マルチメディア・データベースにおけるオブジェクト合成と動的構造化に関する研究

角谷，和俊

(Degree)
博士（工学）

(Date of Degree)
1998-09-30

(Date of Publication)
2015-01-08

(Resource Type)
doctoral thesis

(Report Number)
甲1858

(JaLCDOI)
https://doi.org/10.11501/3156259

(URL)
https://hdl.handle.net/20.500.14094/D1001858

博士論文

マルチメディア・データベースにおける オブジェクト合成と動的構造化に関する研究

1998年 8月

神戸大学 大学院

自然科学研究科 (情報メディア科学専攻)

角谷 和俊

目次

図 目 次

§ 1

序　論

1.1　本論文の背景と目的

近年，計算機ネットワークや蓄積メディアの進歩により，WWWやオンライン情報サービスなどが注目されている．また，家電機器においても，従来からの家電機器に加え，PDA，インターネットTV，CATV，およびディジタル衛星放送端末などのマルチメディアデータを扱う情報家電機器が普及しつつある．これらのデータを設計・管理するための枠組として，従来のデータベースでの対象であった文字や数値のデータ型に加え，静止画，動画，グラフィックス，および音声などを格納するマルチメディア・データベースが注目されている．しかしながら，汎用的なマルチメディアデータベースモデル，メディア間の制約を表現できるデータモデル，および一貫性を保持するための構成管理手法は確立されていない．

家電機器のユーザインタフェース仕様設計においては，ユーザインタフェース仕様(制御パネルの操作手順など)は，文字やグラフィックスなどを組み合わせて記述しなければならない．また，操作仕様を工程の初期の時点で確認するためには，プロトタイピングの手法が必要となってくる．このドメインにおいては，以下の課題がある：

- 大量のラピッド・プロトタイピングを実現可能な枠組が必要である．すなわち，プロトタイピングは試行錯誤を繰り返しながら仕様を決定するために，完全な仕様が完成するまでには大量の仕様が作成される．したがって，これらの仕様を効率的に管理する仕組みが必要である．

- プロトタイピングを段階的に構築する際に、どの段階においても一貫性が保たれていることが必要である．すなわち，途中段階においても他の変更箇所に影響されることなく動作が保証されなければならない．

- ソフトウェアデザインと，インダストリアルデザインの双方に対応できる枠組みが必要である．ソフトウェアデザインは操作仕様や手順などであり，インダストリアルデザインは，ボタン・スイッチ・制御パネルなどである．これらの仕様は独立ではなく，一方の修正により他方が影響を受けることがあるため影響を反映する必要がある．

一方，デジタル衛星放送やインターネットを用いた放送型情報サービスにおいては，大容量のデータが放送モードにより配信されるシステムが開発されている．これらのサービスは，従来のプル型サービスによるものではなく，プッシュ型サービスに基づいている．プッシュ型サービスでは，ユーザの要求，あるいは情報自身の更新により情報が自動的にユーザに配信されるので，ユーザが情報ソースに直接アクセスする必要はない．このサービスの仕組みは，ユーザが興味のあるチャンネルやソースをあらかじめ登録しておくことで，システムが各ユーザごとに定期的に情報を配信することにより実現されている．このドメインにおいては，以下の課題がある：

- 大容量のデータが放送モードにより配信されるシステムでは，ユーザが情報を受信して効率的に必要な情報を選択できる枠組が必要である．
- 配信される情報が，ハイパーリンクなどによって相互に関連付けられているハイパーメディア情報の場合，内容の更新にともなって無効リンクが生じる場合がある．また，リンクの更新を自動的に行なう必要がある．

このように，従来のデータベースの機能だけでは解決することの出来ない課題が存在する．上記の課題をマルチメディア・データベースの要件としてまとめる：

1. **単一データ型だけでは記述が困難である情報のための複合オブジェクト機構**
   複合オブジェクトにおいてはオブジェクト間の整合性が重要である．この整合性を保つための仕組みが必要である．

2. **属性やオブジェクトの振る舞いをユーザに分かりやすく提示するインタフェース**
   オブジェクトの性質を表す種々の属性やオブジェクトの振る舞いなどの，オブジェクト多様性を表現できる枠組が必要である．

3. **複雑に関連した大量データに対する検索手法および問い合わせ処理機構**
   大量のデータに対して連続的に検索を行ない構造を行なう枠組が必要である．

4. **情報の内容が時々刻々と更新されて配信されるデータの時間的一貫性を保つための管理方式**
   配信される情報が，ハイパーリンクなどによって相互に関連付けられているハイパーメディア情報の場合，内容の更新にともなって生じる無効リンクを排除し，動的にリンクを生成する方式が必要となる．

本論文ではこのような要件を満たす手法を提案し議論を行なう．また，実現したシステムについての評価についても述べる．

1.2 本論文の構成

本論文では，前節で述べたマルチメディア・データベースにおける要件を満たすための手法について論じる．本論文は6章から構成される．

第2章では，ビジュアルプロトタイピングにおけるオブジェクトモデルの構築と検索について述べる．ここでは、オブジェクトモデルのクラス階層をスキーマとして，クラスから生成されたインスタンスの集合を仕様のバージョンとしてモデル化を行う方式について述べ，専用言語として設計した記述言語*LAUSIV*について述べる．また，大量のバージョンから検索を行う方式について述べる．これらの方式により，スキーマ進化と仕様のバージョンを明確に分離することが可能となるため，版管理における一貫性の保持を容易に実現可能であることを示す．

第3章では，機器のインタラクション・デザインの枠組みについて述べる．まず，家電機器インタラクション・デザインの要件について議論を行う．インタラクション・デザインモデルとして，ユーザインタフェースと主機能を分離した機能操作制約モデル (FICモデル) を提案する．また，このモデルに基づいて家電製品インタラクションデザイン支援システム *Visual CASE*の設計について議論を行う．このシステムは，ビジュアルシミュレーションの枠組みを提供するものである．また，実開発へ適用した結果と評価についても述べる．さらに，家電ソフトウェアをインタラクション・デザインの観点から部品化を行う手法について述べ，実製品を用いた分析・検討について述べる．

第4章では，デジタル衛星放送システムを用いて送信された大量データの構造化について述べる．まず，一定周期ごとに連続して送信されるテキストデータに対して，連続的に問い合わせを行うことにより、ユーザの所望するデータを検索する方法について述べる．ここでは，電子番組ガイド (EPG: Electronic Program Guide) を対象データとする．また，デジタル放送における，インタラクティブ・データ配信のためのカルーセル方式による配信について述べる．

第5章では，放送型のハイパーメディアにおいて，情報間の参照情報であるリンクを管理する方式について述べる．特に，時間の経過に伴って変更される情報，時系列データ，およびリアルタイム性を持つ情報に対する時間依存リンクについて述べ，時間的一貫性を保証する枠組みについて論じる．また，サーバー側とクライアント側のバージョン管理についても述べる．さらに，提案する方式に基づいて設計したプロトタイプシステム *Mille-feuille*の実装について述べる．特に，拡張可能なマーク付け言語 (XML) を用いた実装方式について議論を行なう．

最後に，**第6章**では，本研究で得られた研究成果をまとめ，さらに，今後の展開について述べる．

§ 2

ビジュアル・プロトタイピングのための製品仕様オブジェクトモデル

2.1 緒言

現在，ソフトウェア開発においてプロトタイピング技術は注目を集めており，たくさんの成果が報告されている[1]．しかし，これらの手法はプログラムコードの仕様のみに適用されており，機能仕様および操作仕様には有用ではない．機能仕様および操作仕様はSUI(Solid User Interface)を持つ製品(例えば制御機器や家電製品)では，特に重要な要素である．ここで，機能仕様とは，製品の本来の目的である主機能を駆動するための物理デバイスの仕様や動作シーケンスの仕様である．この仕様の中で，特に，ソフトウェアで実現できる仕様を対象とする．例えば，モータ，タイマー，センサ制御シーケンスなどの仕様である．一方，操作仕様とは，操作や表示のための物理デバイスの仕様や操作順序などの仕様である．この仕様についても，同様にソフトウェアで実現できる仕様を対象とする．例えば，ボタン，表示管，ランプなどの仕様である．これらの仕様は，テキストやドキュメントで表現しても直観的に理解することは困難である．例えば,『あるボタンを押した時，あるLEDと別のLEDが点灯する．さらに，実行ボタンが押された後，ある機能が駆動される．この機能が終了した時点で，あるランプが点滅する』といった仕様は，図式やアニメーション等で表現した方が理解しやすいと考えられる．これらの仕様を視覚的にシミュレーションできるプラットフォームが有れば，初期設計の段階で，ハードウェアによる試作品(モックアップ)を作成せずに，プロトタイピングによる仕様検討が可能である．

製品仕様には，さまざまな種類の仕様がある．例えば，形状・重量・色・価格・機能・操作などである．対象としている製品仕様は，機能仕様と操作仕様である．これらの仕様は，製品展開の際に付加，あるいは削除の対象となる単位である．従って，新規機能の追加，あるいは操作方

法の変更などにより，仕様の組合せが変更されるため，いくつかのバリエーションが存在する．

例えば，家電製品の分野では，ユーザニーズに対応して，たくさんの製品バリエーションが存在する．電子レンジの例であれば，単機能モデル，グリル付きモデル，コンベクション付きモデルなどである．また，イギリス仕様，フランス仕様，などのように仕向け地によるバリエーションも存在する．一方，一つの製品を開発する過程においても仕様候補が存在する．我々の経験では，一つの製品を開発するのに約100種の仕様が検討されている．例えば，あるセクションで年間100種の製品を開発すると，約10,000種の仕様を検討しなければならない．

これまでに，いくつかのバージョンモデルや構成管理手法が提案されてきた[5][11]．しかし，これらのモデルや手法は，複雑で大規模なバージョン管理に対しては有効ではなかった．なぜなら，大規模な変更はメジャー番号で，小規模な変更はマイナー番号で表現するしかなく，どちらも同じ枠組でしか扱えないためにバージョン間の関係が入り組み，変更の意図が理解しにくくなるからである．

我々のアプローチの概要を以下に示す．まず，オブジェクトモデルのクラス階層をスキーマとして，クラス階層のクラスから生成されたインスタンスの集合を仕様のバージョンとしてとらえる．次に，元となるクラス階層(基本スキーマとよぶ)から，クラスの追加・削除・変更により，派生クラス階層(派生スキーマとよぶ)を生成する．また，基本スキーマから生成された仕様のバージョンと，派生スキーマから生成された仕様のバージョンとは，生成されたスキーマが違うためそれぞれ分離して管理する．これらのアプローチは，スキーマの進化と仕様のバージョンとを明示的に分けることで，版管理における一貫性の保持を容易に実現可能とするものである[14]．

我々の目的は，製品仕様の変更の柔軟性，および再利用性を向上することである．オブジェクトモデルの柔軟性は，設計者に迅速で直観的な設計を提供する．すなわち，設計者が試行錯誤的に視覚的に検証を行い，容易にプロトタイプを行うことを可能とする．一方，オブジェクトモデルの再利用性は，製品仕様の進化を扱う枠組を提供する．すなわち，以前の仕様と現在の仕様との比較・検証を可能とする．一般に，オブジェクトモデルでは，クラス階層が進化した場合に，クラス階層の一貫性を保証する機構が必要である[9][18]．我々はリリース手法を用いて，クラス階層とインスタンスとの一貫性・整合性を保持する手法を提案する．

本章の構成は以下の通りである．まず，2.2節ではビジュアルプロトタイピングの要件について議論する．2.3節では製品仕様オブジェクトモデルの基本概念について述べた後，記述言語，状態の詳細化，スキーマの一貫性，バージョン管理について議論する．2.4節では検索機能について述べる．2.5節ではまとめと今後の課題について述べる．

2.2 ビジュアル・プロトタイピング

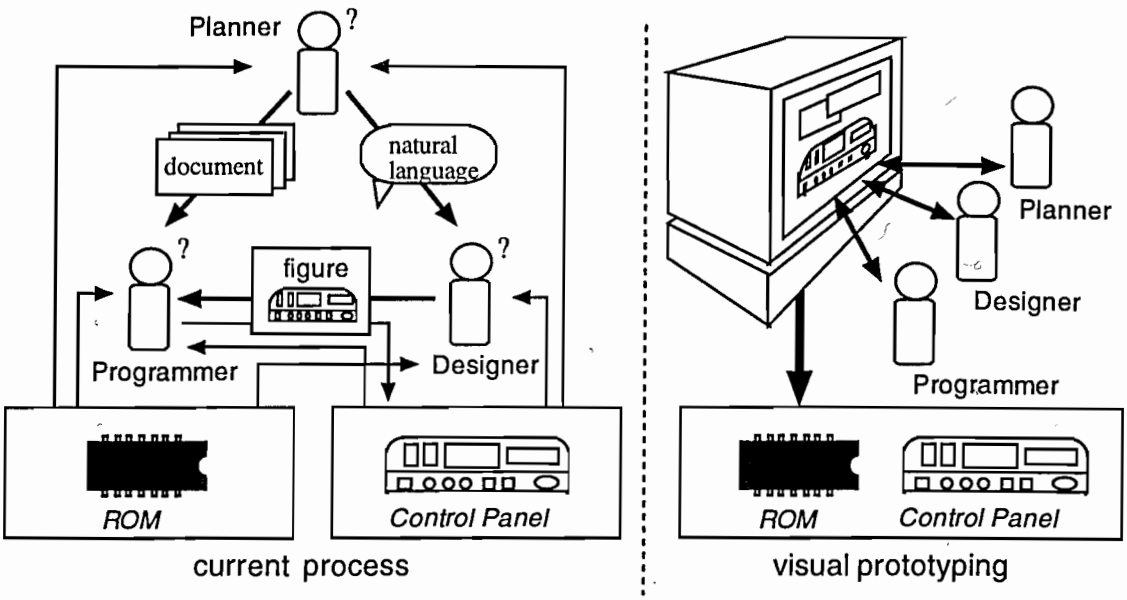


図 2.1: 現状の開発工程とビジュアルプロトタイピング

一般に，ソフトウェア開発におけるプロトタイピング・システムの要件としては，以下の4点が挙げられる [3]. (1) 稼働性 (executability), (2) 環境導入性 (target environment introducibility), (3) 作成や変更の迅速性 (rapid constructability/modifiability) , および (4) 進化性 (step–wise refinability) である.

