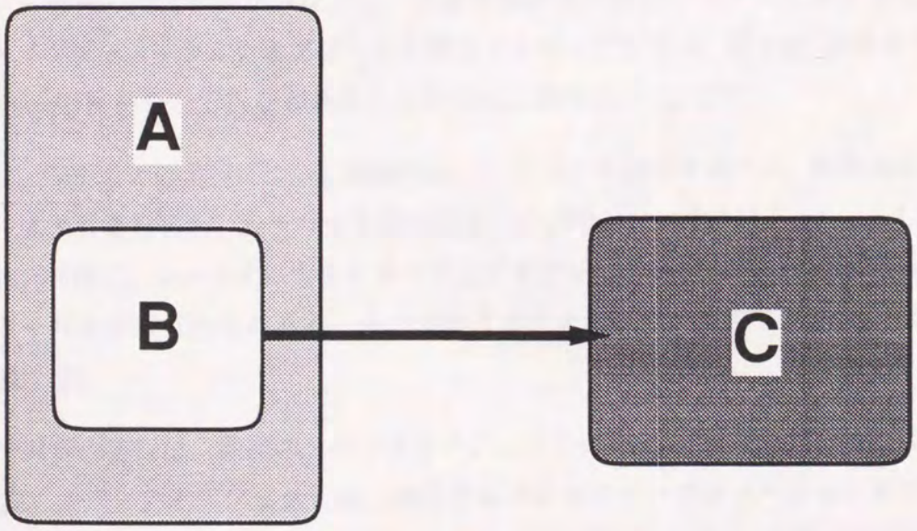

図 3.5: 分岐進行

3.4 システムの実装

本節では，UNIX ワークステーションが LAN に接続されている分散環境上での Harmony の実装について述べる．

Harmony/UI，Harmony/LM，Harmony/DB は，まとめて単一のプロセスとして構築する．Harmony/UI は，ウィンドウシステムのツールキットライブラリである ET++[38] の機能を利用しており，Harmony/DB は，商用のオブジェクト指向データベースである Versant Object Technology 社の VERSANT を使用している．

このプロセスをサーバとして (以降 Harmony サーバと呼ぶ)，メディアの種類ごとにオブジェクトを表示するためのクライアントプロセスを用意する．このクライアントプロセスは，クラスプロセスと呼ばれ，ET++および X ウィンドウシステム [31] のアプリケーションとして実現している．メディアインフォメーションオブジェクトのメッセージ交換やメソッド実行は，クラスプロセスで行なわれ，Harmony では，これらの機能を実現するライブラリ (3.7参照) を提供しており，我々が作成したビデオ編集ツール [8] や汎用の作図ツール [13]，アニメーションツール [12] 等に，このライブラリを組み込むことでクラスプロセスを実現している．このようにシステム構築することにより，1.1で挙げた要件 5 に対応することができる．

Harmony は，Harmony サーバとクラスプロセスがプロセス間通信の機能を用いて互いに情報の授受を行ないながら，処理を進める分散型システムであり [14]，メディアインフォメーションオブジェクト間のメッセージ交換は，図 3.6に示すように，Harmony/DB を介して行なわれる．この時の手順を図 3.7に示す．

次に，シナリオの実行について説明する．シナリオを実行するには，利用者はデータベースより実行したいシナリオを取り出す．この時，シナリオは Harmony/LM に渡され，同時に，シナリオに含まれるメディアオブジェクトがすべて生成される．新しくシナリオが取り出されると，古いシナリオとそのメディアオブジェクトはすべて消滅する．

シナリオの実行は，最初のメディアオブジェクトに play メッセージを送ることで始まる．メディアオブジェクトは，対応するメディアインフォメーションオブジェクトにメッセージを転送し，オブジェクトの実行が開始される．メディアインフォメーションオブジェクトは，実行に関連したイベントを Harmony/LM に送る．

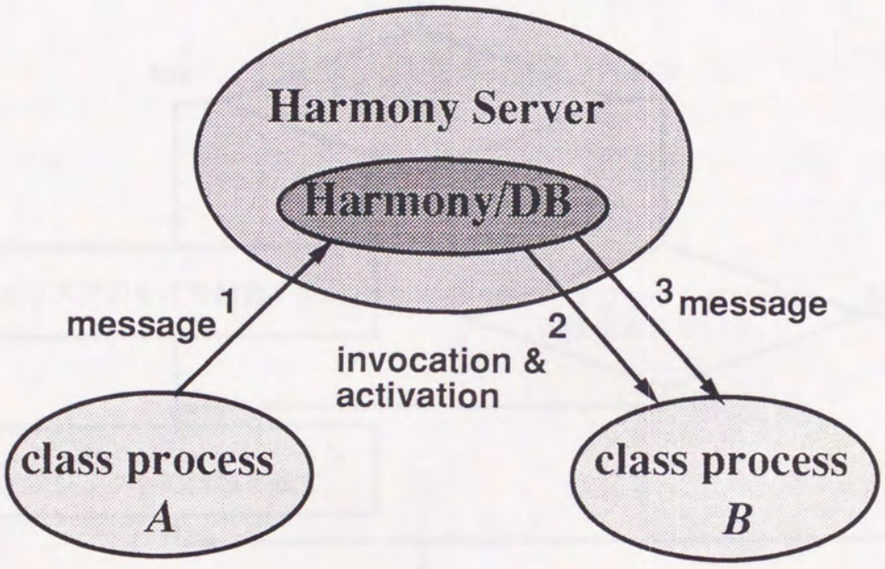

図 3.6: Harmony のメッセージ交換

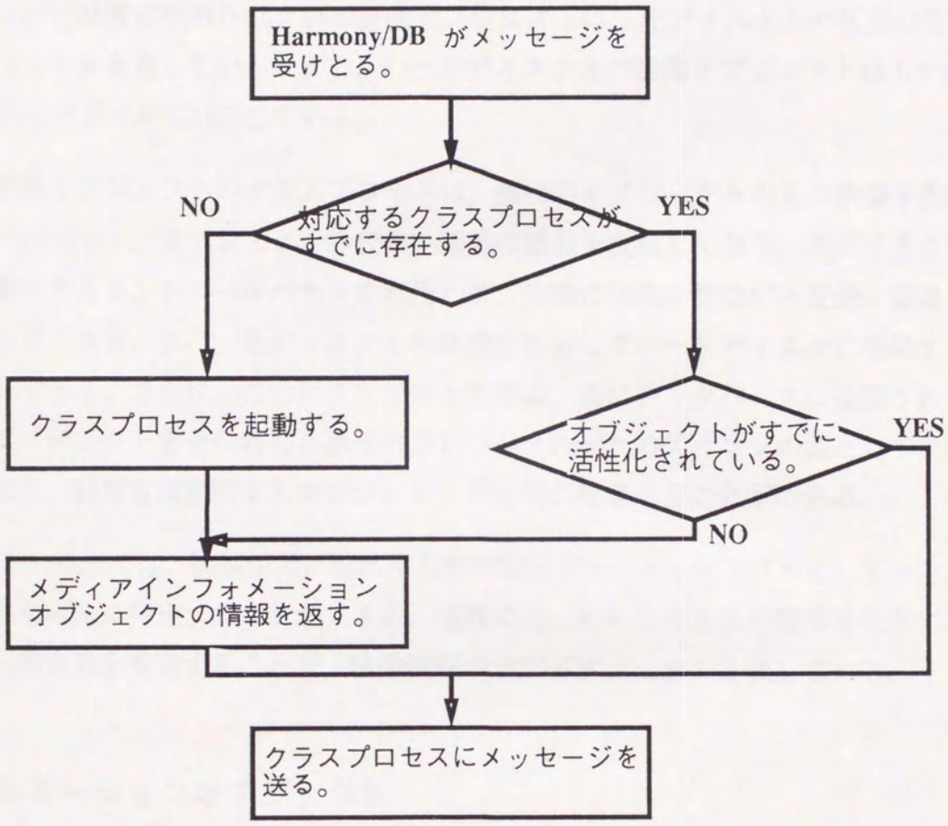

図 3.7: メッセージ交換の手順

図 3.8に Harmony の画面例を示す．以下では，映像，アニメーション，音声，図形，音楽，テキストそれぞれのオブジェクトの実現について述べる．

映像オブジェクト

映像は，追記型の光ディスク装置に収められるものと，JPEG 方式 [37] により圧縮した形式でワークステーションのハードディスク上に格納されるものがある．光ディスク装置に収められている映像オブジェクトは，光ディスク上の任意の連続したフレームを表している．また，ハードディスク上の映像オブジェクトは UNIX の一つのファイルに対応している．

映像オブジェクトのクラスプロセスは，個々のオブジェクトのもつ映像を表示するだけでなく，オブジェクトの登録・編集の機能を提供しており，光ディスク装置を操作するコントロールパネルを利用して，実際に映像を見ながら登録・編集することができる．また，光ディスク上の映像を圧縮してハードディスクに格納することもできる．さらに，このクラスプロセスでは，既にデータベースに登録されているオブジェクトをその最初と最後のフレームそれぞれのアイコンの組として表示しており，利用者は意図するオブジェクトを容易に指定することができる．

Harmony では，映像情報に現れる人物や物体をサブオブジェクトとしてリンクの始点や宛先に指定することができる．実際には，それらの人物や物体を包含するような長方形を指定することで，映像情報のサブオブジェクトを表している．

アニメーションオブジェクト

Harmony が利用しているアニメーションシステム RAS[12] は，実時間でのアニメーションを実現しており，アニメーションのストーリーを記述するための言語を提供している．個々のストーリーは，異なるファイルに記述される．一つのアニメーションオブジェクトは，このファイルに対応している．アニメーションに登場してくる個々のキャラクタは，アニメーションサブオブジェクトとして扱うことができる．

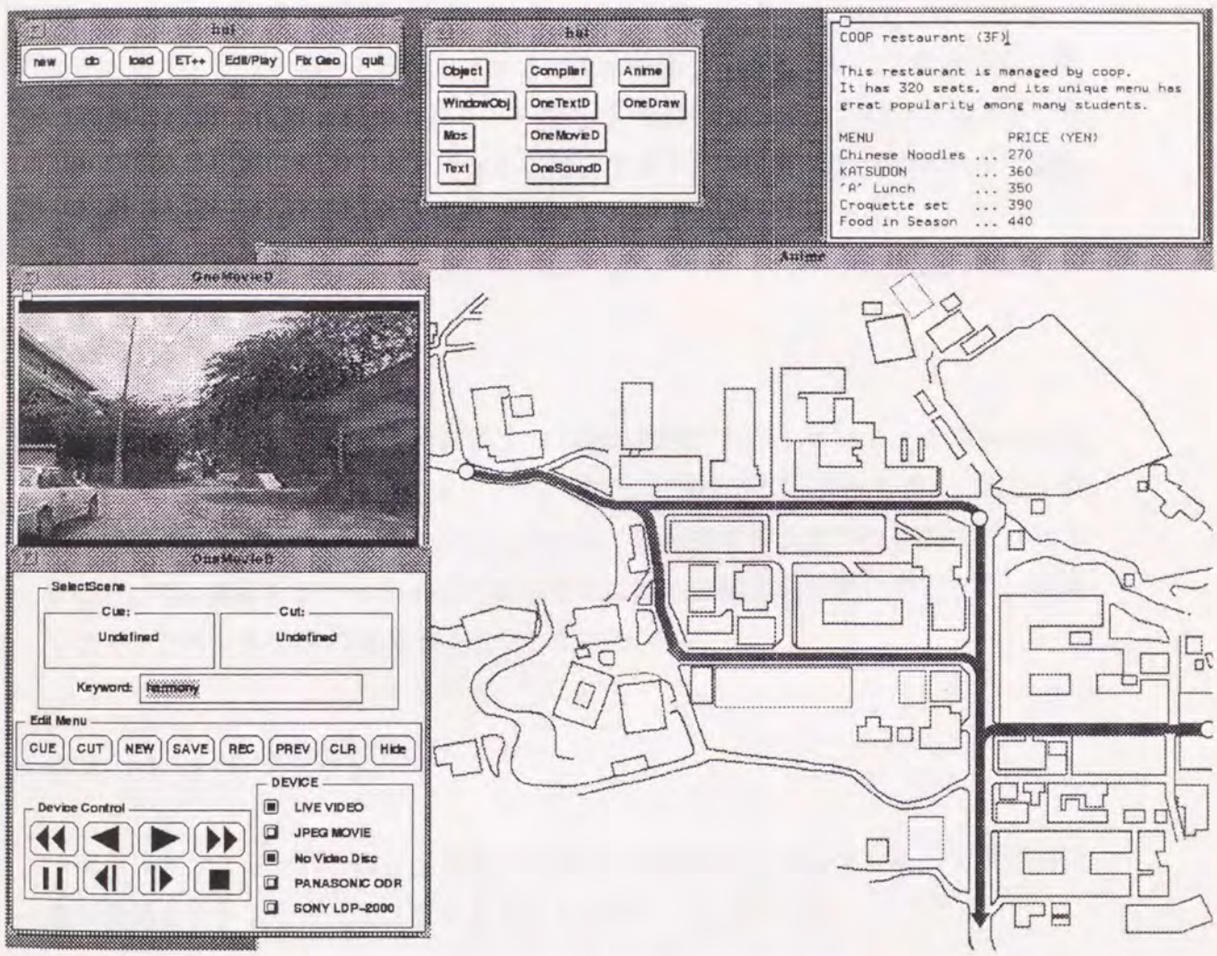

図 3.8: Harmony の画面例

音声オブジェクト

音声オブジェクトは，ワークステーションの音声機能を利用して再生される．一つの音声オブジェクトは，UNIX 上のファイルに対応している．

図形オブジェクト

図形オブジェクトを扱うクラスプロセスは，作図ツール Key3[13] のアプリケーションライブラリを用いて構成している．この作図ツールにおいて，作成された図形の情報は，ファイルに記述される．そのため，図形オブジェクトは，アニメーションオブジェクトと同様，このファイルに対応する．長方形や文字列等といった図形情報の個々の要素は，図形サブオブジェクトとして扱うことができる．

音楽オブジェクト

音楽オブジェクトは，楽曲を表現する MIDI 情報に対応しており，この MIDI 情報は，ファイルに記述されている．したがって，音楽オブジェクトもまた，MIDI 情報が格納されているファイルに対応している．音楽オブジェクトのクラスプロセスでは，MIDI 情報をシンセサイザで再生する以外に，編集を可能にするため，楽譜をウィンドウシステムの画面上に表示している．

テキストオブジェクト

テキストオブジェクトは，テキストの文字列そのものを表している．文字列の任意の範囲をテキストサブオブジェクトとして扱うことができる．

3.5 マルチメディア情報の管理

3.5.1 クラス定義

Harmony で扱われるオブジェクトのクラス定義は，そのクラスのデータ構造の定義とそのデータ構造を扱うためのメソッドの定義からなる．

クラスのデータ構造定義

クラスのデータ構造定義は，"Data Structure Definition"(以下 DSD と略す) と呼ばれるデータ構造定義言語を用いて定義される．DSD では，データ構造は 基本データ型 (primitive data type) と構成子 (constructor) を用いて定義することができる．Harmony で用いている基本データ型，構成子の種類をそれぞれ，表 3.1，表 3.2に示す．

DSD によって記述されたデータ構造は，コンパイラ (**Harmony compiler**) によって解釈され，データベースに登録される．また，同時に C 言語によるデータ定義ファイルが作成され，アプリケーションプログラムで利用することができる．

クラスのメソッド定義

メソッドの定義は，Smalltalk-80[23] の文法を用いて記述されるインタフェース部分と C 言語によって記述される実現部分からなる．Smalltalk-80 によるインタフェース定義は Harmony compiler によって解釈されることにより，オブジェクトのメソッド呼びだしが可能となる．インタフェース定義に対しても，C 言語によるメソッドインタフェースファイルが作成され，データ定義ファイルとともにメディアプロセスを構築する場合に用いられる．図 3.9に映像オブジェクトのクラス定義の例を示す．1 行目の `WindowObj subclass: Video` では，クラス名を `Video` としている．3 行目の `Video subobjectclass: #SubVideo!`では，映像オブジェクトのサブオブジェクトのクラス名を `SubVideo` としている．5 行目以降の `linkManager`，`changeMode`，`play`，`stop` は，映像オブジェクトのメソッドの定義を行なっている．例えば，`play` メッセージが送られてきた場合，実際には C 言語で記述された `HmPlay` という関数が呼ばれる．関数 `HmPlay` は，クラスプロセスを作成する時に実現しな

表 3.1: 基本データ型

int	整数 (4 バイト)
u_int	正整数 (4 バイト)
short	整数 (4 バイト)
float	実数 (4 バイト)
double	実数 (8 バイト)
char	文字
String	文字列
Voidp	任意の基本型，構成子へのポインタ

表 3.2: 構成子

tuple	任意の基本データ型より構成
List	1 種類の基本データ型のリスト
Array	1 種類の基本データ型の配列
Set	1 種類の基本データ型の集合

```
WindowObj subclass: Video
        category: 'Model of Video Object in ET++'!

Video subobjectclass: #SubVideo!
!Video methodsFor: 'system fundamental' temp:NIL!
linkManager
        <primitive: HmRegistManager>!
changeMode: EditMode
        <primitive: HmChangeMode>!
play
        <primitive: HmPlay>!
stop
        <primitive: HmStop>!!

Video parseDsd: 'int x
                int y
                int width
                int height
                int in
                int out
                String name
                Array of int subobjects'!
Video printDsd!
Video putCurDsdDB!
Video fileOutDsd!
!Video fileOut: #Video_init.c!
!Video dbOut!
```

図 3.9: クラス定義の例

ければならない．13 行目から 20 行目は映像オブジェクトのデータ構造を定義している．21 行目から 25 行目でデータベースにデータ構造を登録し，クラスプロセスの構築に必要なファイル (例では `Video_init.c`) の生成を指示している．

3.5.2 オブジェクト管理

Harmony/DB のデータベース・カーネルとして，商用のオブジェクト指向データベース VERSANT[40] を用いている．VERSANT では，オブジェクト識別子の機能が提供されているが，データベースのクライアントアプリケーション以外のアプリケーションにおいてオブジェクトの利用を考慮していないため，オブジェクト識別子を Harmony/DB で管理する必要がある．Harmony におけるオブジェクト識別子は，次のようなクラス識別子 ($class_id$) とインスタンス識別子 ($instance_id$) の二つからなる．

$$< class_id, instance_id >$$

クラス識別子は Harmony システム全体においてユニークな識別子であり，インスタンス識別子は，クラス内でユニークなものとなっている．つまり，クラス識別子により，オブジェクトのメディアの種類を特定することができる．通常クラス識別子，インスタンス識別子は正整数であるが，クラスそのものを表すオブジェクト識別子では，$instance_id = 0$ となる．

Harmony/DB のために表 3.3に示すようなシステムクラスを定義した．システムクラスはメディアオブジェクトのクラスやインスタンスを格納・管理するために用いられる（図 3.10参照)．

HClass はクラスに関する情報を格納するために用いられる．メディアプロセスが起動される場合，このクラスを検索し必要な情報を取り出し起動する．つまり HClass はメディアプロセスの定義を表している．HObj は，クラス識別子とインスタンス識別子の組とそれに対応する情報 (Bchunk) を保持する．harmonyDB では，クラス識別子とインスタンス識別子をキーとして扱っており，これらの識別子を与えると対応するオブジェクトを検索することができる．また，Harmony では，データベースのクライアントプロセス (Harmony/DB) とそれ以外のプロセス (クラスプロセス) の間でオブジェクトの受け渡しをプロセス間通信で行わなければならない．そのた

表 3.3: クラス構成

クラス名	機能
HObject	データベース内の全オブジェクトの基本抽象クラス.
Bchunk	バイトストリームを格納するために用いるクラス.
HClass	クラスの情報を保持するために用いるクラス.
HObj	クラス識別子とインスタンス識別子の組を持つ.
harmonyDB	データベース内の全てのオブジェクトとその対応するキーを持つ.