(1) と (2) においては，仕様記述言語と開発環境が必要であり，(3) と (4) では仕様やデータ管理システムが必要であると考えられる.

プロトタイピングは製品開発において，設計品質を向上させるために非常に有用である．設計者は，試行錯誤を繰り返しながらたくさんの仕様候補を検証することができる．プロトタイピングの中で，特にビジュアル・プログラミング [12] の手法を用いているものをビジュアル・プロトタイピングと定義する。ビジュアル・プロトタイピングは，テキストやドキュメントでは表現が困難な仕様の作成に有用である．例えば，CISP[4] は，制御機器のコントロールパネルのためのプロトタイピングツールである．CISP は Hyper Card の拡張であり，VTR などの機器分野に特化したオブジェクトを積極的に導入している．しかし，以下の二つの点で問題がある．

- プロトタイピングで検討された結果が直接ターゲットシステムに反映されないこと
- 仕様検討の数が少ない場合は仕様の管理が容易であるが，検討数が増加した場合は管理ができなくなり仕様の再利用が困難になること

これらの問題を解決するには，まず，ターゲットシステムとの整合性を保つために，すべての開

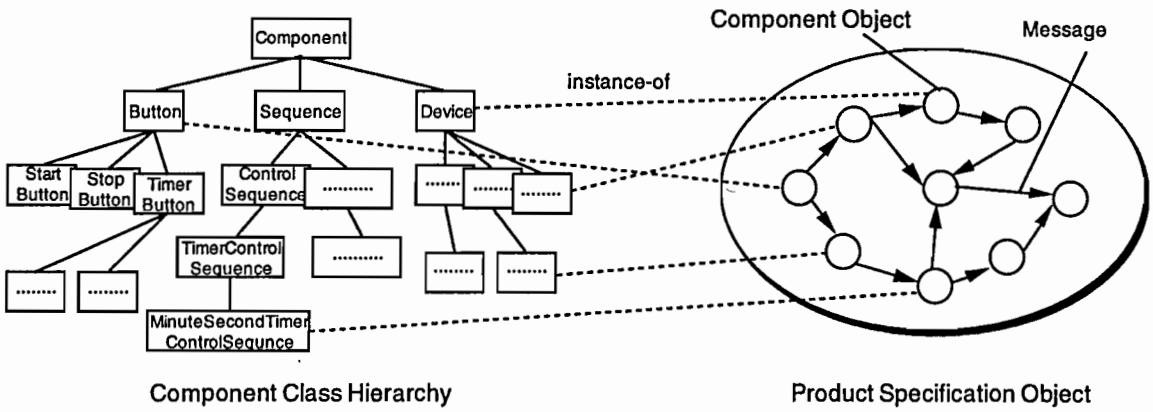


図 2.2: 部品階層と製品仕様オブジェクト

発プロセスを通して交換可能な設計情報の枠組みが必要である．実世界を直観的に表現できるオブジェクト指向モデル [6] は，この枠組みに適しており，これを基にドメインに適合したオブジェクトモデルを設計することで解決をはかる．このとき，企画，意匠デザイン，プログラム開発の各工程ごとのビューを提供し，それぞれの立場で仕様を反映する仕組みが必要である (図2.1)．また，設計仕様は頻繁に変更されるものであるが，一般にオブジェクト指向分析/設計では，適切なクラスライブラリー標準部品を用意するコストが大きい [7]．従って，クラスライブラリー間の整合性を保ちながら，仕様の追加，変更，および削除に対応するための拡張が必要である．

2.3 製品仕様オブジェクトモデル

2.3.1 本モデルの基本概念

製品の機能仕様と操作仕様を管理するために，機能とそれを駆動する操作とにより，製品仕様をオブジェクトモデルとして表現する．すなわち，製品仕様を構成する部品を部品オブジェクトとして捉え，部品オブジェクトの集合を製品仕様オブジェクトとして表現する．機能仕様は機能を表現する部品オブジェクト，操作仕様はこのオブジェクトを駆動する部品オブジェクトとしてそれぞれ定義される．

提案する**製品仕様オブジェクト**は**部品オブジェクト**を構成要素とするコンテナ型のオブジェクトである．一つの製品仕様オブジェクトは一つの製品仕様に対応する．製品仕様オブジェクトの生成は，部品オブジェクトを登録することで行われる．この時，部品オブジェクトは，部品オブジェクトの部品クラス階層 (以下，部品階層とよぶ) からインスタンスオブジェクトとして生成される．図2.2に部品階層と部品オブジェクトから構成される製品仕様オブジェクトとを示す．図中の矢印は部品オブジェクト間のメッセージを示している．

一つの部品階層から複数の製品仕様オブジェクトを作成することが可能である．一般に，コンテナ型オブジェクトは，その構成オブジェクトを管理する枠組みを提供している．コンテナ型オブジェクトに関する議論は[17]に詳しい．コンテナ型オブジェクトがその構成オブジェクトに対して何も束縛しなければ，構成オブジェクトは追加・削除に関して何も制約を受けない．従って，コンテナ型オブジェクトは，構成要素を柔軟に入れ換え可能な枠組みを提供することが可能である．

2.3.2 記述言語 LAUSIV

記述言語*LAUSIV* [1]は我々が提案するオブジェクトモデルのための動作記述言語である．部品クラスは，状態属性(state)，属性(attribute)，振る舞い(behavior)の3種類の情報から構成される．状態属性には部品オブジェクトがとりうる状態，属性には変数やパラメータ，振る舞いにはメッセージを受信した時の動作がそれぞれ記述されている．状態属性は従来のオブジェクトモデルでの属性の一種と考えられる．しかし，部品オブジェクトの性質を決定するための重要な情報であると考えられるので，状態をオブジェクトの属性から独立して考える．

以下は，Sequenceクラスのサブクラスとして，TimerSequence クラスを記述した例である．

```
class TimerSequence : Sequence{
 /* definition of state attributes */
 state:
   timer_state =
         {'waiting','setting','executing'};
                ....
                ....

 /* definition of general attributes */
 attribute:
   integer   start_time;
   integer   end_time;
   integer   interval;
                ....
                ....

 /* definition of behavior */
 behavior:
   SetTimer  from < class TimerButton> {
     if (timer_state == 'waiting'){
          timer_state = 'setting';
          interval = end_time - start_time;
       }
                ....
                ....
   }
}
```

この例では，timer_state が 状態属性として，start_time, end_time, intervalが属性としてそれぞれ定義されている．また，振る舞いとしてSetTimer が定義されている．

[1]LAUSIV: ビジュアルな情報を implicit に言語で記述することから ＶＩＳＵＡＬのつづりを逆順に並べている．

2.3.3 状態の詳細化

状態属性は一般の属性のような継承機構のほかに詳細化という継承を行う．サブクラスの状態属性はスーパークラスの状態属性がさらに分割された状態属性を含むことが可能である[2]．逆に，上位クラスの状態属性は，詳細化された派生クラスの状態属性に対して汎化された状態属性であると考えられる．

下記の例では，TimerSequence クラスで timer_state が waiting, setting, executing と定義されているが，サブクラスMinuteSecond TimerSequence では setting が setting_minute と setting_second に詳細化されている．

```
class TimerSequence {
 state:
  timer_state =
       {'waiting, 'setting', 'executing'}
}

class MinuteSecondTimerSequence :
                        TimerSequence {
 state:
  timer_state.setting =
     {'setting_minute', 'setting_second'}
}
```

状態属性を属性から独立させ，更に，継承機構を用意することで，OMT[10] などの手法で問題であった，オブジェクトモデル，動的モデル間の不整合の解消が可能である．すなわち，オブジェクトモデル，動的モデルを別々に設計するのではなく，クラス定義のときに同時に設計する枠組みを提供することで，オブジェクトの動的な側面をクラス定義に明示的に反映することが可能である．上記の例では，TimerSequence クラスでは，waiting, setting, executing がオブジェクトの動的な側面であるが，サブクラスMinuteSecond TimerSequence では，setting に関して違う動的を記述することが可能である．

2.3.4 スキーマの一貫性

現在，スキーマ進化のための方式がいくつか提案されている [18][9]．Zicari らは，2 つの一貫性について議論を行っている [18]．一つは，structural consistency であり，もう一つは，behavioral consistency である．structural consistency はデータベースの静的な特性を扱い，一方，behavioral consistency は動的な特性を扱うものである．しかし，ここで提案されている behavioral consistency は一貫性を保証するのには非常に有効であるが，スキーマ進化に対して非常に強い制約である．特に，スキーマが頻繁に変更されるような場合 (例えば，プロトタイピングなど) においては，静

[2] 上位クラスの状態属性を，そのまま継承することも可能である

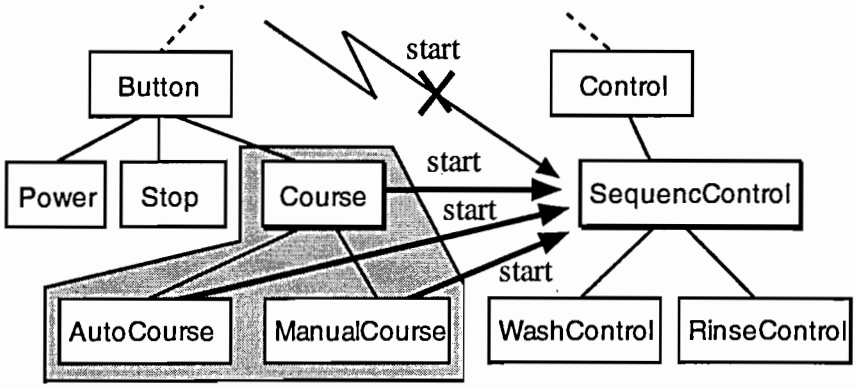


図 2.3: メッセージの送信・受信制約

的に一貫性が保たれてさえいれば，すべての組合せにおける動的な一貫性が保証されていなくても，部分的な一貫性が保たれていれば十分である．我々は，この枠組のために weakly behavioral consistency を提案する．weakly behavioral consistency は，メソッドの不実行(ランタイムエラー)の回避，および，想定しないメソッドの実行の回避を実現することが可能である．

我々の提案する手法では，メッセージの送信先は，受信するオブジェクトを指定するのではなく，受信する特定のクラスを指定することが可能である．すなわち，ある部品オブジェクトからある特定の部品クラスに送信されたメッセージは，製品仕様オブジェクトに存在する指定された部品クラスおよびそのサブクラスの部品オブジェクトが受けとる．もし，指定された部品クラスおよびそのサブクラスの部品オブジェクトが製品仕様オブジェクトに存在しない場合は，どの部品オブジェクトもメッセージを受けとらない．この時，受信する部品オブジェクトが存在しないという理由でのランタイムエラーは発生せず，メッセージが無視されるだけである．

同様に，メッセージの受信側は，特定のクラスからのみ受信することを指定可能である．すなわち，メッセージを受信した部品オブジェクトは，そのメッセージに関して，送信した部品クラスにより受信するかどうかを判断できる．この時，指定した部品クラスおよびそのサブクラスの部品オブジェクトからのメッセージは受信し，それ以外の部品クラスのオブジェクトからのメッセージはたとえメッセージ名が同名でも受信しない．

総括すると，提案する手法は以下の2つの制約にまとめられる．

送信制約 送信側オブジェクトは，受信側オブジェクトのクラスを指定することが可能である．この制約は以下の記法で記述される．

```
MessageName to <class ReceiverClassName>
```

ここで，`ReceiverClassName` は受信オブジェクトのクラス，`MessageName` はメッセージ名

(振る舞い)である.

受信制約 受信側オブジェクトは，送信側オブジェクトのクラスを指定することが可能である．この制約は以下の記法で記述される.

```
MessageName from <class SenderClassName>
```

ここで，SenderClassName は送信オブジェクトのクラス，MessageName はメッセージ名(振る舞い)である.