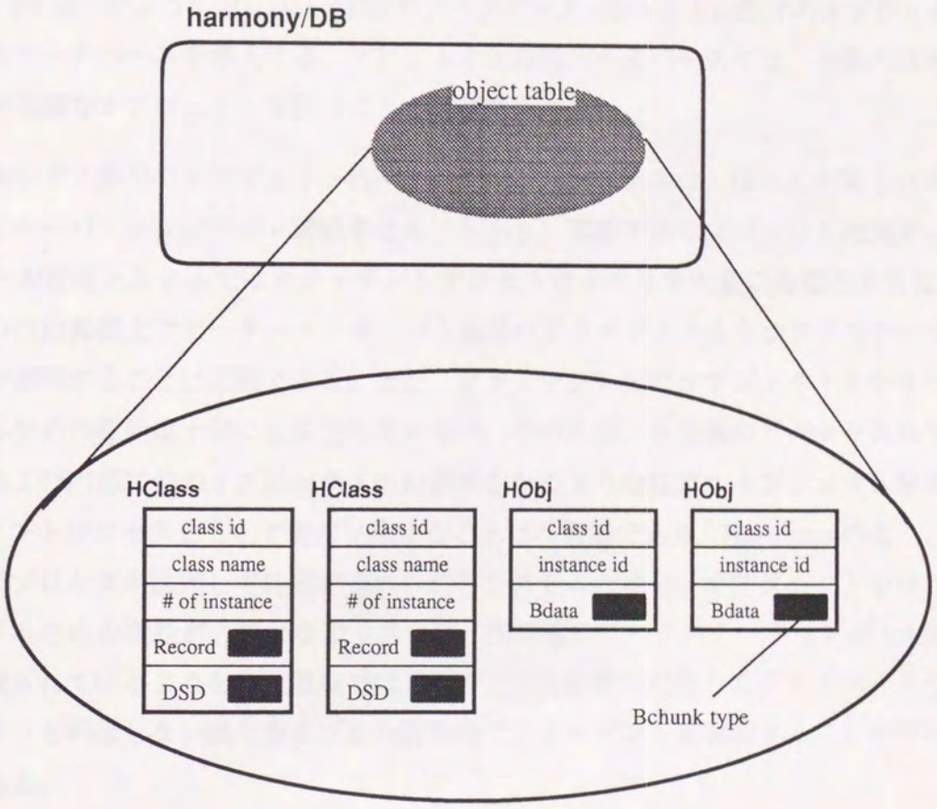

図 3.10: システムクラスの構成

め，データベースカーネル内ではオブジェクトはバイトストリームの形で格納されている．Bchunk はこのバイトストリームを扱うために用意されている．

3.6 分散型システムのオブジェクト指向データベースによる実現

3.2で述べたように Harmony/DB のデータベース・カーネルに既存のオブジェクト指向データベースを導入する．オブジェクト指向データベースでは，複数のスキーマや複雑なオブジェクトを扱うことが可能である．

現存する商用のオブジェクト指向データベースシステムは，ほとんど図 3.11のようにサーバ・クライアント構成をとる．しかし，現存するオブジェクト指向データベース管理システムではクライアントプロセスはメモリを大量に必要とするため，一つの計算機上でデータベースサーバと複数のクライアントをもつアプリケーションを実現することは困難である．また，クライアント間でオブジェクトをやりとりするための機能は十分に提供されていない．そのため，可変長のデータや入れ子構造および内部に他のオブジェクトへの参照をもつような複雑なオブジェクトをクライアントプロセスどうしで直接交換することは不可能であり，Harmony のように複数のプロセスが協調して処理が進められるシステムの場合，オブジェクトをやりとりするための機能が必要となる．さらに，異機種のワークステーションが LAN に接続されているような分散環境では，すべての計算機に対応したデータベースライブラリを利用しない限りさまざまな計算機でクライアントを実現することが不可能である．

そのため，Harmony の一つの実現方法として，図 3.12のようにサーバ・クライアントアーキテクチャとし，Harmony/DB のみがデータベースへのインタフェースをもち，他のプロセスからのデータベースに対する操作要求を受けつける．Harmony/DB と各クラスプロセスの間でオブジェクト交換の機能を用いることにより，データベースに格納されているオブジェクトをやりとりする．Harmony では，オブジェクトの内容をそのオブジェクトのデータ構造を得ることによりバイトストリームに変換して通信をしている．

この結果、プロセス間でのオブジェクトのやりとりが可能となり，さまざまな分

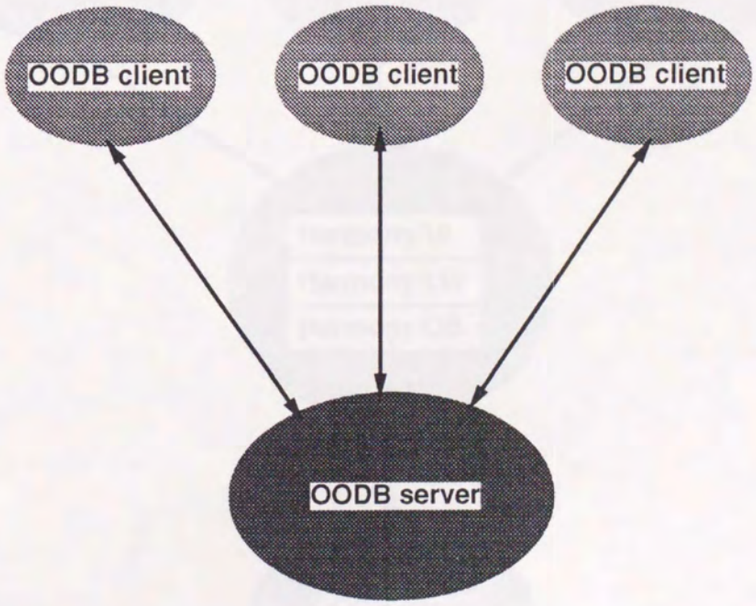

図 3.11: オブジェクト指向データベースの構成

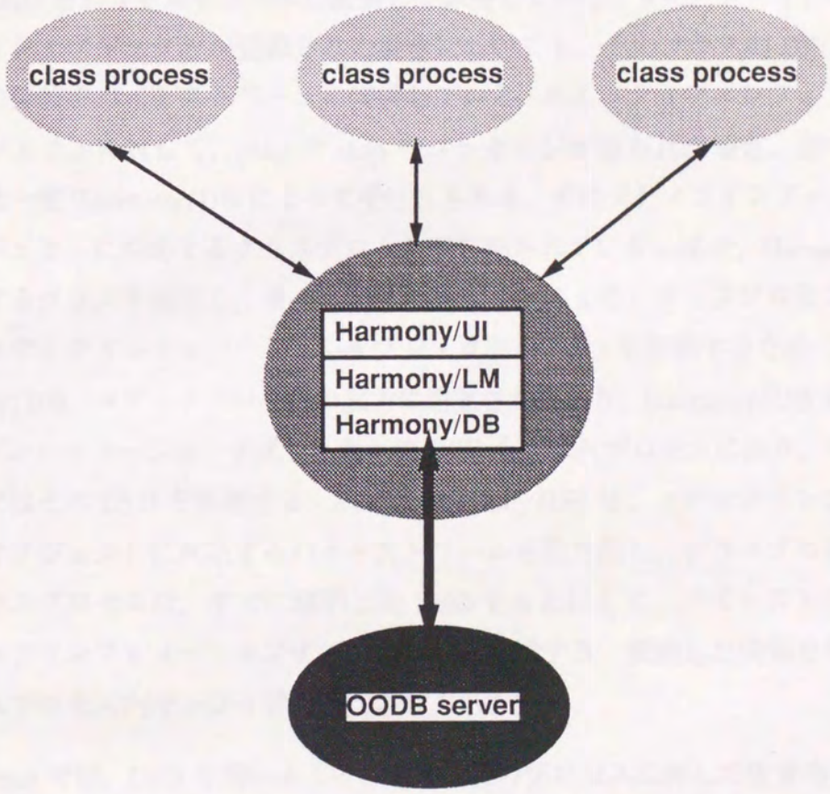

図 3.12: サーバ・クライアントアーキテクチャ

散型マルチメディアシステムに対応することができる.

3.6.1 オブジェクトの実行

Harmony では，Harmony/DB とクラスプロセス間でメッセージ交換が行われ，クラスプロセスに必要な情報を送らなければならない．そのため，Harmony/DB では，メディアインフォメーションオブジェクトのクラスが定義された時に，そのクラスの DSD をバイトストリームに変換して保持している．また，メディアインフォメーションオブジェクトが登録された場合においても，そのクラスの DSD を用いて，その情報をバイトストリームに変換している．あるメディアインフォメーションオブジェクトに対して，play や stop のメッセージが送られた場合，全てのメッセージは一度 Harmony/DB によって受けとられる．そのメディアインフォメーションオブジェクトに対応するクラスプロセスが起動されていない場合，Harmony/DB は対応するクラスを検索し，クラス情報を得ることにより，クラスプロセスを起動する．メディアインフォメーションオブジェクトの DSD を解釈するための機能は，Harmony/DB，メディアプロセスの双方に用意されており，Harmony/DB はそのメディアインフォメーションオブジェクトの DSD をクラスプロセスに送り，クラスプロセスではその DSD を解釈する．次に，Harmony/DB は，メディアインフォメーションオブジェクトに対応するバイトストリームを取り出し，クラスプロセスに送る．クラスプロセスは，すでに解釈した DSD をもとにして，バイトストリームからメディアインフォメーションオブジェクトに変換する．変換した情報をもとにしてクラスプロセス内でメソッドが実行される (図 3.13).

Hamrony では，DSD を用いることにより任意のプロセスに対して任意のオブジェクトを送ることができる．新たなメディアを採り入れたり既存のメディアのデータ構造が変更された場合でも，DSD のみを追加・変更することだけでシステムの拡張ができ，1.1の要件 5 の開放型システムとして実現することができる.

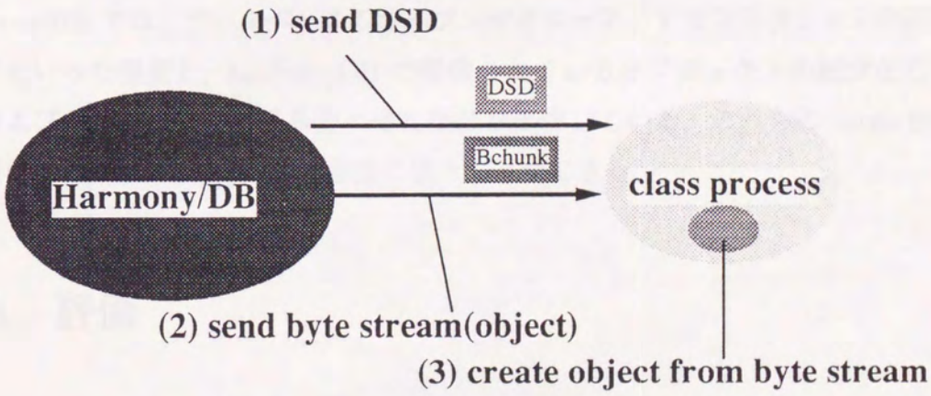

図 3.13: DSD を用いたオブジェクト実行

3.7 クラスライブラリの設計

3.7.1 アプリケーションプログラム・インタフェース

Harmony/DB の機能を提供するために，serverDB と呼ばれるアプリケーションプログラム・インタフェースを定義している．Harmony/LM および Harmony/UI は，直接 serverDB を通じて Harmony/DB の機能を利用する．また，メディアプロセスから Harmony/DB の機能を利用するために，remoteDB と呼ばれるアプリケーションプログラム・インタフェースを定義している．remoteDB は，プロセス間におよぶ遠隔メソッド呼びだしにより，serverDB を利用する． 遠隔メソッド呼びだしは，Harmony/DB が提供している機能である．(図 3.14)．

serverDB では，データベースのオープンやクローズ，トランザクションの開始や終了といった機能と，harmonyDB で提供されているオブジェクトの検索などの機能およびさらにそれらを組み合わせた機能を提供している．これらの serverDB の機能により，Harmony/DB を容易に扱うことができる．

3.8 評価

3.8.1 拡張性と移植性

ここでは，1.1の要件 5 で挙げた開放型システムとして実現することの有効性について評価する．

Harmony では，メディアの種類ごとにオブジェクトを扱うクラスプロセスが存在するため，メディアごとに独立して開発することが可能である．これにより，Harmony では，新たなメディアを追加する場合，既存のシステムを維持したまま開発が行なえる．また，メッセージ交換とメソッド実行を実現するライブラリで提供しているインタフェースをそのメディアを扱える既存のツールに組み込み若干の変更を行なうだけで，新たなクラスプロセスを容易に開発することができる．さらに，システムの基盤として，オブジェクト指向データベースを利用し，柔軟なスキーマの変更や追加といった特長を生かすことで，拡張性を高めている．実際に Harmony では，図形，テキストの二種類の静的なメディアに加えて，映像，アニメーション，音声，

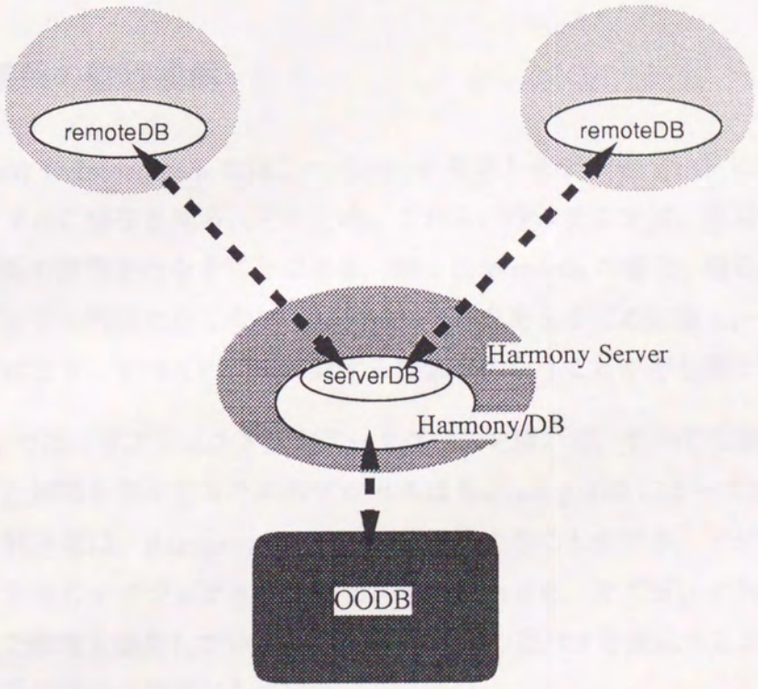

図 3.14: Harmony/DB のインタフェース

音楽の4種類の時系列メディアを扱うことができている．

Harmonyでは，映像をワークステーションの画面上に表示するために，特別のビデオプロセッサを使用しているが，最近では，さまざまな計算機に対して同様のビデオプロセッサが開発されており，このような計算機上に移植することは容易である．音楽情報に関しても同様に，MIDI端子を備えている計算機に対して，容易に移植することができる．また，Harmonyは分散型システムであり，Harmonyサーバとクラスプロセスをネットワーク上の異なる計算機に分散させることができるため，ネットワーク上のさまざまな資源を有効に活用することが可能である．

3.8.2 情報の管理機構

HyperCardやIntermediaでは，ハイパーテキストモデルのノードに相当する情報は，ファイルに保存される．そのため，これらのシステムでは，直接アプリケーションが情報の管理を行なうことになる．特にIntermediaの場合，情報を取り出す際に，メディアの種類に応じたアプリケーションをあらかじめ起動し，その上で操作することになり，すべての情報に対する検索を行なうことが不可能である．

Harmonyでは，オブジェクト指向データベースを用いて，すべての情報を一括管理しており，情報を表示するためのプロセスはHarmony/DBによって起動される．このため，利用者は，Harmony/UI上でのみ操作することができ，メディアの違いに関わらず任意のオブジェクトにアクセスするといった，オブジェクトの操作において統一した環境を提供している．このように1.1の要件4を達成することで，システム全体の操作環境を簡潔にしている．

3.8.3 プロトタイプシステムの評価

ここでは，実システムの応答性や問題点について述べる．

現在，Harmonyのプロトタイプシステムが既に完成しており，主なプログラミング言語としてC++を用いている．Harmony/UIでのET++，Harmony/DBでのVERSANTは，ともにC++のプログラムインタフェースを提供しており，Harmonyではそれらを利用している．プロトタイプシステムは，UNIX上で稼働しており，映像をワークステーションSPARCstationの画面上に表示するためにParallax社の

XVideo を搭載しており，また，ワークステーション NeXT に MIDI インタフェースボードとシンセサイザを接続することで音楽情報の再生を可能にしている．これらのワークステーションは，Ethernet で接続されている (図 3.15参照).