以下に，図2.3に示す部品階層におけるメッセージの送信・受信制約の記述例を示す．Course クラスは，振る舞いonにおいてSequenceControl クラスにメッセージStart を送信する．一方，SequenceControl クラスではStart メッセージを受信する．但し，Course クラス，およびCourse クラスのサブクラス以外から送信されたStart メッセージは受信しない．すなわち，Button クラス，およびButton クラスのサブクラスであるAutoCourse クラスとManualCourse クラスとからのStart メッセージは受信するが，その他のクラスからのStart メッセージは受信しない.

```
/* Sender Class */
class Course : Button {
                    ....
 behavior:
  on {
                    ....
     Start  to < class SequenceControl> {
                    ....
                    ....
  }
}
/* Receiver Class */
class SequenceControl : Control {
                    ....
 behavior:
   Start  from < class Course> {
          if (state == 'waiting'){
              state = 'setting';
                    ....
                    ....
          }
   }
}
```

メッセージの送信・受信制約は，クラス階層の進化にともない，あるメッセージに対するクラス間の整合性が保てなくなった場合に有効である．すなわち，双方のクラスを修正するのではなく，一方のクラスに制約を記述することで整合性を保証することが可能である．整合性のチェックはクラス定義の際に以下の手順で行われる．まず，ある変更に対して影響を受ける可能性のあるすべてのクラスを，メッセージの送受信関係から自動的に抽出する．これは，送信先あるいは受

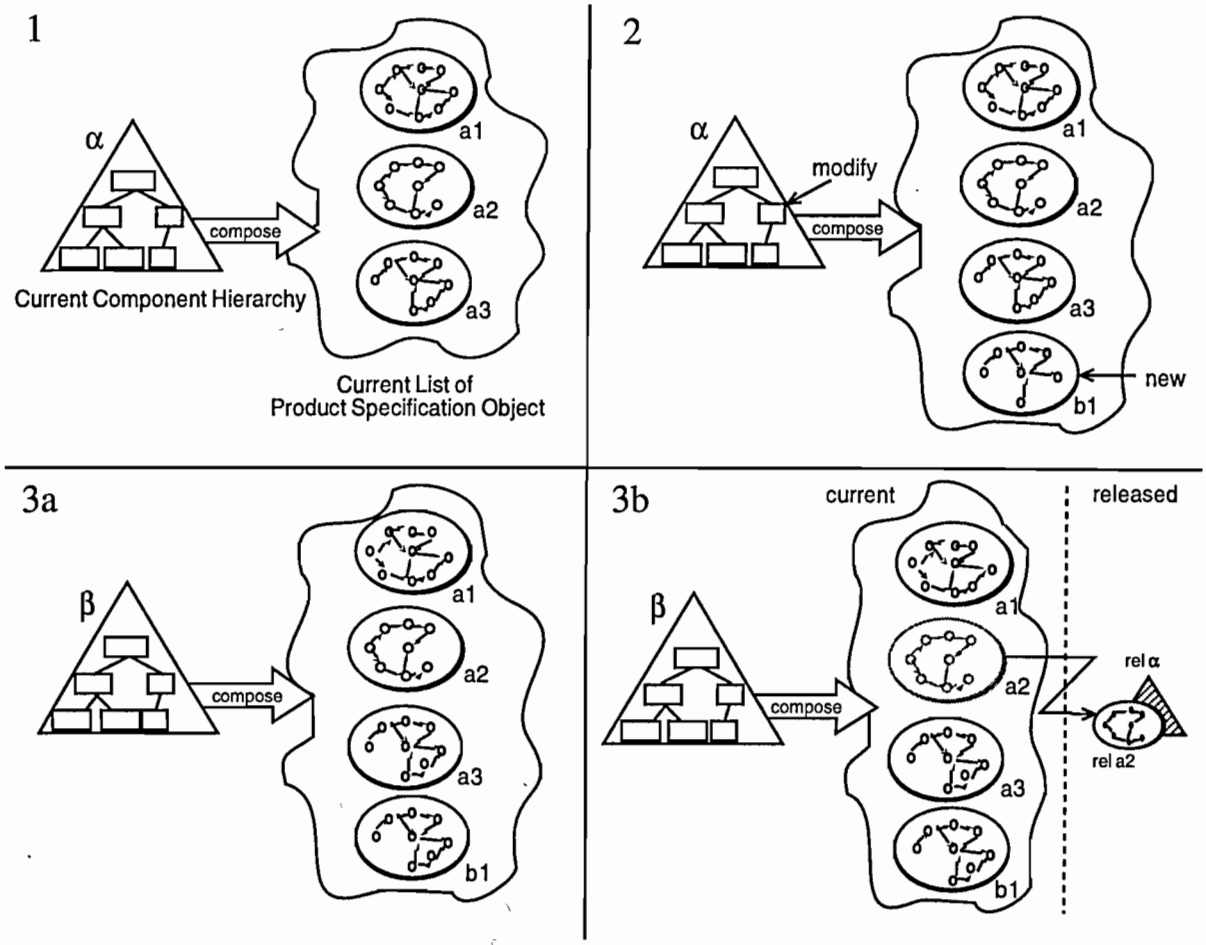


図 2.4: リリース手法

信先にこのクラスが指定されているすべてのクラスを抽出することによって行う．次に，このクラスの振る舞いをユーザがチェックし，必要があれば送信・受信制約を用いてユーザが変更する[3]．

2.3.5 バージョン管理

製品仕様オブジェクトにはたくさんのバージョンが存在する．なぜなら，ひとつのクラス階層から複数の製品仕様オブジェクトが生成可能であるからである．例えば，電子レンジの場合，'95-Grill-MODELのクラス階層から，'95-English-design ，'95-French-design ，および'95-German-designが生成される．ここで，'95-Grill-MODELは基本モデル，'95-English-design ，'95-French-design ，および'95-German-design はそれぞれ派生モデルと考えられる．

一方，クラス階層は新規クラスの追加，クラスの変更，クラスの削除などにより変更される．例えば，'95-Grill-MODEL から'96-Grill-MODELへの変更の過程において，10-MinutesButton

[3]クラスの動的なチェックを自動的に行い，振る舞いを自動的に変更する手法については現在検討中であり今後の課題である．

クラスが追加されたり，SteamSensor クラスが変更されたりする．この時，クラス階層と製品仕様オブジェクトは矛盾するかも知れない．例えば，SteamSensorクラスの仕様が変更された場合，'95-English-designと'95-German-designは動作するが，'95-French-design は動作しないかもしれない．なぜなら，'95-Grill-MODEL のクラス階層においてSteamSensor クラス '95-French-design に限り，正しく動作しない場合があるからである．

我々は，これらの問題を解決するために，コンフィギュレーション管理手法を提案する．この手法をリリース手法とよぶ．この手法は，クラス階層が進化しても，既存のクラスの動作を保証するものである．図2.4はリリース手法を示している．

Step 1 製品仕様オブジェクト a1 は最新のクラス階層 α から生成される．同様に，a2 と a3 も生成される．このとき，製品仕様オブジェクトの最新リストは a1, a2, および a3である．

Step 2 クラス階層 α においてあるクラスが更新され，新しい製品仕様オブジェクト b1が生成される．この時，もし a2が変更されたクラスのオブジェクトを含んでいたら，製品仕様オブジェクトの動作が正しいかどうかを確認する．もし，正しい動作ならば，Step 3a へ．そうでなければ，3b へ．

Step 3a 最新のクラス階層はクラス階層 α から進化した β であり，製品仕様オブジェクトの最新リストに b1 が追加される．

Step 3b 最新のクラス階層はクラス階層 α から進化した β であり，a2 は *rel* α と一緒に *rel a2* としてリリースされる．この場合，製品仕様オブジェクトの最新リストから a2が削除され，b1が追加される．

リリースされた製品仕様オブジェクトのバージョンは最新リストから外される．この時，製品仕様オブジェクトが生成されたクラス階層も製品仕様オブジェクトと一緒にリリースされる．製品仕様オブジェクトが生成されたクラス階層も製品仕様オブジェクトと一緒にリリースされる理由は，(1) 製品仕様オブジェクトが正しく動作することを保証するため，(2) リリースされたクラス階層も更に独自に進化することが可能なため，である．最新リストから外されなかった製品仕様オブジェクトは，リリースされた古いクラス階層で作成されているにもかかわらず，最新のクラス階層で正しく動作するので最新リストに保存される．古いクラス階層と一緒に保存されない理由は，一般にプロトタイピングは，次々に機能追加・機能変更を行い，それらを検討する試行錯誤的なプロセスであるため，且つ，検討が基本的に最新の仕様候補であるためである．頻繁に変更を行うので，プロセス全体で整合性をとるのは一般的には困難である．但し，以前に作成した仕様候補も保存しておきたいため，リリース手法を使用して動作を保証している．

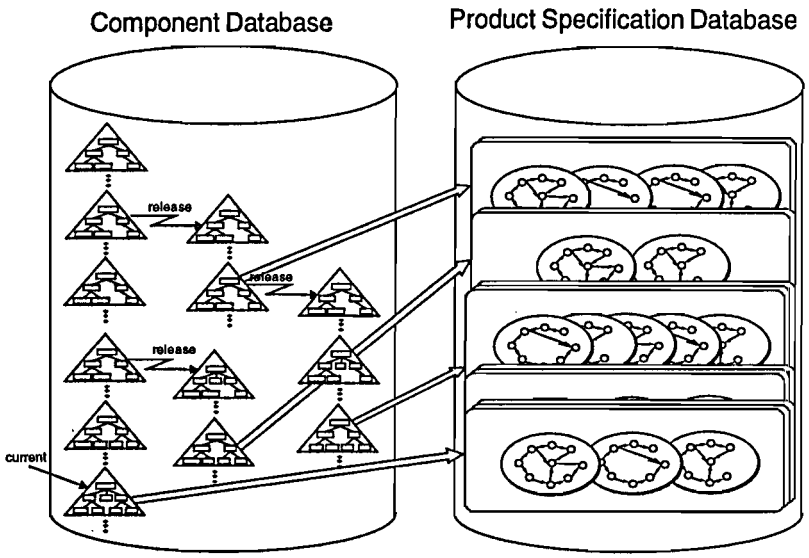


図 2.5: 部品データベースと製品仕様データベース

リリースされたクラス階層は，図2.5に示すように，どのクラス階層から派生したかという情報と共に部品データベースに格納される．また，リリースされたバージョンは，どのクラス階層で生成されたかという情報と共に製品仕様データベースに格納される．この方法により，製品の基本モデルと派生モデルを効率的に管理することが可能である．例えば，NorthEuropean-MODELがSweden-MODELやNorway-MODELに進化することも可能であり，また，更にNorthAmerican-MODELなどに進化しうることも可能である．

2.4 製品仕様データベースにおける操作に基づく検索機能

2.4.1 製品仕様の検索

製品の操作に関する仕様が格納された製品仕様データベースに対してSQLなどの従来の検索式のようにテキストを用いて検索式の入力を行なうのではなく，検索のための入力自体を製品に対する操作により行なう検索方法について述べる．この方法における質問指定は，製品に対する操作列(製品の操作ボタンに対する操作の順序列)となる．「製品Pと製品Qが与えられた時，製品Pに対する操作列で実行される機能が，製品Qでその機能に対応する機能を実行するためにはどの操作列が必要か」を検索する．例えば，製品Pでのタイマー予約機能は，

1. タイマー設定ボタンを押す

2. 時間ボタンを押す

3. 分ボタンを押す

の操作列で実行される．一方，製品Qでは，この機能は，

1. タイマー設定ボタンを押す

2. モード設定ボタンを押す

3. 時間ボタンを押す

4. 分ボタンを押す

の操作列で実行される．一般にある機能を実行するための操作列は複数存在する．従って，検索結果は複数存在することになるが，複数の候補が存在する場合の戦略についてはユーザの検索意図に依存するので，特に規定しない．

ここで，前節で述べた製品仕様データベースのための製品仕様オブジェクトと部品オブジェクトを用いてこの検索方法を考える．製品仕様オブジェクトは製品に対応し，部品オブジェクトは製品を構成する部品に対応する．また，部品オブジェクトの中でユーザが操作出来る部品オブジェクト (例えばボタン，スイッチ等) に対するアクションを操作とする．また，操作の順序列を操作列とする．

2.4.2 操作に基づく検索方法

製品仕様オブジェクト $\mathcal{P}$，製品仕様オブジェクト $\mathcal{Q}$，製品仕様オブジェクト $\mathcal{P}$ に対する操作列の三つの情報を入力として検索を行ない，それに対する製品仕様オブジェクト $\mathcal{Q}$ での操作列を求める．この時，前章で示した状態の抽象化・詳細化，部品オブジェクト間の制約をどのように用いるかを述べる．以下に検索の手順を示す．

入力

- $\mathcal{P}$, $\mathcal{Q}$: 製品仕様オブジェクト
- $[o_1, o_2, o_3,...]$: 製品仕様オブジェクト $\mathcal{P}$ に対する操作列

出力

- $[\pi_1, \pi_2, \pi_3,...]$: 製品仕様オブジェクト $\mathcal{Q}$ に対する操作列

アルゴリズム

STEP 1 製品仕様オブジェクト $\mathcal{P}$ に対する操作列 $[o_1, o_2, o_3,...]$ により状態変化を起こす製品仕様オブジェクト $\mathcal{P}$ に含まれている部品オブジェクト x を取り出す.

STEP 2 製品仕様オブジェクト $\mathcal{P}$ に対する操作列 $[o_1, o_2, o_3,...]$ による部品オブジェクト x の属するクラス $\mathcal{X}$ での状態遷移 $\{\sigma_1, \sigma_2,...\}$ を取り出す.

STEP 3 製品仕様オブジェクト $\mathcal{Q}$ に含まれる部品オブジェクトで部品オブジェクト x に性質の近い[4]部品オブジェクト y を取り出す.

STEP 4 部品オブジェクト x の属するクラス $\mathcal{X}$ と部品オブジェクト y の属する $\mathcal{Y}$ との共通の上位クラス $\mathcal{Z}$ を取り出す.

STEP 5 状態遷移 $\{\sigma_1, \sigma_2,...\}$ をクラス $\mathcal{Z}$ での状態遷移 $\{\Sigma_1, \Sigma_2,...\}$ に変換する.

STEP 6 状態遷移 $\{\Sigma_1, \Sigma_2,...\}$ を部品オブジェクト y の属するクラス $\mathcal{Y}$ での状態遷移 $\{\rho_1, \rho_2,...\}$ に変換する.