このプロトタイプシステムを用いて，本学紹介やミュージカル指導用のマルチメディアドキュメントによるプレゼンテーションを作成した．

これらのマルチメディアドキュメントでは，映像，アニメーション，音声，音楽，図形，テキストの 6 種類を組み合わせたリンクおよび映像，アニメーション，図形，テキストの 4 つのサブオブジェクトを始点とするリンクを付与し，プロトタイプシステムの応答性の評価を行なった．これらのリンクのナビゲーションでは，クラスプロセスからのイベントが Harmony/DB を介して Harmony/LM に送られ，Harmony/LM でリンクが検索されたのち，対応したメッセージが Harmony/DB を介してクラスプロセスに送信されるが，組合せの種類にかかわらず，すべて実測で 0.5 秒以内で行なわており，通常のプレゼンテーションを行なうには，十分な応答性能であると言える．さらに，映像オブジェクトや映像サブオブジェクトを用いたリンクでは，光ディスク上の映像であれば光ディスク装置の制御に時間を要するが，これらのリンクにおいても，0.5 秒以内でナビゲーションされている．

また，プレゼンテーション作成の経験により，二つの問題点が明らかになった．一つは画面表示の問題である．マルチウィンドウシステムを利用したハイパーメディアシステムでは，個々のメディアを表示するウィンドウが多数存在し，画面上が繁雑になる．これらのウィンドウのレイアウトを制御する機能が必要であり，リンクにおいてレイアウト情報を表現することが考えられる．

二番目の問題点は，複数のオブジェクトの表示の同期をどのようにして保証するかということである．例えば，映像に合わせて音楽を再生する場合，Harmony では，それぞれ別々のクラスプロセスで再生される．現状の UNIX のようなオペレーティングシステムでは，このようなマルチプロセスにおける同期や時間的制約の指定が困難であるため，映像と音楽の再生にずれが生じる場合がある．

そこで，次章では，マルチメディア情報間の同期に関して取りあげる．

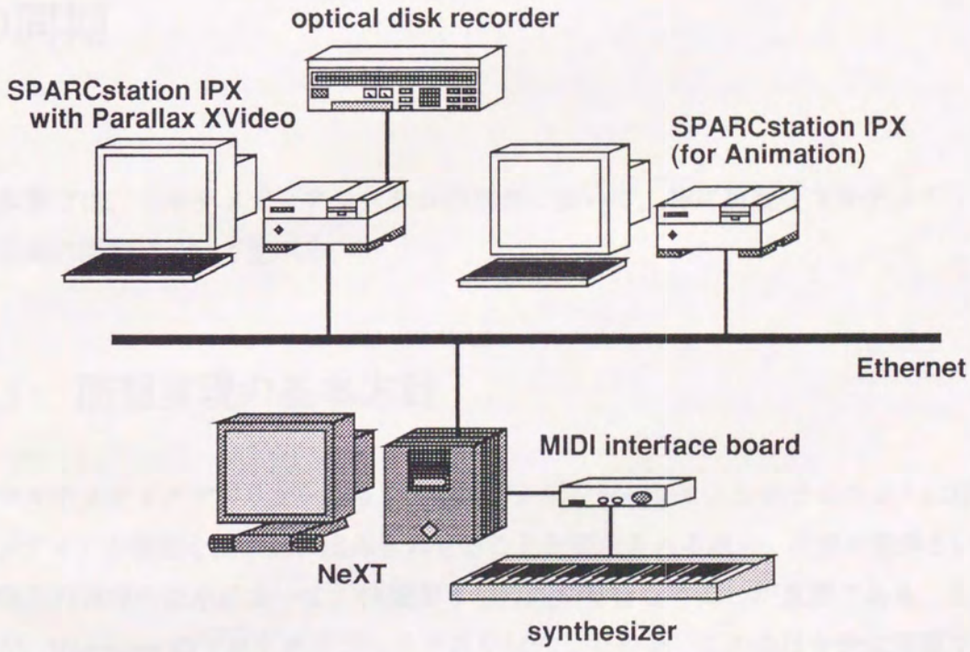

図 3.15: Harmony のハードウェア構成

第 4 章

分散型マルチメディアシステムにおける情報間の同期

本章では，マルチメディアシステムの構築において，特に重要なマルチメディア情報間の同期について述べる．

4.1 同期実現の基本方針

マルチメディアアプリケーション，特にプレゼンテーションシステムのように種々のメディアの情報を効果的に組み合わせることが要求される場合，映像や音声といった時系列情報の表示において，情報どうしの同期を行なうことが重要である．ところが，Harmony のプロトタイプシステムを試作した結果，この点は十分に実現できていないことが明らかになった．

まず，アニメーションが終った直後に映像を開始するといったシナリオの場合に，映像がすぐ表示されず，若干の遅れが生じる場合がある．また．ある情報の表示が終った直後に音声と映像を同時に表示させたい場合にも，これらの表示が同時には始まらず若干のずれが生じることがある．

これは，アニメーションや外部の光ディスク装置から表示される映像などの場合，前処理や装置を駆動させるための若干の時間が必要となり，直前の情報の終了後，起動してもすぐには表示できないためである．また，異なるメディアの情報どうしの同期においては，メディアによって準備時間に違いがあることと，Harmony のよう

にメッセージ送信でメディア表示の開始を行なった場合，表示するプロセスが，別のワークステーションにあるようなときには，メッセージがネットワークを通して伝わるため，ローカルなプロセスとの間に受信時刻のずれが生じることがあり，これが表示のずれの原因となる．

また，実時間性を保証することが困難であることもわかった．ここでの実時間性とは，映像の内容が30秒間であれば，表示が開始されてから正確に30秒後に終了することを意味する．光ディスク装置から表示される映像などはハードウェアによってあらかじめ実時間性が保証されている．しかし，アニメーション，音声，ハードディスクから表示される映像などの場合，CPUの負荷の変動などによってこの実時間性が保証されないため，表示を間引くなどの機構が必要となる．さらに，CAIに用いる場合など，マルチメディアアプリケーションの種類によっては表示する内容を損なわないためにも，表示を間引くよりも表示時間を伸ばすことで対応したい場合もある．この場合，それと同期して表示される他のメディアの情報やそれに続く以降のシナリオもこの表示時間の変更に対応する．

本研究では，複数のワークステーションを用いた分散環境上で実現されるマルチメディアプレゼンテーションにおける情報間の同期の問題を分析し，プロセス間のメッセージ通信によりOS等に手を入れることなく，アプリケーションプログラムレベルでできるだけ同期を実現する方法を提案する．

4.2 マルチメディア情報の同期モデル

Harmonyのプロトタイプシステムにおいて問題となったマルチメディア情報によるプレゼンテーションにおける同期は，以下のように詳しくモデル化することができる．

(1) 開始点同期

映像と音声を同時に開始したいなど，複数のマルチメディア情報が同期して同時刻に表示される場合の同期を開始点同期と呼ぶ．

マルチメディア情報はデータ構造や容量，蓄積形式の違いによって，表示の指示がなされてから，実際に表示されるまでにかかる時間が異なってくる．例えば，光ディスク装置上の映像は，再生が始まる前に頭出しといった操作が必要

である．このように，実際に情報が表示されるまでに行なわれる準備にかかる時間を考慮しなければ，表示にずれが生じることになる．さらに，Harmony のような分散システムでは，遠隔のワークステーション上のプロセスに情報を表示させる場合，メッセージの伝送遅延をも考慮に入れなければならない．

(2) 実時間連続同期

実時間連続同期とは，映像，音声，アニメーションといった時系列情報を表示する場合に，例えば，内容が 30 秒間であれば，表示が開始されて定期的に進み具合を監視しながら正確に 30 秒後に終了することを意味する．

特殊なハードウェアを用いて音声や映像を扱う場合には，再生時の実時間性が保証されるが，ソフトウェアのみによって再生が行なわれる場合，実時間性を保証する機構が必要となり，ワークステーションの負荷が増大するとプロセスの実行速度が遅くなるなることに対応しなければならない．

(3) 適応型連続同期

アニメーションや映像を教育的な目的に使う場合などでは，実時間性を優先して表示を間引くことよりも内容を確実に表示することが望まれることがある．この場合，重要なマルチメディア情報をゆっくり再生することで内容を確実に表示することができるが，それに同期する他のメディアの情報も合わせて進行を遅らせる必要がある．このような複数のマルチメディア情報の同期を適応型連続同期と呼ぶ．

適応型連続同期を実現するためには，ワークステーションの負荷の変動に対応して，情報の再生間隔を変化させるとともに他の情報も同期して再生間隔を変更する機構が必要である．

4.3 関連した同期モデル

マルチメディア情報間の同期を扱った代表的な研究としては，文献 [25](Muse)，文献 [27](HyTime,MHEG)，文献 [20] などが挙げられる．また，Macintosh では，マルチメディアアプリケーションのプラットフォームとして QuickTime[39] が提供されている．

文献 [25] の Muse では，time dimension と呼ばれる抽象的な時間軸に対してマル

チメディア情報の開始・終了を指定する方法で同期を行なっている．しかし, Muse では映像などは，外部のレーザーディスクから供給しているため，表示速度が変化することなく，本稿で述べる実時間性の保証と適応型連続同期を行なっていない．

文献 [27] の HyTime では，ハイパーメディアドキュメントを表現する標準構造を提案している．そこでは，音楽時間と呼ばれる抽象的な時間軸に対して開始点と継続時間を指定することで他の情報との同期を表現する．しかし, HyTime はハイパーメディアドキュメントの構造のみを定義するものでその実現については何も述べていない．

マルチメディア情報の統一的な符号化標準を定めようとしている MHEG では，連接同期, 並列同期, 基本同期, 条件つき同期, 循環同期, 連鎖同期の 6 種類の同期を定めているが, 本稿で述べる適応型連続同期のような意味的な同期に相当するものは定義されていない．また, これらの同期を現実のアプリケーションでどのように実現するかについては, 触れられていない．

文献 [20] では，時系列メディアの表示に関する同期などを種類別・方法別に，次のように分類している．

- **直列同期と並列同期**
  同期の種類として，直列同期と並列同期がある．直列同期とは，あるひとつの情報の表示において，その表示速度を一定に保つことである．また，並列同期とは，複数のメディアの情報が互いに表示速度を合わせながら，再生することである．

- **一点同期と連続同期**
  同期の方法としては，一点同期と連続同期に分類することができる．一点同期とは，情報の同期をある一点時間のみで実現するものである．連続同期とは，表示再生中のある一定時間内で同期を実現するものである．

本稿でいう単一メディアの実時間性を保証することは，ここでいう直列同期を実現することになる．また, 適応型連続同期とは, 基本的にはここでいう並列同期のことであるが, 実行時の負荷に対応して, 再生速度を変化させるところが異なっている．

また，Macintosh の OS に組み込まれている QuickTime では，時系列情報の再生における実時間性を保証しているが, 適応型連続同期は実現していない．

4.4 同期機構の実現

ここではまず，4.2で挙げたような同期を表現するために，ハイパーオブジェクトモデルを拡張する．また，これらの同期を実現するためにプレゼンテーションのシナリオをあらかじめ解析してオブジェクトの表示開始時間を決めておくとともに，実システムの個々のプロセスに同期を実現する機構を組み込む．Harmony は，複数のプロセスからなる分散型アプリケーションであるため，これらの同期は，プロセス間でメッセージをやりとりすることにより，OS を拡張することなくアプリケーションレベルで実現されている．

4.4.1 ハイパーオブジェクトモデルの拡張

ハイパーオブジェクトモデルにおいて，4.2であげた実時間再生や適応型連続同期を表現することができるように，以下のように拡張する．

(1) マルチメディア情報の表示において，実時間性を保証するかどうかを指定することができるように，メディアオブジェクトに実時間性を指定するための属性をもたせる．

(2) 複数のマルチメディア情報間の適応型連続同期を指定するために，同期させたいオブジェクトをグループ化して，ひとまとまりのオブジェクトとして表す(グループオブジェクト)．グループオブジェクトには，一つのマスターオブジェクトと複数のスレーブオブジェクトを置く．スレーブオブジェクトはマスターオブジェクトの表示速度が変更した場合，それに合わせてスレーブオブジェクトの表示を行なう．

開始点同期に関しては，一つのメディアオブジェクトから複数のメディアオブジェクトへリンクを付与することで自然に表現できる．

4.4.2 開始点同期の実現

これまでのシナリオ実行の流れ

Harmony では，クラスプロセスどうしでメッセージを交換することによりプレゼンテーションが進行する．すなわち，シナリオが開始すると，クラスプロセスがオブジェクトを表示する．あるオブジェクトの表示が終了したり，利用者がサブオブジェクトをクリックしたりするとそれぞれ **finished**，**selected** イベントがクラスプロセスから Harmony/LM の方に送られる．Harmony/LM では，送られてきたイベントから該当するリンクを探し，その内容にしたがって対応するクラスプロセスにメッセージを送ることで次のオブジェクトの表示が開始する．

ところが，この方法では，前のオブジェクトが終了してから，イベントを Harmony/LM に送り，処理し，次のクラスプロセスに送るまでの時間と，メディアによっては起動がかかっても頭出しなどの時間が必要となり，次のオブジェクトが表示されるまでにかなりの時間がかかってしまう．

シナリオの解析

そこで，開始点同期を行なうために，Harmony ではあらかじめプレゼンテーションのシナリオを解析し，個々のオブジェクトの表示開始時間を計算しておく．クラスプロセスは個々のメディアの準備時間を考慮して表示の指示を早めることで開始点同期を実現することができる．また，あらかじめ個々のクラスプロセスが自分の表示するオブジェクトの開始時間を知ることができるため，メッセージ交換を行なう必要がなく，そのための表示のずれが生じない．

ただし，利用者からの対話的な操作によってプレゼンテーションを選択的に進行する部分や，テキストなどの静的なメディアや分岐のあるアニメーションなど実行しないと再生時間のわからない時系列メディアのオブジェクトが用いられている部分がある場合，それに続くオブジェクトの開始時間を前もって計算することができない．そのため，シナリオを利用者の選択によって分岐する部分や静的メディアの部分で分割する (図 4.1)．分割された個々のシナリオは，**サブシナリオ**と呼ばれる．一つのサブシナリオは，先頭のオブジェクトの開始時間が決まると，他のすべてのオブジェクトの開始時間を決定することができる．

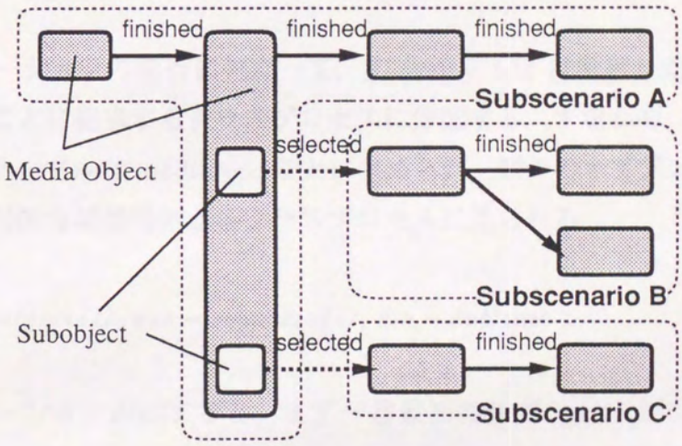
finished
finished
finished
Subscenario A
Media Object
selected
finished
Subscenario B
Subobject
selected
finished
Subscenario C

図 4.1: シナリオの分割

Harmony/LM では，シナリオを解析し，図 4.2のようなプレゼンテーションタイムテーブルを作成する．このテーブルの個々の要素は，

$$< mediaobjectID, mediatype, sub-scenarioID, mo-subtime >$$

となっている．ここでは，メディアオブジェクト ($mediaobjectID$) の表示開始時刻 ($mo-subtime$) はサブシナリオ ($sub-scenarioID$) の開始時刻からの相対時間で表されている．$mediatype$ はメディアの種類ごとに異なり，クラスプロセスを識別する．

プレゼンテーションタイムテーブルの分配

プレゼンテーションの実行に先だって，Harmony/LM は解析したシナリオをメディアの種類ごとに相当するクラスプロセスに分配する．すなわち，プレゼンテーションタイムテーブルは $mediatype$ ごとに集められ，個々のオブジェクトのサブシナリオ内の相対的な開始時刻が各クラスプロセスに送られる．

$$< mediaobjectID, sub-scenarioID, mo-subtime >$$

プレゼンテーションが始まると，まず一番最初のサブシナリオの絶対的な開始時刻が決定する．Harmony/LM は，このプレゼンテーション開始時刻の絶対時刻 ($p-starttime$) と $sub-scenarioID$ を組にして，各クラスプロセスに送信する．これによって，各プロセス内で最初のサブシナリオに属するオブジェクトの絶対的な開始時刻が決定する．以降，シナリオが進行するにしたがい，各サブシナリオの開始時刻 (絶対時刻で表される) が決定するたびに，

$$< sub-scenarioID, p-starttime >$$

を各クラスプロセスに送ることにより，プレゼンテーションが進行していく．

利用者からの対話的操作があった場合や表示時間の一定でないオブジェクトが終了した場合，それぞれ対応した selected イベントや finished イベントが発生し Harmony/LM に送られる．この場合，Harmony/LM は，別のサブシナリオにプレゼンテーションが進行したことを知り，あらたな開始時刻を各クラスプロセスに送る．

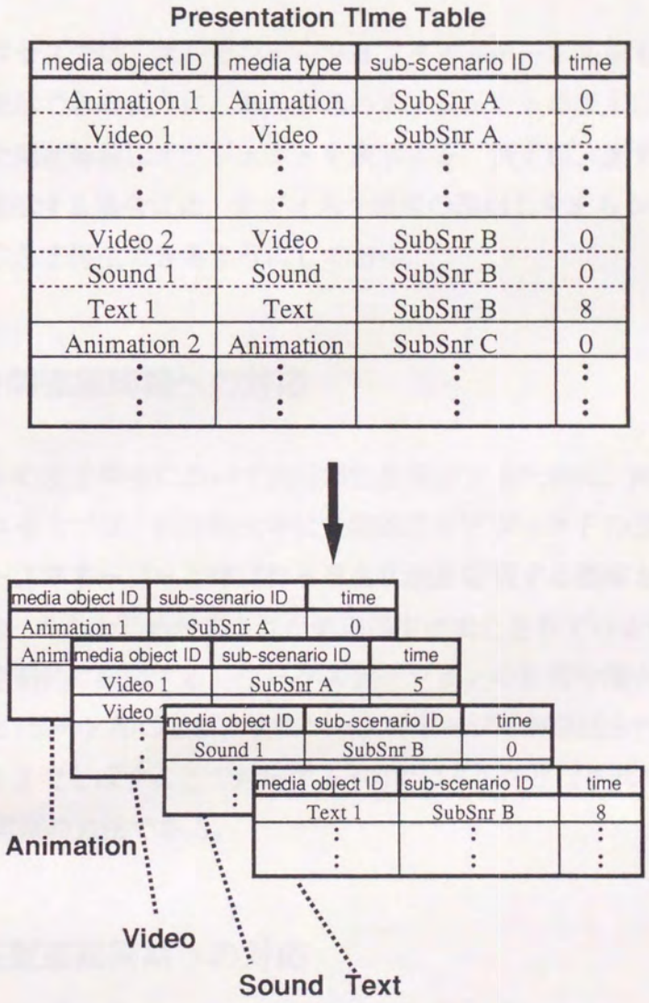

図 4.2: プレゼンテーションタイムテーブル

適応型連続同期を行なうために，あらかじめ決められた開始時刻が遅れることがある．この場合は，その時点で遅れた分だけ修正した開始時刻を送り直すことにより，それぞれのオブジェクトの開始時刻を修正することができる．また，分岐がある場合には，選択されなかったサブシナリオに対して，開始時刻を無限大として送ることにより，そのサブシナリオに属するオブジェクトの実行を無効にすることができる．