STEP 7 製品仕様オブジェクト $\mathcal{Q}$ において状態遷移 $\{\rho_1, \rho_2,..\}$ を起こす操作列 $[\pi_1, \pi_2, \pi_3,...]$ を取り出す.

上記に示した手順を図2.6と図2.7とを用いて説明する. 図2.6に部品オブジェクト x と部品オブジェクト y が属する部品オブジェクトのクラス階層を示す. 部品オブジェクト x はクラス X に, 部品オブジェクト y はクラス Y に属している. ここで, クラス Z は状態属性 $s = \{s_1, s_2, s_3\}$ を, クラス X は状態属性 s_1 が詳細化された 状態属性 $s.s_1 = \{t_1, t_2\}$ を, クラス Y は状態属性 $s2$ が詳細化された状態属性 $s.s_2 = \{u_1, u_2, u_3\}$ を持つ.

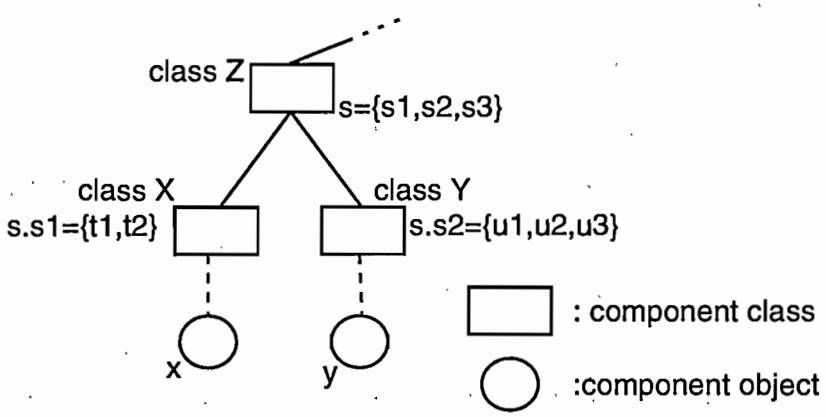


図 2.6: 部品オブジェクトのクラス階層の例

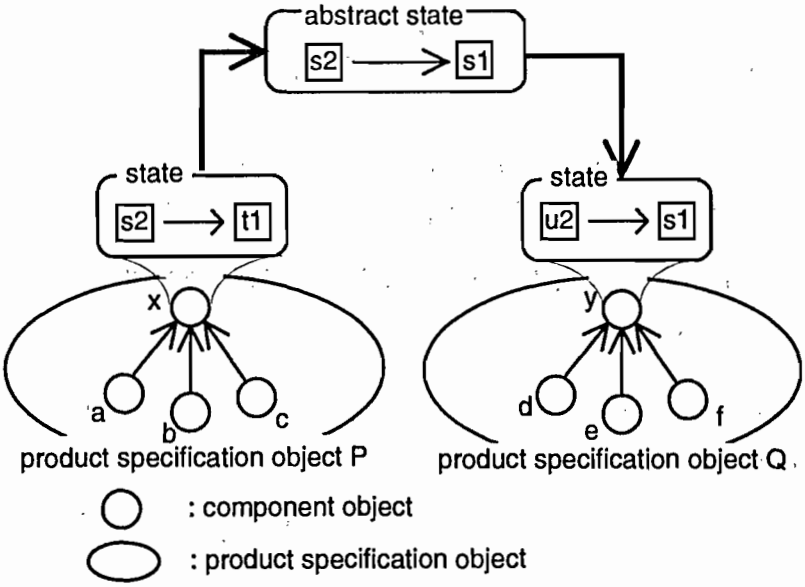


図 2.7: 操作に基づく検索

図2.7の製品仕様オブジェクト P は部品オブジェクト a, b, c, x で，製品仕様オブジェクト Q は部品オブジェクト d, e, f, y で構成されている．これらの部品オブジェクトは図2.6に示す部品オブジェクトのクラス階層に属するオブジェクト (インスタンス) である．製品仕様オブジェクト P に対する操作列 $[a, b, c]$ を入力とする．この操作列 $[a, b, c]$ は部品オブジェクト a, b, c を操作することに対応している．この操作により部品オブジェクト x だけが状態変化を起こすとする．

まず，操作列 $[a, b, c]$ により，製品 P に含まれる部品オブジェクトで状態変化を起こす部品オブジェクト x を取り出す **(STEP 1)**.

次に，操作列 $[a, b, c]$ を実行し，部品オブジェクト x の状態遷移 $\{s_2, t_1\}$ を取り出す **(STEP 2)**.

さらに，製品仕様オブジェクト Q に含まれる部品オブジェクトの中で部品オブジェクト x に最も性質が近い部品オブジェクトを取り出す **(STEP 3)**. この時,「性質が近い部品オブジェクト」にはいくつかの定義が可能であるが，ここでは「属するクラスの位置関係が近いオブジェクト」と定義する．この定義に基づくと以下の戦略が考えられる．これは，まず同じクラス，次にそのすぐ上位あるいは下位のクラスという順序で部品オブジェクトを検索するものである．図2.7の例では以下の優先順位での3. により部品オブジェクト y が選択される．

[4]「性質の近い部品オブジェクト」には様々な定義が可能であるが，この定義については後で述べる．

1. 製品仕様オブジェクトQに含まれる部品オブジェクトで部品オブジェクトxの属するクラスXに属する部品オブジェクト
2. 製品仕様オブジェクトQに含まれる部品オブジェクトで部品オブジェクトxの属するクラスXの一階層上位か，あるいは一階層下位のクラスに属する部品オブジェクト
3. 前手順(2.)の一階層下位のクラスに属する部品オブジェクト

さらに，部品オブジェクトxの属するクラスXと部品オブジェクトyの属するクラスYとの共通の上位クラスを求める(**STEP 4**)．この上位クラスにはクラスXと クラスYの共通の親クラスで，そのうち最も距離[5] が近いものが選ばれる．ここではクラスZ が選ばれる．

さらに，部品オブジェクトxが属するクラスXでの状態遷移$\{s_2, t_1\}$がクラスZでどのような汎化状態属性の変化に対応するかを求める(**STEP 5**)．s_2 はクラスZの状態としてあるため抽象状態もs_2である．一方，t_1はクラスZの状態として存在しない．ここで，t_1はs_1を詳細化した状態であるので，t_1はs_1 に対応することになる．従って，状態属性$\{s_2, t_1\}$はクラスZの状態属性$\{s_2, s_1\}$に変換される．

さらに，クラスZの状態属性$\{s_2, s_1\}$がクラスYでのどのような状態属性に対応するかを求める(**STEP 6**)．s_1はクラスYで存在するので詳細化された状態属性もs_1である．s_2はクラスYの状態属性としてs_2, u_1, u_2, あるいはu_3に対応する．この時，製品仕様オブジェクトQに含まれる部品オブジェクトの中で，これらの状態から他の状態へ変化させる部品オブジェクトを取り出す(**STEP 7**)．複数の部品オブジェクトが検索された場合，(STEP 3)で用いた優先順位をもとに部品オブジェクトが選択される．ここでは状態遷移$\{u_2, s_1\}$，部品オブジェクトd, e, fが求まる．ここで，選択された部品オブジェクトd, e, fは操作列$[d, e, f]$に対応している．以上の手順により操作列$[d, e, f]$が求まる．

2.5 結言

本章では，ソフトウェア開発におけるプロトタイピングためのオブジェクトモデルについて論じた．我々のアプローチは，製品の動作に関する仕様を製品仕様オブジェクトと部品オブジェクトによって定義するものである．また，リリース手法を提供することで，仕様の進化に柔軟に対応することが可能になった．本モデルの特長は，クラス階層と大量のインスタンスオブジェクトとの整合をとることが可能な点である．また，操作に基づいて製品仕様を検索する方法について述べた．

[5]上位クラスと下位クラスのパスを距離として，クラス間の最短パスを距離と定義する(部品オブジェクトのクラス階層は木構造である)．

今後の課題として，リリース手法の検証，設計プロセスに対応したツールのカスタマイズ機能，長時間トランザクション機能，ユーザごとのビューの提供などがあげられる．

§ 3

家電機器インタラクション・デザインシステム

3.1 緒言

マイコンを組込んだ家電機器の機能が増大するのに伴って，操作性の向上が大きな課題となっている．家電機器の操作性の向上は，単に「操作をわかりやすく」するだけでなく,「わかりやすさ」を通して得られる「操作する楽しさ」,「安心感」なども要求される．例えば，高齢者に対する配慮や，用語の統一，ボタンの形状，色彩，報知音などに対する配慮などである [26].

家電機器の操作は，製品コンセプトの企画者，インダストリアルデザイナ，機械分野のエンジニア，電気分野のエンジニア，ソフトウェアエンジニアなどの手によって設計され，試作品を用いて評価される．このような家電機器の操作や表示などのユーザインタフェースの設計は，ユーザと機器のインタラクションを扱うのでインタラクションデザインと呼ばれている [31]．インタラクションデザインには，表示画面のGUIに関するソフトウェアデザインと，ボタン・スイッチなどの専用機器，個別表示を使った物理的なパーツの操作を扱うSUIのデザインがあり，操作だけではなく，視認性や表示の意匠，状態遷移もデザインの対象となる [27]。

家電機器メーカでは，企画や営業の意見などを基に製品コンセプトを決定し，製品コンセプトに基づいて，インダストリアルデザイナや各分野のエンジニアがそれぞれの分野の設計を行なう．この過程で，さまざまなコミュニケーションが行なわれ，互いに関連する項目について調整しながら作業を進める．しかし，それぞれの分野には固有の考え方，用語などがあり，また他の分野の詳細までを深く理解することが難しいため，誤解が生じることが多い．このため，操作パネルを何度か試作し，製品コンセプトとの整合性や，操作性などを確認する．

操作性においては,「操作を最小にする」といった最適性だけでなく，直観的に理解出来ること

や,「覚えやすい」,「使っていて疲れない」などの快適性も求められる．このような要求に対しては，実際に操作してみた評価のフィードバックが不可欠である．それも，設計の早期から，繰り返して行なうことが望まれる．操作性の確認には，モニタユーザに評価してもらうことも多い．しかしながら，操作パネルの試作は時間と費用がかかるので，せいぜい数例の評価しかできないのが通例である．そのため，ユーザインタフェースの設計完成度を向上させるために，操作パネルの試作，モニタ評価，修正のサイクルを短くすることが大きな課題となっている．

最近では，家電機器組込みマイコンの能力が急速に進歩し，ソフトウェアで機能を実現する割合いが大きくなっている．これに伴い，製品の開発に占めるソフトウェア開発の比重が高くなっており，ソフトウェアの生産性や品質もこれまでに増して重要となってきている．また，製品の仕様変更がソフトウェアの仕様に及ぼす影響も大きくなっており，仕様を早期に確定することのメリットは非常に大きいものとなっている．このような背景から，これからのインタラクションデザインの支援には，

1. 可視性
2. 操作性
3. SUIのサポート
4. 機器分野に特化した部品ライブラリ
5. 機器組込みプログラムへの反映

などの項目を重視したシステムが望まれる．

ユーザインタフェースのプロトタイピングのツールには，パソコンで稼働する汎用のstoryboard(紙芝居)形式のものと，対象となる機器分野に特化したものとがある．

前者に，インダストリアルデザイナなどがよく用いているMacintoshのHyperCardやMacromind Directorなどがある．簡単なボタンなどが部品として用意されており，決められた操作の大筋を説明することに向いているが，操作仕様の変更や修正を行なうには時間を要する．

また，後者にはTrillium[21]，CISP[41]などがある．Trilliumは操作パネルのデザインや振舞いを*functioning frame*と呼ばれる単位でシミュレーションすることが可能なソフトウェア開発者用のツールで，コピアやプリンタのインタフェースのプロトタイピングに用いられた．

CISPはHyperCardの拡張であり，VTRなどの機器分野に特化したオブジェクトを積極的に導入している．また，ユーザに操作させて，そのイベントを記録する仕組みを持っているが，機器組込みプログラムの仕様にその結果を直接反映する仕組みを持っていない．また，どちらも操作パネルのデザインと振舞いを本体の機能から切離しているため，操作パネルのみのシミュレーションになっている．

この他に，ユーザインタフェースを含めたプログラミングの効率化を目的とするシステムにIntelligentPad[38]やHi-Visual [23]がある．また，要求仕様の検証・データベース化・実行，お

よび要求仕様からの設計支援を目的とするシステムに CARD [33] がある．いずれも対象はコンピュータ上のアプリケーションプログラムであり，機器のユーザインタフェースや機器組込みマイコンのプログラムを扱っていない．

本章では，このような既存のインタラクションデザインシステムにおける問題点の考察，および上記5項目を満たす家電機器インタラクションデザイン支援システム *Visual CASE*の基本構想とそのインプリメントについて述べる．また，実際の家電製品の開発に適用した結果を踏まえて，システムの評価についても述べる．

以下，本章の構成を示す．まず，3.2節では家電機器のインタラクションデザインについて説明し，インタラクションデザインモデルとして 機能操作制約モデル (FICモデル) を提案する．3.3節では *Visual CASE*の構成と，家電製品の開発への適用とその結果の評価について述べる．また，3.4 節では家電ソフトウェアの機能の部品化について述べる．さらに，3.5節ではまとめと今後の課題について述べる．