各クラスプロセスでは，プレゼンテーションタイムテーブルをもとに，表示開始時刻にすぐに表示できるように，あらかじめオブジェクトの表示に必要な準備を行い，指定された開始時刻にオブジェクトを表示する．例えば，光ディスク装置からの映像情報を表示する場合には，光ディスク装置の頭出しをあらかじめ行っておき，開始時刻にすぐさま再生できるようにしておく．

4.4.3 実時間連続同期への対応

オブジェクトの表示再生において実時間性を保証するために，時系列メディアを扱うクラスプロセスでは，表示再生中に定期的にオブジェクトの進み具合を確認するために，**ペースマネージャ**と呼ばれる再生状況を監視する機構を組み込む．ペースマネージャは，あらかじめ設定された処理速度が満たされているかどうかを図 4.3 の矢印ように定期的に確認する．ワークステーションの負荷の増大により表示が遅れている場合は (図の×印の 5,8)，次にペースマネージャが確認を行った時点で表示を正しいところまでとばすことで対応する (図では 6,9 がとばされている)．これは，QuickTime と同様の方法である．

4.4.4 適応型連続同期への対応

実時間再生と同様に適応型連続同期の場合も，各クラスプロセスのペースマネージャにおいて表示の進み具合を監視する．適応型連続同期を実現するために，イベントの種類として，オブジェクトの表示中の再生の遅れを知らせる (**paceChanged**) を加える．また，適応型連続同期のグループオブジェクト内のあるオブジェクトの再生が遅れた場合に，グループ内の他のオブジェクトに再生速度を遅らせるように指示するメッセージ (**setPace**) を用意する．

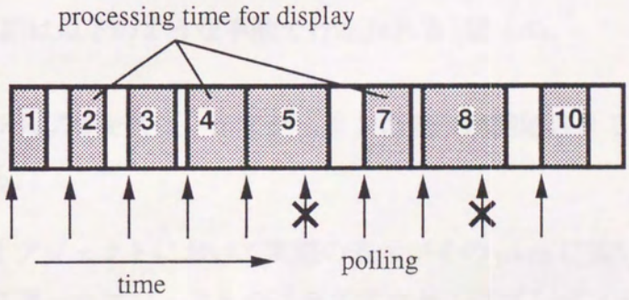

図 4.3: 実時間連続同期

適応型連続同期を行なうオブジェクト間の調整を行なうため，Harmony/LM にプレゼンテーションマネージャと呼ばれるプレゼンテーション管理部を組み込む．個々のクラスプロセスでは，基準となる再生速度が決められている．例えば，映像を扱うクラスプロセスでは単位時間あたりのフレーム数となり，単位時間を 1 秒にすると，1 秒あたり静止画 30 フレームが基 準の再生速度となる．プロセス間での再生速度の調整をおこなうため，共通の指標を以下のように定義する．

$$pace = \frac{\text{実際の再生速度}}{\text{基準の再生速度}}$$

$$(\text{通常の再生速度のとき } pace = 1, \text{静止状態のとき } pace = 0)$$

pace とは，単位時間あたりに処理される単位数である．映像を扱うクラスプロセスでは，$pace = 0.5$ が与えられた場合，単位時間を 1 秒にするならば 15 フレームの静止画が表示される．

適応型連続同期は以下のような手順で行なわれる (図 4.4)．

[step1] 個々のクラスプロセスは，オブジェクトの表示が開始されると $pace = 1$ として再生する．

[step2] マスターオブジェクトにおいて実際の表示がその pace に追い付かなくなった場合，マスターオブジェクトのクラスプロセスはプレゼンテーションマネージャに **paceChanged** イベントを送る．このとき，マスターオブジェクトが要求する *pace* を paceChanged イベントの引数として渡す．この時点では，表示をとばして pace を守る．

[step3] プレゼンテーションマネージャは同期しているすべてのオブジェクトに paceChanged イベントから受けとった pace の値を引数として **setPace** メッセージを送る．

[step4] メッセージを受けとったクラスプロセスは引数の値にしたがって再生割合を変更する．

[step5] マスターオブジェクトでの表示に余裕ができた場合には，step2 と同様に **paceChanged** イベントを送り，pace を元の値に近づけるようにする．

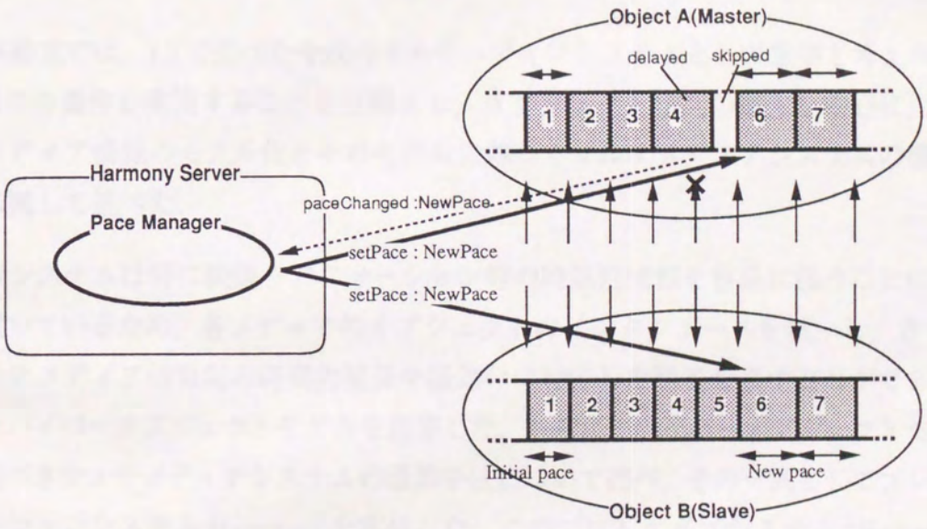
Object A(Master)
delayed
skipped
1
2
3
4
6
7
Harmony Server
Pace Manager
paceChanged :NewPace
setPace : NewPace
setPace : NewPace
1
2
3
4
5
6
7
Initial pace
New pace
Object B(Slave)

図 4.4: 適応型連続同期

第 5 章

結論

本論文では，1.1で述べた今後のマルチメディアシステムとして重要と考えられる5項目の要件を実現することを目標とし，オブジェクト指向の概念を中心に，マルチメディア情報のモデル化とそのモデルに基づくマルチメディアシステムの構築技術に関して述べた．

本システムは特に映像やアニメーション等の時系列情報を容易に扱うことに重点を置いているため，各メディアのオブジェクトのインタフェースを統一し，さらに，マルチメディア情報間の時間的関係や振舞いに対応した関連を表すことができるようなハイパーオブジェクトモデルを提案した．さらに，ハイパーオブジェクトモデルに基づきマルチメディアシステムの構築手法について述べ，その一例としてプレゼンテーションシステム Hamrony を実装した．このプロトタイプシステム “Harmony” の実時間性やマルチメディア情報間の同期の問題点を分析し，その解決法を考え，アプリケーションプログラムレベルにおける実現方法について述べた．特に，従来の研究では考慮されていなかった計算機の負荷の変動などに対処できる適応型連続同期の概念を提案した．

提案した方式を実装した結果，プレゼンテーションを行なうためには，問題のないレベルでの同期が実現されることが明らかにされた．現状の計算機のオペレーティングシステムでは，プロセスの処理にかかる時間の上限が規定されないため，アプリケーションプログラムにおいて実現している実時間性や情報間の同期が厳密に保証されない．しかし，ここで提案している実時間性や同期は，既存のオペレーティングシステムにおいても問題のないレベルで実現されており，実時間性を有するオペレーティングシステム上でこのような機能をもったアプリケーションプログラム

を実現することにより，さらに確実な実時間性や同期を行なうことができる．現在，実時間性を有するオペレーティングシステムの研究が行なわれてきており，今後は，これらのレベルにおいて時間的制約を実現する機構について研究を進めていくことも重要である．また，データベースシステムに関しては，ネットワーク上に分散した複数の利用者による同時アクセスや協調作業を実現する環境を提供することを考えている．さらに，問い合わせ言語等を用いた検索機能も備え，オブジェクトの検索を柔軟に行なえる共有システム環境の実現も今後の課題である．

謝辞

本研究を行うにあたり，本研究の機会を与えてくださり，直接ご指導くださった，大阪大学基礎工学部情報工学科の宮原秀夫教授に心から感謝致します．

本研究をまとめるにあたり，有益なご助言をくださった，大阪大学基礎工学部情報工学科の都倉信樹教授，谷内田正彦教授，並びに同大学工学部情報システム工学科の西尾章治郎教授に深謝致します．

筆者が大阪大学基礎工学部情報工学科および同大学院に在籍中，ご指導，ご教授くださった，大阪大学基礎工学部情報工学科の嵩忠雄教授，柏原敏伸教授，菊野亨教授，首藤勝教授，谷口健一教授，鳥居宏次教授，西谷紘一教授，橋本昭洋教授，同大学産業科学研究所の豊田順一教授，北橋忠宏教授，溝口理一郎教授，同大学教養部の藤井護教授，並びに同大学医学部の田村進一教授に深謝致します．

本研究の細部に渡り，熱心なご討論と有益なご助言をいただいた，大阪大学大型計算機センターの下條真司助教授，並びに同大学情報処理教育センターの松浦敏雄助教授に深謝致します．

筆者の抱くさまざまな疑問に対して熱心にご討論いただいた，大阪大学基礎工学部情報工学科の村田正幸助教授，奈良先端科学技術大学院の山口英助教授，並びに大阪大学基礎工学情報工学科情報ネットワーク学講座の方々に深謝致します．

最後になりましたが，本研究を行うにあたり，快適な計算機利用環境を与えてくださり，筆者の研究ぶりをあたたかく見守ってくださった，大阪大学大型計算機センター長松田治和教授をはじめとする大型計算機センターの皆様に深謝致します．

参考文献

[1] 表武史, 有吉勇介, 藤川和利, 下條真司, 江澤義典, 宮原秀夫：“ポータブルマルチメディアプレゼンテーションシステム Harmony のアーキテクチャ,” 情報処理学会, 第 45 回全国大会, Vol.1R-3, pp.4-103–4-104 (1992-10).

[2] 川原稔：“ハイパーメディア・オブジェクト指向システム Harmony におけるデータベースの構築,” 京都大学大学院工学研究科応用システム科学専攻修士論文 (1990-02).

[3] 藤川和利：“動画像を含んだハイパーメディア・オブジェクト指向システム Harmony の構築,” 平成元年度大阪大学大学院基礎工学研究科物理系専攻修士学位論文 (1990-02).

[4] 藤川和利, 東浩, 下條真司, 宮原秀夫：“分散型オブジェクト指向システムにおけるインスタンスの実現,” 情報処理学会, 第 37 回全国大会, Vol.2Y-2, pp.601–602 (1988-09).

[5] 藤川和利, 有吉勇介, 下條真司, 松浦敏雄, 西尾章治郎, 宮原秀夫：“協調型プレゼンテーション作成支援システム HarmonyII,” 日本ソフトウェア科学会, 第 10 回ソフトウェア研究会, SW-92-10 (1992-04).

[6] 藤川和利, 梶本雅人, 有吉勇介, 下條真司, 松浦敏雄, 西尾章治郎, 宮原秀夫: “マルチメディアプレゼンテーションシステム Harmony,” 電子情報通信学会, ヒューマンコミュニケーション研究会, 電子情報通信学会技術研究報告, HC90-7～14 (1990-07).

[7] 藤川和利, 田島孝一, 有吉勇介, 下條真司, 松浦敏雄：“実時間性を考慮した分散型マルチメディアシステムの構築,” 日本 UNIX ユーザ会, 第 20 回 UNIX シンポジウム予稿集, pp.161–172 (1992-10).

[8] 藤川和利, 下條真司, 松浦敏雄, 香取啓志, 小島増生 : "ワークステーション上での映像編集システムの試作," 日本 UNIX ユーザ会, 第 14 回 UNIX シンポジウム予稿集, pp.67-75 (1989-11).

[9] 藤川和利, 下條真司, 松浦敏雄, 西尾章治郎, 宮原秀夫 : "オブジェクト指向に基づくハイパーメディアシステム Harmony," システム制御情報学会, 第 34 回システム制御情報学会研究発表講演会 (1990-05).

[10] 藤川和利, 下條真司, 松浦敏雄, 西尾章治郎, 宮原秀夫 : "オブジェクト指向に基づくハイパーメディアシステム Harmony の構築", 電子情報通信学会論文誌 D-I, Vol.J75-D-I, No.11, pp.1015-1024, (1992-11).

[11] 藤川和利, 下條真司, 松浦敏雄, 西尾章治郎, 宮原秀夫 : "分散型ハイパーメディアシステム Harmony における情報間同期機構の実現," 電子情報通信学会論文誌 D-I, (投稿中).

[12] 松浦敏雄, 梶本雅人, 谷口健一 : "ワークステーション上での実時間アニメーションシステムとその評価", 電子情報通信学会論文誌 D-I, Vol.J75-D-I, No.7, pp.469-478 (1992-07).

[13] 松浦敏雄, 直田 創, 中村 眞 : "図形の部品化および接続包含関係の保存機能をもつ作図ツール Key3," 電子情報通信学会論文誌 D-I, Vol.J73-D-I, No.11, pp.864-872 (1990-11).

[14] 下條真司, 藤川和利, 宮原秀夫 : "分散型オブジェクト指向システム O^2NE," 情報処理学会, マルチメディア通信と分散処理研究会, SIGDSP41-4 (1988-03).

[15] M. Accetta,, R. Baron, W. Bolosky, D. Golub, R. Rashid, A. Tevanian, and M. Young : "Mach: A New Kernel Foundation for UNIX Development," Proceedings of the Summer 1986 USENIX Conference, pp.81-92 (Jul. 1986).

[16] M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, and S. Zdonik : "The Object-Oriented Database System Manifesto," Deductive and Object-Oriented Databases(W. Kim, J.M. Nicolas, and S. Nishio eds.), pp.223-240, North-Holland (1990).

[17] J.A. Blakeley, C.W. Thompson, A.M. Alashqur : "Strawman Reference Model for Object Query Languages," Proceedings of OODBTG Workshop (Oct. 1990).

[18] H.P. Brøndmo, and G. Davenport : "Creating and Viewing the Elastic Charles - a Hypermedia Journal," Proceedings of Hypertext2 (Jun. 1989).

[19] D.C.A. Bulterman, G. van Rossum, and R. van Liere : "A Structure for Transportable, Dynamic Multimedia Documents," Proceedings of the Summer 1991 USENIX Conference, pp.137–156 (Jun. 1991).

[20] D.C.A. Bulterman, G. van Rossum, and D. Winter : "Multimedia Synchronization and UNIX –or– If Multimedia Support is the Problem, Is UNIX the Solution?," Proceedings of the EurOpen Autumn 1991 Conference (1991).

[21] K. Fujikawa, S. Shimojo, T. Matsuura, S. Nishio, and H. Miyahara : "Multimedia Presentation System Harmony with Temporal and Active Media," Proceedings of the Summer 1991 USENIX Conference, pp.75–93 (Jun. 1991).

[22] D. Le Gall : "MPEG: A Video Compression Standard for Multimedia Applications," Commnications of The ACM, Vol.34, No.4, pp.46-58 (Apr. 1991).

[23] A. Goldberg and D. Robinson : "Smalltalk-80 : The language and its Implementation," Addison Wesley (1983).

[24] F.G. Halasz : "Reflections on NoteCards: Seven Issues For The Next Generation of HyperMedia Systems," Communications of The ACM, Vol.31, No.7, pp.820-835 (Jul. 1988).

[25] M.E. Hodges, R.K. Sasnett, and M.S. Ackerman : "A Construction Set for Multimedia Applications," IEEE Software, Vol.6, No.1, pp.37-43 (1989).

[26] P. Kahn and N.K. Meyrowitz : "Guide, HyperMedia, and Intermedia : A Comparison of Hypertext/Hypermedia Systems," IRIS Technical Report, Vol.88-7 (1987).

[27] B.D. Markey : "Emerging Hypermedia Standards Hypermedia Marketplace Prepares for HyTime and MHEG," Proceedings of the Summer 1991 USENIX Conference, pp.59–74 (Jun. 1991).

[28] P. Moore, A.E. Wade : "An Approach to Standard DDL for OODBMSs," Proceedings of OODBTG Workshop (Oct. 1990).

[29] J. Nielsen : "Hypertext & Hypermedia," Academic Press (1990).

[30] E. Perez : "A Strawman Reference Model for an Application Program Interface to an Object-Oriented Database," Proceedings of OODBTG Workshop (Oct. 1990).

[31] R.W. Scheifler and J. Gettys : "The X Window System," ACM Trans. Graphics, Vol.5, No.2, pp.79-109 (Apr. 1986).

[32] S. Shimojo, T. Matsuura, K. Fujikawa, S. Nishio, and H. Miyahara : "A New Hyperobject System Harmony: Its Design and Implementation," Proceedings of International Conference on Multimedia Information Systems, McGraw-Hill, pp.243–257 (Jan. 1991).

[33] S. Shimojo, T. Matsuura, K. Fujikawa, S. Nishio, and H. Miyahara : "Architectural Issues on Multimedia Presentation System Harmony," Proceedings of the IFIP TC8/WG8.1 Working Conference the Object Oriented Approach in Information Systems, North-Holland, pp.381-402 (Oct. 1991).

[34] B. Strousstrup : "The C++ Programming Language," Addison-Wesley (1986).

[35] N. Yankelovich, B.J. Haan, N.K. Meyrowitz, and S.M. Drucker : "Intermedia: The Concept and the Construction of Seamless Information Environment," IEEE Computer, Vol.21, No.1, pp.81–96 (Jan. 1988).

[36] N. Yankelovich, G. Landow, and P. Heywood : "Designing Hypermedia 'Ideabases'–The Intermedia Experience," IRIS Technical Report, Vol.87-4 (1987).

[37] G.K. Wallace : "The JPEG Still Picture Compression Standard," Commnications of The ACM, Vol.34, No.4, pp.30-44 (Apr. 1991).

[38] A. Weinand, E. Gamma, and R. Marty : "Design and Implementation of ET++, a Seamless Object-Oriented Application Framework", Structured Programming, Vol.10, No.2 (Jun. 1989).

[39] "QuickTime Developer's Kit version 1.0", Apple Computer, Inc. (1991).

[40] "VERSANT System Reference," VERSANT Object Technology (Sep. 1990).

[41] "MacroMind Director – Overview Manual", MacroMind Inc. (March 1989).