3.2 家電機器のインタラクションデザイン

3.2.1 家電機器ソフトウェアの開発とインタラクションデザイン

家電機器のインタラクションデザインが，コンピュータソフトウェアのユーザインタフェースデザインと大きく異なる点は4つある．

1. 操作のための入出力装置
   コンピュータでは操作を指示するための入力装置として，キーボードやマウスなどを用いる．GUIを用いる場合でも，画面上のボタンは実際にはポインティングデバイスとしてのマウスを操作することで行なわれる．家電機器の場合は，ボタンそのものが物理的な装置であり，直観的に操作方法を理解できることが求められる．これはGUI に対して，SUIと呼ばれる．出力を表示する表示装置も現時点では大きな違いがある．一般に，コンピュータでは，数百×数百の画素を持つビットマップディスプレイが使われるのに対し，家電機器では，表示する文字や図形を限定した蛍光表示管などを用いている場合が多い．

2. ハードウェアエンジニアやインダストリアルデザイナなどの参画
   コンピュータでは，ハードウェアが仮想化されており，主にソフトウェアエンジニアがインタラクションデザインを行う．これに対し家電機器では，製品の仕様がソフトウェアとハードウェアが密接に関係して実現されるので，ハードウェアエンジニア，インダストリアルデザイナ，あるいは営業や企画職能のメンバーも含めて製品の仕様を確認する場が必要になる．

3. 機器に対するユーザの知識

   コンピュータソフトウェアの場合は，ユーザに対してハードウェアやソフトウェアの知識をある程度期待できるのに対し，家電機器の場合は，ユーザに機器に対する知識をあまり期待できない．したがって，誤操作に対する安全性などもメーカ側でかなりの部分を保証する必要がある．また子供や高齢者が理解できるなどの配慮も必要である．

4. 機種展開

   家電機器の製品は，基本モデルの機能や性能にバリエーションをつけて，機種展開されるのが通例である．輸出用の製品では，地域性に合わせて，数十種類の機種が同時に設計される場合もある．

本節では，家電機器ソフトウェアの開発とインタラクションデザインとの関係について説明し，上述した4項目を考慮した機能操作制約モデル (FICモデル) を用いたデザインプロセスについて述べる．

コンピュータソフトウェアのプロトタイピングについては様々な取組みがなされており，[24]でその効用が述べられている．一方，家電機器ソフトウェアの開発におけるプロトタイピングについては，これまであまり多くの取組みがなされていなかった．しかし，ソフトウェアが家電機器の機能に占める割合いが大きくなっており，インタラクションデザイン支援システムが，製品の機能仕様や操作仕様などの設計・確認だけでなく，設計した製品の仕様を反映した家電機器ソフトウェアのプロトタイプとしても機能することが求められている．ハードウェアエンジニアやインダストリアルデザイナからみれば，単なるボタン1個の追加であっても，ソフトウェアの修正が膨大になることも稀ではない．機能仕様や操作仕様の変更がソフトウェアに及ぼす影響を把握したうえでの変更の決定が望ましい．また，インタラクションデザイン支援システムが迅速に評価結果をフィードバックして，早期に仕様が確定されることのメリットはソフトウェアの品質と生産性に著しくあらわれる．

3.2.2 機能操作制約モデル (FICモデル) を用いたデザインプロセス

インタラクションデザインモデルとしては，ユーザインタフェースと主機能を分離したモデルが適している．ユーザインタフェースを変更する際に主機能になるべく影響を及ぼさないほうが，プロトタイピングの効率からも機器組込みプログラムコードを再利用するという観点からも好ましいためである． 機能とユーザインタフェースを分離するモデルのうち， GUIを構築するためのモデルとしては，SmalltalkのMVC[19]モデルがある． MVCモデルはウィンドウシステムのアプリケーションの部品化と再利用の効率を高めることに適している．しかしながら，ウィンドウシステム上のアプリケーションに主眼をおいているため，実世界との対応を表すには不十分で

ある．また，階層構造をもつ建築構造物などを表示するために，構造と表示を分離したモデルにPAC[20]がある．PACは実世界における構造の抽象化を，階層化されたマルチエージェントで実現するモデルである．しかしながら，PACモデルの主眼は表示と抽象化された物理構造との分離であり，機能を持つ対象の操作を考慮していない．これらに対応するために，操作可能な機能を持つ機器を対象とするインタラクションデザインモデルとして，機能操作制約モデル(FICモデル)を提案する．

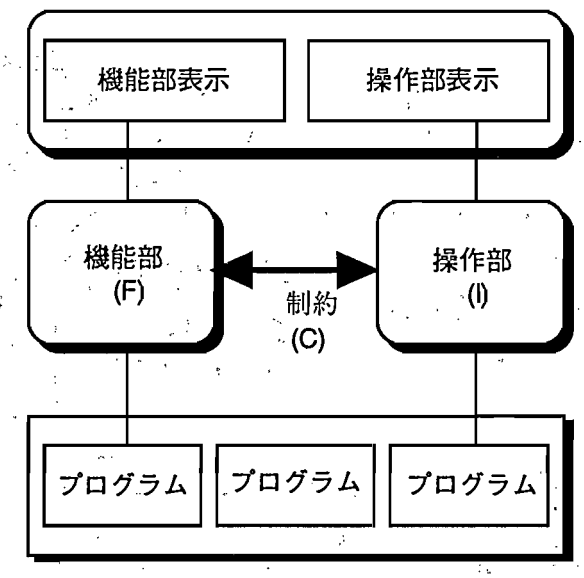


図 3.1: 機能操作制約モデル (FICモデル)

図3.1にFICモデルを示す．FICモデルは機器を操作部，機能部，および操作部と機能部の間の制約の3つの構成要素で表現される．

機器には主機能(これを単に機能部と呼ぶ)を駆動させるための操作があり，操作を行なうために操作部が装備されている．機能部から分離された操作部は，機能部を駆動させることができなければならない(これを操作可能性と呼ぶ)．操作部を変更する場合は操作可能性を必ず保持しなければならない．一方，機能部は操作部から必ず駆動されなければならない．すなわち，いずれの機能部もそれを駆動する操作部が存在しなければならない．

インタラクションデザインのプロトタイピングにおいては，柔軟性のある変更が要求される．そのためには，変更箇所以外の他の部分に影響を与えないことと，変更を反映した機器が即時に実行できることが必要である．この要件を機能部と操作部との制約によって実現する．ここで，操作部の機能部に対する操作可能性を保つための機構を制約と定義する．制約には，操作部と機能部との存在制約や組合せの制約などが考えられる．迅速なプロトタイピングのためには，このような制約は非常に重要である．*Visual CASE*はFICモデルを家電機器ソフトウェアに適用したオ

ブジェクトモデルに基づいて設計されている.

3.3 Visual CASE

本節では家電製品インタラクションデザイン支援システム *Visual CASE*の実現について述べる[36][40][35]. *Visual CASE*はSun OS 4.1 の Open Windows 2.0 とオブジェクト指向ソフトウェア開発環境ActivePage[30]上で稼働している.

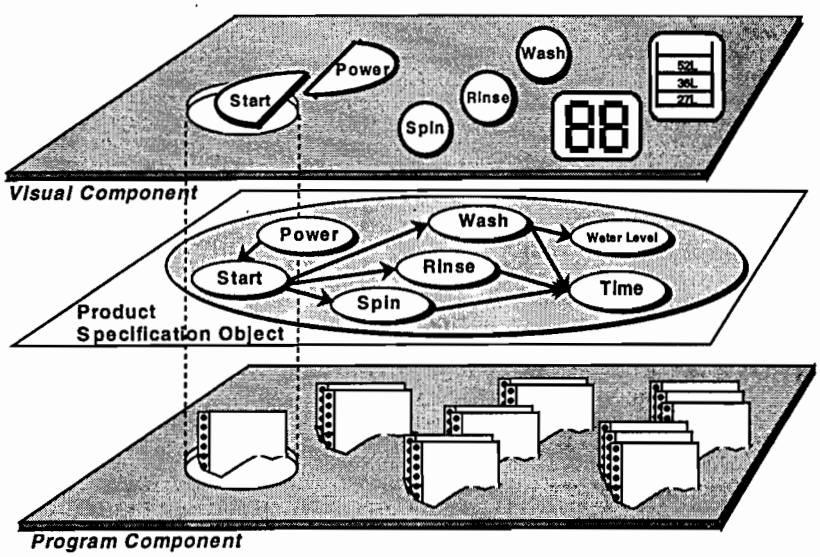


図 3.2: 部品の構成

3.3.1 Visual CASEにおける部品の構成

*Visual CASE*における部品は前章で述べたオブジェクトモデルを基に設計されており，以下の3つの部分から構成されている.

- ビジュアル部(PEL[30]で記述)
- 動作部(LAUSIVで記述)
- プログラム部(C言語，およびアセンブリ言語で記述)

動作部には，各部品の動作を記述言語LAUSIVで記述する．動作部は前章で述べたオブジェクトモデルの部品オブジェクトに対応し，この動作部の集合で製品全体の動作を定義する．ビジュアル部には，その部品の物理的な動作を画面上でのグラフィカルアニメーションとしてPELで記

述する．例えば,「ボタンが押された」,「ランプがつく」，および「LEDの表示が変わる」等の動作を記述する．プログラム部には，組込みマイコン用のプログラムをC言語，およびアセンブリ言語によって記述する．なお，部品は前章で述べた部品オブジェクトのクラス階層と同様の階層(以下，部品階層と呼ぶ)を持っている．図3.2に *Visual CASE* における部品の構成を示す．

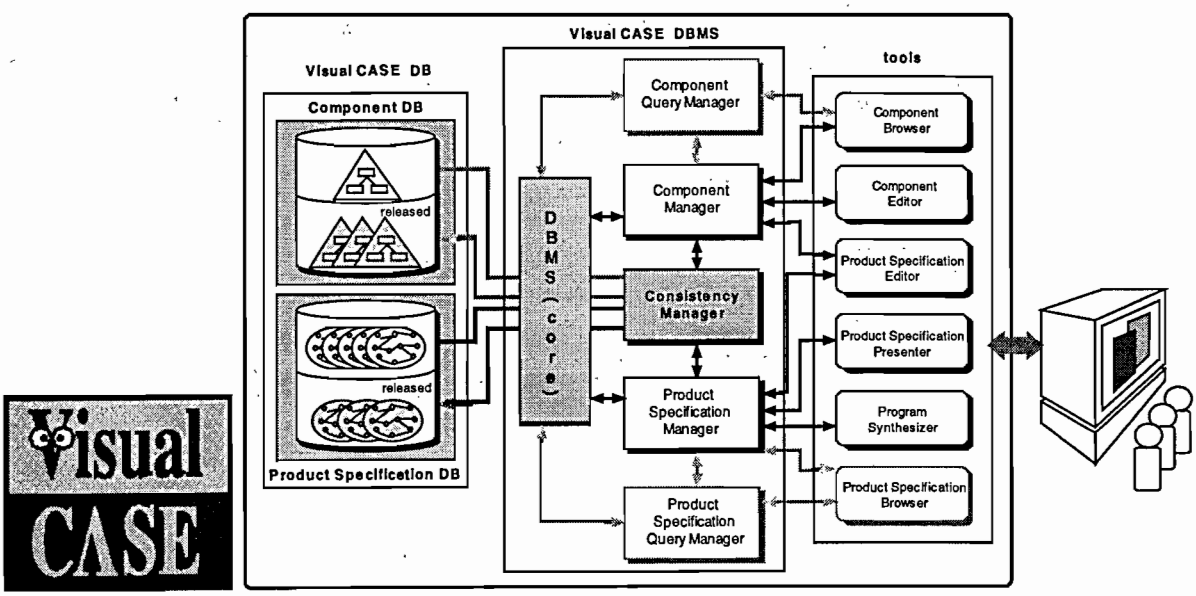


図 3.3: Visual CASE の構成図

3.3.2 Visual CASE の構成

図3.3に *Visual CASE* の構成図を示す．*Visual CASE* は製品の仕様をブラウズ，エディット，および格納する以下のツールから構成される．

製品仕様シミュレータ 製品仕様シミュレータは *LAUSIV* により記述されたメッセージ送受信を解釈する．ユーザはマウスなどにより画面上のボタンなどの部品をクリックすることで，機器操作のシミュレーションを行なう．これにより，ユーザは機器の動作を画面上で確認することができる．

製品仕様エディタ 製品仕様エディタは製品の仕様を作成・編集するツールである．製品の仕様を作成するために部品ブラウザから部品を追加・削除することで製品の仕様を編集する．なお，製品仕様エディタと製品仕様シミュレータはモード切替えにより，同一ツールで実現されている．

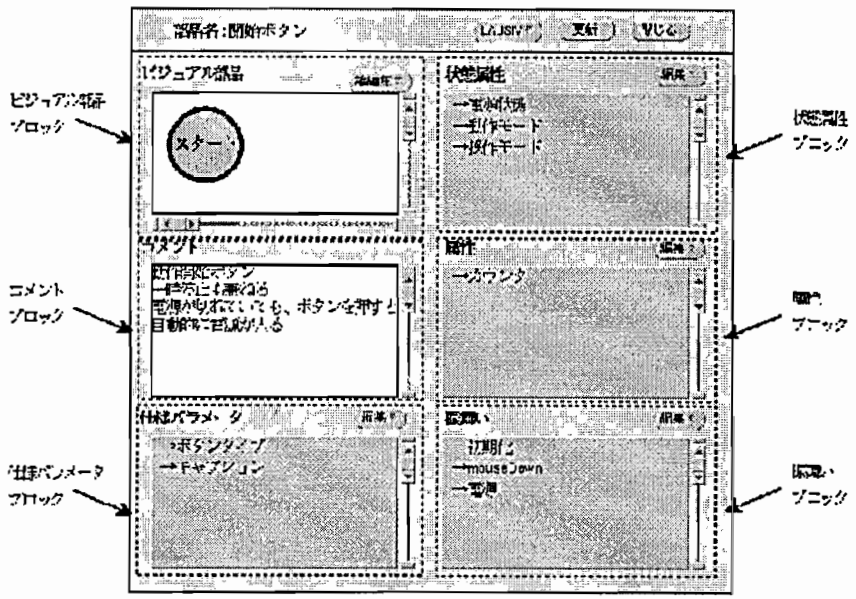


図 3.4: 部品エディタの画面イメージ

部品エディタ　部品エディタは新たに部品を作成したり，既存の部品の内容を変更するツールである．部品階層は機種展開にともなって追加・変更される．

部品ブラウザ　部品ブラウザは部品の内容の確認，および部品の検索を行なうツールである．また，部品階層の表示も行なう．図3.5に部品の階層を表示する部品ブラウザの画面イメージを示す．

プログラム骨格生成部　プログラム骨格生成部は動作記述言語*LAUSIV*で記述された部品間の動作部の振舞いと状態からプログラムのスケルトンを生成する．

図3.11に *Visual CASE*の画面例のイメージを示す．これは，全自動洗濯機の操作パネル設計の例である．図の上部が製品仕様エディタ，下部左側が部品階層を示す部品ブラウザ，下部右側が「コースボタン」クラスを示す部品エディタである．

3.3.3 Visual CASE を用いた家電製品インタラクションデザインプロセス

ここでは，*Visual CASE*の家電製品インタラクションデザインプロセスへの適用について述べる．手順の概要は以下の通りである．

1. 動作部品，およびビジュアル部品を作成し部品階層を構築する

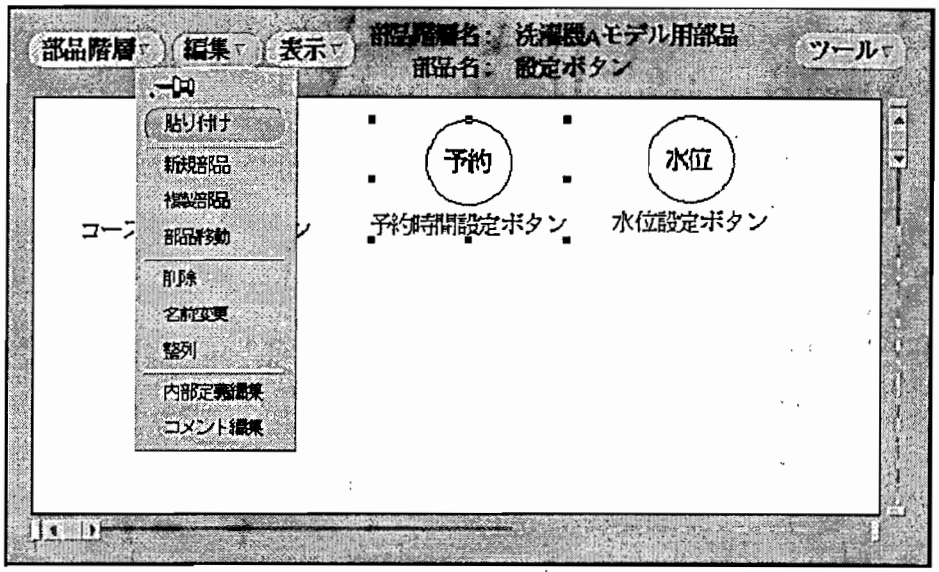


図 3.5: 部品ブラウザの画面イメージ

2. 部品階層から必要な部品を選択することで製品仕様を作成する

3. 製品仕様のビジュアルシミュレーションを行うことで仕様を確認する

以下，3.3.2節で述べたツールを用いてインタラクションデザインを行なう手順について述べる．

部品階層の構築

部品の構成要素の中で，インタラクションデザインのために必要な要素は動作部とビジュアル部である．部品の作成には部品エディタを用いる．図3.4にスタートボタン部品を作成中の部品エディタの画面イメージを示す．ビジュアル部品ブロックは，ビジュアル部作成のためのブロックである．このブロックの絵編集メニューを選択することで図3.7のパネルを起動し，形状・色・大きさの入力，およびPELによるグラフィカルアニメーション動作の記述を行なう．

一方，仕様パラメータブロック・状態属性ブロック・属性ブロック・振舞いブロックは，*LAUSIV*を記述するための動作部の入力領域である．コメントブロックは部品に対するコメントの入力領域である．部品エディタの LAUSIV メニューを選択すると*LAUSIV*記述の内容を表示するパネル(図3.6)が起動する．

製品仕様の作成

作成した部品階層から必要な部品を選択することで，製品仕様の作成を行なう．部品階層から部品を検索するためには部品ブラウザを用いる．図3.5に部品ブラウザの画面イメージを示す．こ

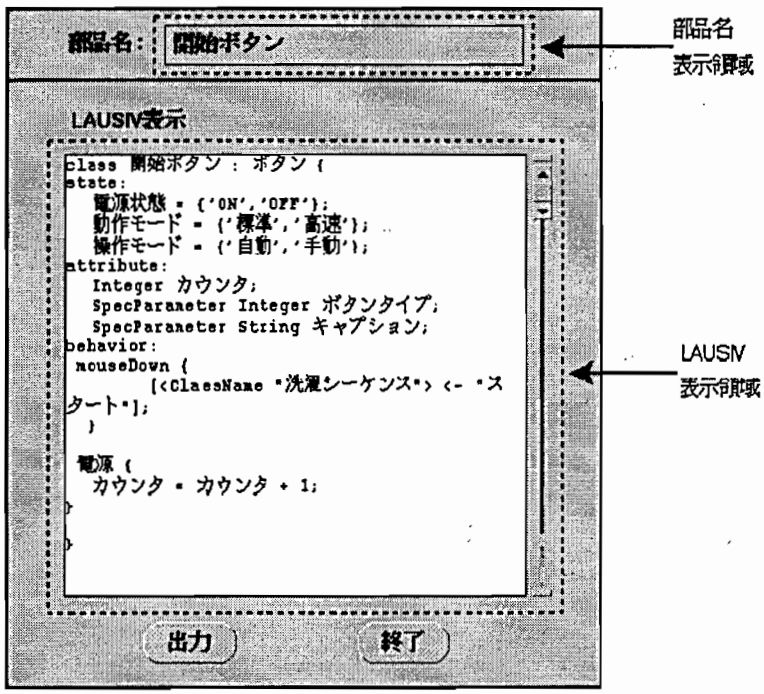


図 3.6: LAUSIV 記述の内容を表示するパネルの画面イメージ

の例では，設定ボタンクラスの下位クラスに3つのクラスがあることを示している．表示メニューを選択することで，部品階層のどの部分を表示するかを指定できる．また，図3.11の左下に示すように，部品階層をツリー表示するパネルを起動することも可能である．

選択した部品を製品仕様に追加するためには，部品ブラウザに表示される部品をクリックし，編集メニューの「貼り付け」を選択する．図3.5の例では，予約時間設定ボタンが選択され，製品仕様に追加された例を示している．この時，製品仕様エディタ (図3.11の上部) に予約時間設定ボタンのビジュアル部が取り込まれる．このビジュアル部をマウス等でドラッグして，適当な場所に配置する．この時，ビジュアル部と同時に動作部も自動的に製品仕様に取り込まれている．

製品仕様の確認

製品仕様エディタをシミュレーションモード (製品仕様シミュレータ) にし，ボタンなどのビジュアル部をマウスなどでクリックする．クリックされたビジュアル部は，各々の動作部にメッセージを送信する．動作部は各々の振舞いに応じて，他の動作部にメッセージを送信する．メッセージを受けとった動作部はその振舞いに応じて，さらに他の動作部にメッセージを送信するか，各々のビジュアル部に表示メッセージを送信するかを決定する．

図3.8にビジュアル部と動作部との動作摸式の例を示す．この例では以下のようにメッセージが

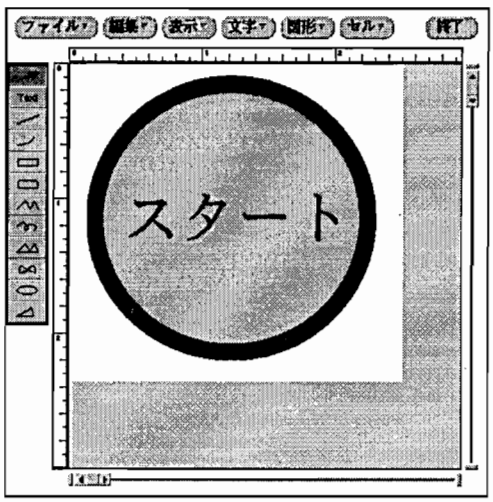


図 3.7: ビジュアル部編集パネルの画面イメージ

送信される.

(1) 画面上のボタンのビジュアル部がマウス等でクリックされると，"ButtonPressed" メッセージがボタンの動作部に送信される

(2) ボタンの動作部から" 点灯" メッセージが LED の動作部に送信される

(3) LED の動作部から"LightOn" メッセージが LED のビジュアル部に送信され，このビジュアル部は画面上で点灯する

ビジュアルシミュレーションによる製品仕様の検討

作成した製品仕様のビジュアルシミュレーションを行なうことで，操作仕様を検討する．画面上には数個の製品仕様を同時に表示し，他の製品仕様と比較検討を行なう．内部検討で集約された数種類の有力仕様案と過去の仕様を同一画面に表示し，モニター調査を行なう．この時，マウス操作に慣れていない一般ユーザのために，ワークステーションのディスプレイにタッチパネルを取りつけて，実際の操作パネルを操作する感覚でモニター調査を行なうことを可能とした．

3.3.4 実開発への適用と評価

全自動洗濯機の操作パネル仕様検討に *Visual CASE* を用いた．操作パネル設計での検討内容は以下の通りである．

1. 操作部，および表示部に関する検討
   入力デバイス (ボタンやスイッチなど) と表示デバイス (LED や表示管など) の選択，それら

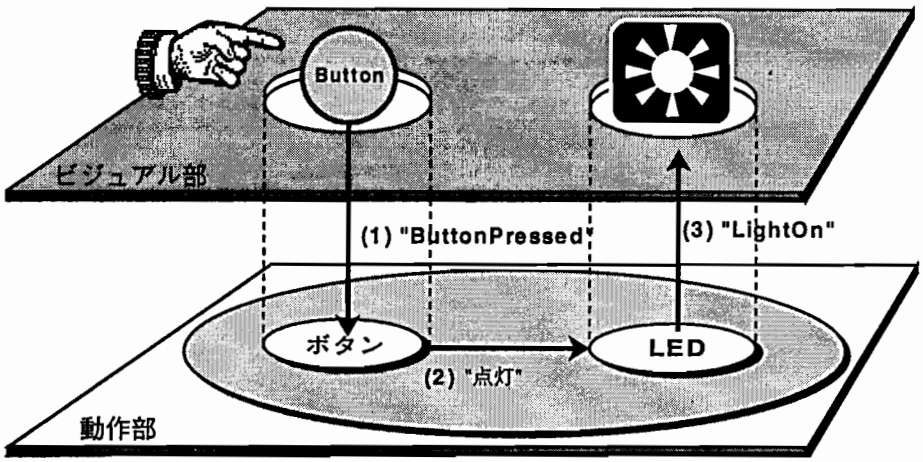


図 3.8: ビジュアル部と動作部との動作摸式

の大きさ・色・形状・配置等に関するデザインの検討

2. 操作仕様に関する検討
   洗濯コースや予約設定等の操作手順，ガイドメッセージや進行表示等の表示手順の検討

3. その他の検討
   背景色，印刷文字の内容や大きさ，品版やロゴの配置，操作部や表示部を区切る補助線の検討

図3.9に従来の操作パネル仕様設計と *Visual CASE* を適用した設計工程の比較図を示す．*Visual CASE* を適用した効果は以下の通りである．

- 操作仕様の早期確定
  企画会議終了時点で操作パネル仕様が確定した．これは，パネル試作の後に行なわれていたモニター評価が，仕様確定の早期段階で実現できたことによる．

- 設計品質の向上
  多数の操作仕様案(従来の約10倍)を作成することが可能になった．これにより，従来よりも，さらに設計完成度が向上した．

- マイコンプログラム開発効率の向上
  操作仕様が早期に確定したことにより，プログラム開発途中に仕様変更による手戻りが減少した．

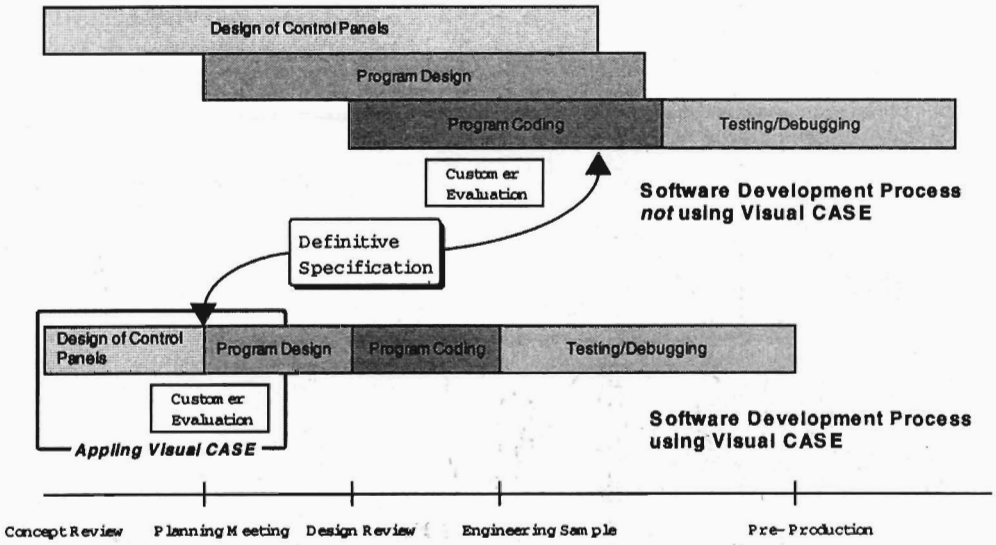


図 3.9: 従来の開発工程と Visual CASE を適用した開発工程

*Visual CASE*は電子レンジの操作パネル設計にも適用されており，同様の効果を確認している(図3.10).

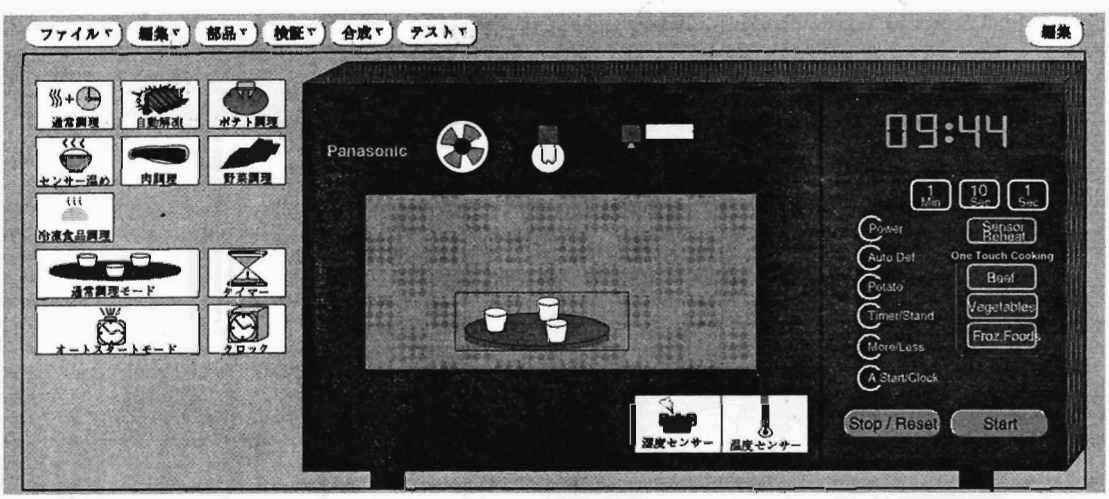


図 3.10: 電子レンジの操作パネル設計への適用

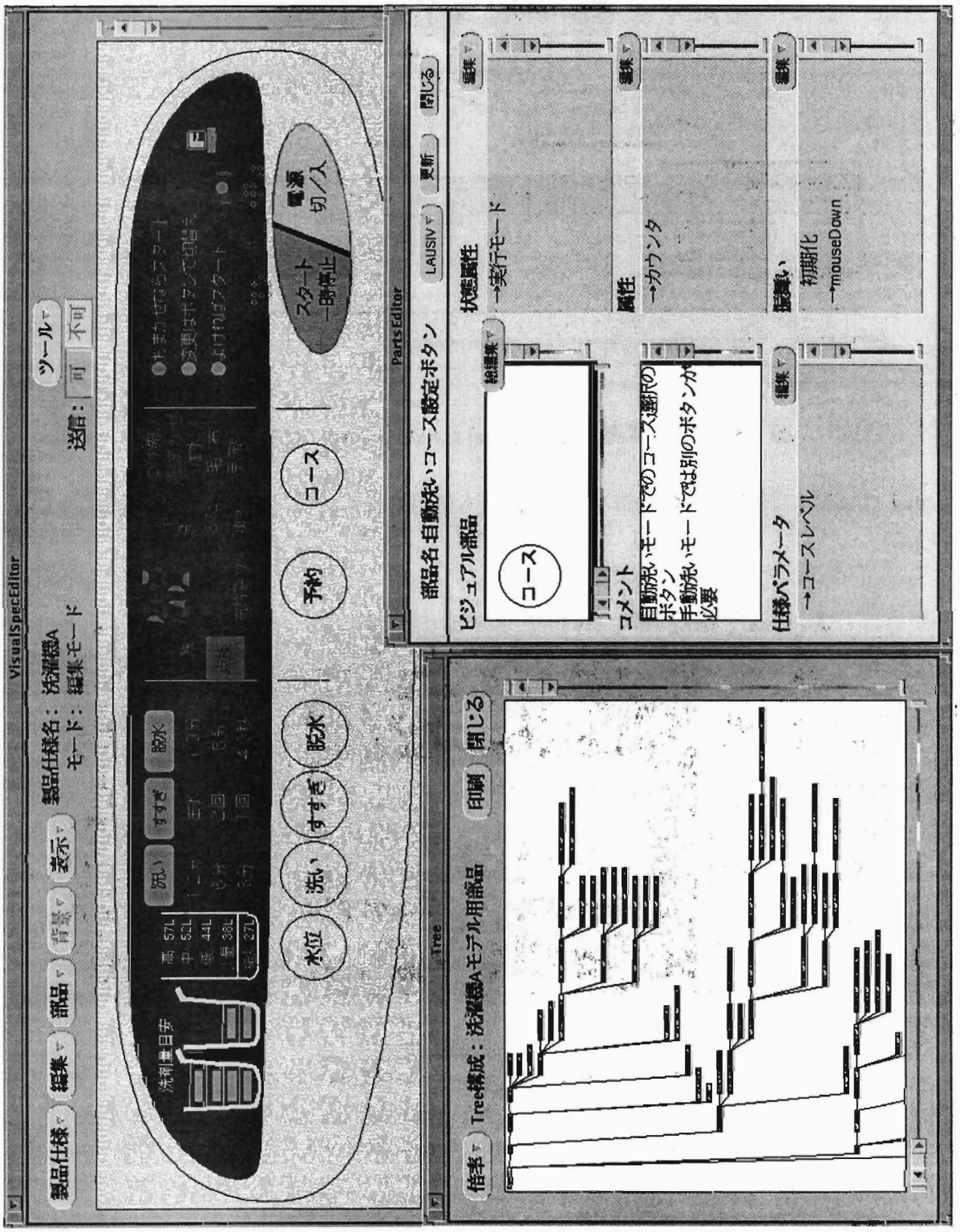


図 3.11: Visual CASE の画面イメージ

3.4 家電ソフトウェアの部品化手法

3.4.1 家電製品の仕様検討

現在，多くの家電製品には制御用のマイコンが組み込まれている．この制御用のマイコンのためのソフトウェアを家電ソフトウェアと呼ぶ．家電ソフトウェアが実現する操作仕様は，以下の特徴を持つ．

- 操作仕様の検討が多い
  家電製品の機能は年々高機能化しており，操作が複雑になっている．しかし，多くのユーザが使用する家電製品では，ユーザ毎に使いやすさの基準が異なる．このために，製品を設計する段階で，操作に関する多くのサンプル調査を行ない，検討する必要がある [41].

- 多くの機種を検討する
  家電製品は，価格帯や出荷地域により，ほとんど同じ機能を持つ製品でも，パラメータやボタンやLEDなどの操作を行なう対象が少しづつ異なる．このために，機種毎に操作仕様の検討を行なう必要があるので，多くの機種を検討する必要がある．

従来，製品の操作の検討をする際には，操作とその操作に対する製品の動作を記述した文章や図などの紙上の情報が用いられてきた．しかし，この様な情報では，製品の動きを十分に検討することができない．このため，製品の試作品を作成し，実際に操作を行い動作させることで，細かい操作仕様の検討を行なっていた．試作品の作成には多くの時間がかかるため，設計期間内に十分な検討を行なうことが困難であった．

操作仕様の検討内容を分析した結果，仕様の検討内容には偏りがあり，その内容により仕様の変更に要する作業量が変化することが解った．この結果を基に，変更が行なわれた際に，変更の如何に関わらず仕様の変更に要する作業量を少なくする部品を作成する手法を考案した．

分析結果

ある特定の製品に対する百数十回の仕様検討の内容を，仕様の変更の種類で分析した．この結果，仕様検討の内容を以下の様に分類することができた:

1. ボタンやLEDなどのハードウェアの変更
   操作や表示を行なうハードウェアの変更である．ユーザが直接，接する部分であるので，十分に検討を行なう必要がある．全体の約7割を占める．例として，次の様な変更が存在する．
   - ボタン形状の変更
   - 表示文字の変更

- LEDの色の変更
- ボタン位置の変更

2. 操作方式や表示方式などの方式の変更

操作を行なった時にどの様な順番で選択が行なわれるか，表示を行なう時にどのLEDを点滅させるかなどの変更である．全体の約2割を占める．例として，次の様な変更が存在する．

- 入力する時間の刻幅の変更
- 表示する文字パターンの変更
- コースを選択する順序の変更

3. 制御シーケンスの変更

製品の制御シーケンスの追加や，削除，シーケンスの内容の変更である．仕様検討の初期段階で決定する場合が多い．全体の約1割を占める．例として，次の様な変更が存在する．

- コース毎の初期値の変更
- 時計機能の削除

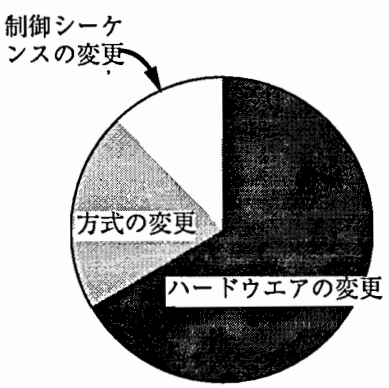


図 3.12: 仕様検討の分析結果

分類した仕様の変更に対して，*Visual CASE* での対応について述べる:

1. の変更に対して *Visual CASE* は，部品の色や形状を簡単に変更することができるので，少ない作業量で対応することが可能である．

2. の変更に対して，発生した現象について，例を用いて説明する．図3.13に示す部品でLEDの表示を行なっていたと仮定する．この時，LEDはボタンを一つ押す毎に，表示している数字を一つづつ減らす．

この製品に対して,「一桁の数字を表示する際には，十の位に0を表示する」という仕様の変更が発生したとする (仕様の変更を図3.14に示す).

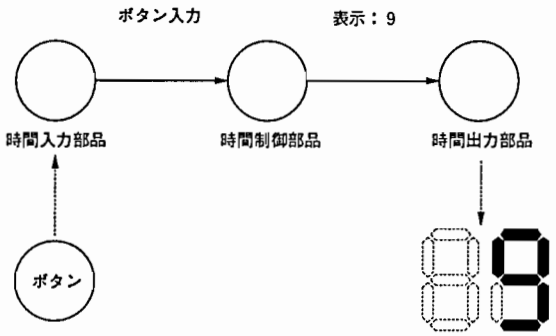


図 3.13: 部品と部品間を流れるイベント

図 3.14: 仕様の変更

この際，時間出力部品が数字を表示する機能は変更されていない．しかし，時間出力部品は数字の表示方式を最初から決めているので，表示方式を変更するために新しい出力部品を作成しなければならなかった．

3.の変更に対しては，新しい制御シーケンスが導入される度に，新規に部品を作成して対応した．これは，将来，導入される制御シーケンスを予測することが不可能なためである．本部品化手法は，この変更に対して直接サポートをしない．しかし，入力や出力の検討は，制御シーケンスの検討とは独立に行なうことができる．よって，制御シーケンスと入力や出力とを独立させ，インタフェースを決めることにより，新規の部品作成が容易になる．

3.4.2 部品化手法

部品の分類と設計

Visual CASE では，部品を入れ換えることにより仕様の変更を行なう．頻繁に発生する仕様の変更に対して，部品の再利用が可能ならば，早期の仕様検討が可能となる．一方，再利用できる部品が存在しなければ，新たに部品を作成しなければならない．部品の作成は入れ換えに比べて時間がかかるため，効率の良い仕様検討を行なうためには，入れ換えに適した部品を作成しなければならない．

部品を入れ換えて再利用するためには，以下の二つの条件を考慮して部品を設計する必要がある．

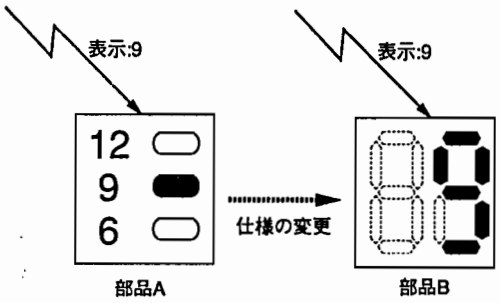


図 3.15: 表示部品の入れ換え

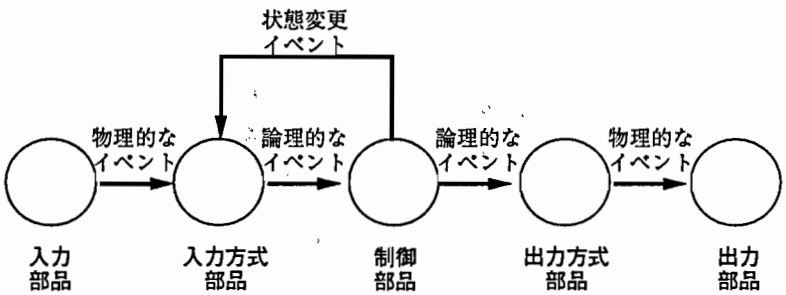


図 3.16: イベントの伝達方向

- 部品の粒度を仕様変更の単位に合わせる
  作成する部品の大きさは，形状や方式の変更などの仕様変更と同じ単位で一つの部品とする方が良い．なぜなら，部品の粒度が小さすぎると，変更の際に大量の部品を入れ換えなければならず，変更の影響範囲を特定することが難しくなるためである．一方，大きすぎると，入れ換えに適当な部品が無いために，部品を最初から再設計しなければならないからである．
- 同じ機能を持つ部品のインタフェースを揃える
  部品のインタフェースを共通化することにより，同じイベントを受けとる様に設計する．例えば，数値を表示する表示デバイスは「表示:(数字)」というイベントを受けとる様に設計する．こうすることで，図3.15に示す様に2つの部品の入れ換えが可能となる．

上述の分析結果と上記の条件を考慮し，再利用性の高い部品の設計を行なう．電源ボタンやコース設定ボタンなどの入力用ハードウェアは，位置や形状の変更を頻繁に受けるので，この単位で部品とする．この部品を，**入力部品**と呼ぶ．同様に，LEDなどの出力用ハードウェアは，位置や色や大きさの変更を受けるので，この単位で部品とする．この部品を，**出力部品**と呼ぶ．

予約機能や制御シーケンスは，その内容が変更されることは少ない．むしろ，機能やシーケン

ス自身が付加されるか，削除されるかが検討対象となる．各々の機能やシーケンスを部品とし，**制御部品**と呼ぶ．

操作方式や操作手順は，入力部品や制御部品の組合せが同じであっても変更されることがある．従って，方式や手順を機能やシーケンス等から独立させて部品とする．この部品を，**入力方式部品**と呼ぶ．同様に，表示方式も出力部品や制御部品の組合せが同じであっても，変更されることがある．従って，表示方式も機能やシーケンス等から独立させて部品とする．この部品を，**出力方式部品**と呼ぶ．

以下に，各部品の特徴とインタフェースについて述べる．

- 入力部品

  入力部品は，ユーザの操作より直接イベントを送る部品である．この部品は，操作されたことを検知し，入力方式部品に対してイベントを送信する．送信するイベントは，製品の状態の如何にかかわらず，操作をされたことだけを伝える．この様なイベントを，**物理的なイベント**と呼ぶ．例えば，電源の入／切を兼用するボタンは電源ボタンが押されたことのみを伝え，それが「電源入」なのか,「電源切」なのかを判断しない．

- 入力方式部品

  入力方式部品は，入力の内容を判定する部品である．この部品は，入力部品より物理的なイベントを受けとり，製品の状態よりその入力の内容を判断する．そして，適当なイベントに置き換えて制御部品に送信する．送信するイベントは，入力の内容を判定したイベントであるため，**論理的なイベント**と呼ぶ．また，入力方式部品は制御部品より状態変化のイベントを受けとることもある．例えば，水位入力方式部品は制御部品より設定可能な水位の情報を受けとる．これにより，水位の変更を行なうイベントが来た際に，現在選択可能な水位にのみ遷移する．

- 制御部品

  制御部品は，機器の状態遷移を管理する部品である．入力方式部品より状態遷移を開始するイベントを受けとり，必要な処理を行う．表示の変更が必要な場合は，出力方式部品に対してイベントを送信する．このイベントでは，ハードウェアの表示方法は考慮せず，表示を行なう内容を送信する．この様なイベントを，**論理的なイベント**と呼ぶ．最後に，状態の遷移結果を示すイベントを出力方式部品や入力方式部品に対して送信する．また，制御部品に含まれる他の部品に対しても送信を行なう．この様なイベントを，**状態変更イベント**と呼ぶ．

- 出力方式部品

  出力方式部品は，出力部品の表示方式を決定する部品である．制御部品より表示の変更を行

なうという論理的なイベントを受けとり，表示系に出力する際の出力方式を決定し，出力部品に表示の方法を示すイベントを送信する．送信するイベントは，直接LEDのON／OFFを行なうイベントである．この様なイベントを，物理的なイベントと呼ぶ．

- 出力部品
  出力部品は，出力方式部品より物理的なイベントを受けとりユーザに対して製品がどの様な状態にあるかを示す部品である．この部品では，その出力の方法や機器の状態などに関係なく，送られてくるONやOFFの命令を解釈して，出力用ハードウェアを制御する．

部品間のイベントの伝達方向を図3.16に示す．このように部品を分離し部品間を流れるイベントのインタフェースを決定することにより，各分類の内部で部品の入れ換えを実現することができる．

部品化手順

本章では前節で設計した部品を作成する手順を説明する．図3.17に示す操作パネルの一部の仕様検討を例に使用する．このボタンとLEDは，予約時間の設定を行う．LEDは，ボタンが押される度に図3.18の様に変化する．

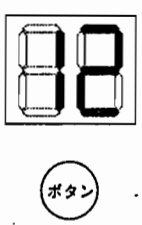


図 3.17: 操作パネルの一部

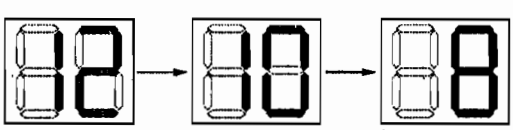


図 3.18: LEDの変化

入出力部品の作成

1. 部品の抽出
   入力部品は実際にユーザが操作を行う入力用ハードウェアと一対一に対応する．例の場合，

入力を受けつけるものはボタンである．このボタンを入力部品とし，時間入力部品と名付ける．出力部品は表示を行うハードウェアに対応する．例の場合，LEDが出力部品の対象となるが，部品の大きさが問題となる．LED一つ一つを部品として定義することもできるが，この例ではLEDの集合が数字を表すと考え，LEDの集合を出力部品として時間出力部品と名付ける．

2. インタフェースの設計

時間入力部品が受ける操作は，ボタンを押すことである．よって，この部品はボタンを押す度に，押されたことを伝える物理的なイベントを送信する．時間出力部品は，内部のLEDを点灯／消灯することで数字の表示を行なう．任意のLEDの点灯／消灯を行なうために，この部品はLEDの表示パターンを示す物理的なイベントを受信する．

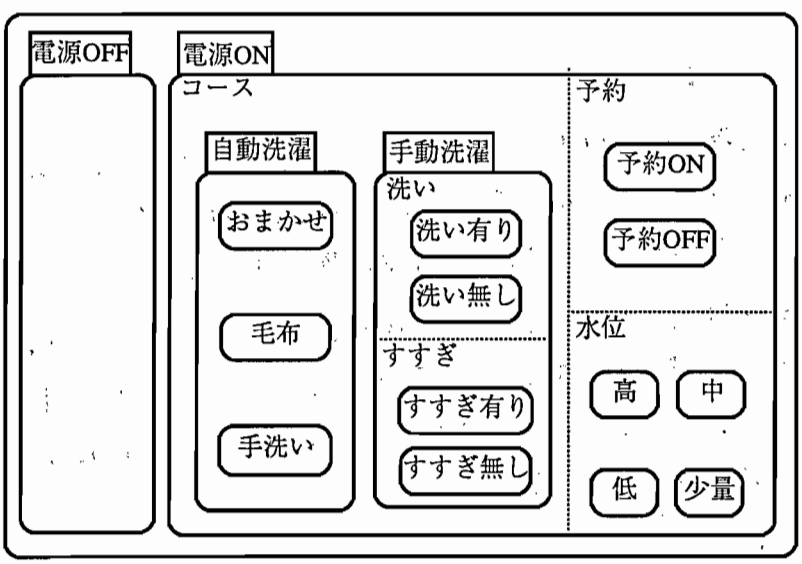


図 3.19: 状態の階層の例

制御部品の作成

1. 部品の抽出

制御部品は，機器全体の状態遷移を行なう部品である．制御部品の抽出には，機器の状態を階層化することで行なう．従来，機器の動作を表現する手法としては，状態遷移モデルを利用して表現することが有効であると考えられている．しかし，状態遷移を記述する際に状態遷移表を使用した場合，複雑な対象を表現すると状態数が増大し，記述が困難になる．そこで，機器の状態を階層的に表現する [39]．図3.17の機器が持つ状態を，階層を使って記述し

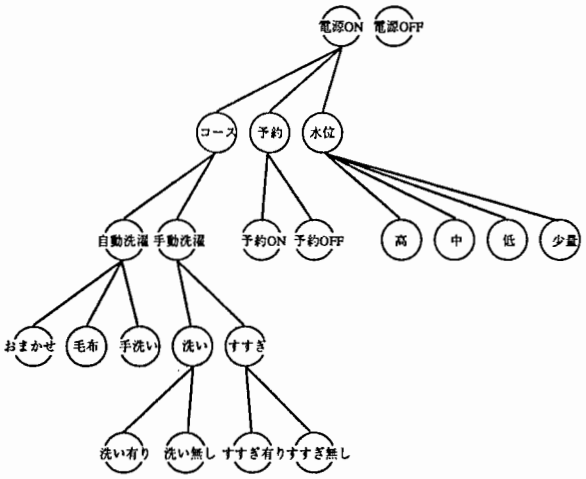


図 3.20: 制御部品の階層

たものを図3.19に示す.

この例では，電源のONとOFFの状態があり，電源ONの状態はコースの状態，予約の状態，水位の状態の３つの状態から構成される．本手法では，状態に対応する部品を作成し，作成した部品を状態の階層にしたがって部品階層を作成する．

例えば，図3.19の状態の階層から部品を作成する．まず，状態「電源ON」と「電源OFF」から２つの部品を作成する，次に，部品「電源ON」の下位に,「コース」「予約」「水位」の３つの部品を作成する．これを繰り返して，全ての状態を部品とする．図3.20に，この手順により作成した制御部品の階層を示す.

2. インタフェースの設計

制御部品は，入力方式部品より状態遷移を開始する論理的なイベントを受けとる．イベントを受けとった部品は，機器の状態とイベントより遷移を起こすかどうかを決定する．遷移を起こす場合は，必要な処理を行なった後に状態変更イベントを全ての入力方式部品，出力方式部品，制御部品に対して送信する.

この場合,「予約ON制御部品」と「予約OFF制御部品」の2つの制御部品が，時間の設定を要求する論理的なイベントを受ける．もし，機器の状態が「予約ON」状態ならば,「予約ON制御部品」がイベントを受けとり，処理を行った後，時間が設定されたことを全ての入力方式部品，出力方式部品，制御部品に対して送信する．また，機器の状態が「予約OFF」状態なら,「予約OFF制御部品」がイベントを受けとる．しかし,「予約OFF」状態なので，受信したイベントは無視される.

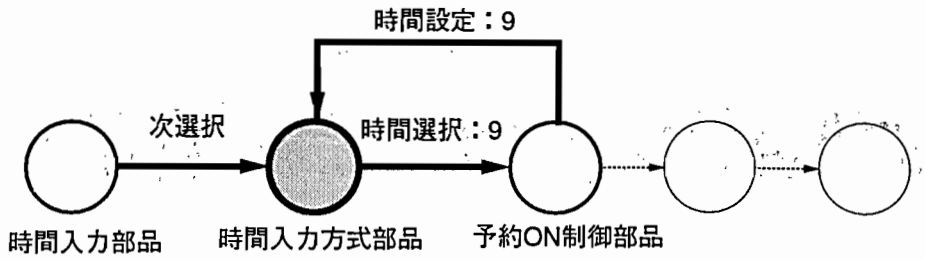


図 3.21: 時間入力方式部品のインタフェース

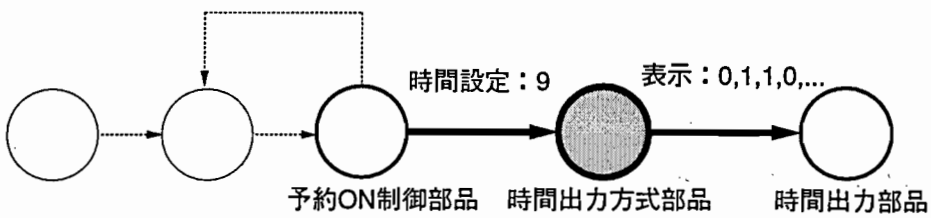


図 3.22: 時間出力方式部品のインタフェース

入出力方式部品の作成

1. 部品の抽出

   入力方式部品は，入力部品より送られるイベントの内容を判定する部品である．図3.17に示す例の場合，時間入力部品に対して時間入力方式部品を作成する．出力方式部品は，出力部品の表示方式を決定する部品である．図3.17に示す例の場合は，時間出力部品に対して，時間出力方式部品を作成する．なお，入力部品や出力部品が同じ組み合わせであっても，方式の変更を受けるので，入出力部品に対して方式部品は複数存在することがある．

2. インタフェースの設計

   入力方式部品より送信するイベントは，入力部品より送信される物理的なイベントの入力の内容を判定し，論理的なイベントに変換したものである．ボタン以外のダイヤルなどの入力部品を使用しても，入力部品が送信するイベントのインタフェースを合わせておく．これにより，いずれの入力部品に対しても入力方式部品を使用することが可能となる．入力方式部品は，入力部品からのイベント以外に，制御部品からの状態変更イベントを受信する．

   図3.21に，例の場合に作成する時間入力方式部品のインタフェースを示す．時間入力方式部品は，ボタンより「次選択」という物理的なイベントを受信し,「時間選択:9」という論理的なイベントを送信する．予約ON制御部品は，イベントを受け，時間を9に遷移するかどうかを判断する．その結果，予約ON制御部品は，遷移が可能ならば，時間が9に遷移したこ

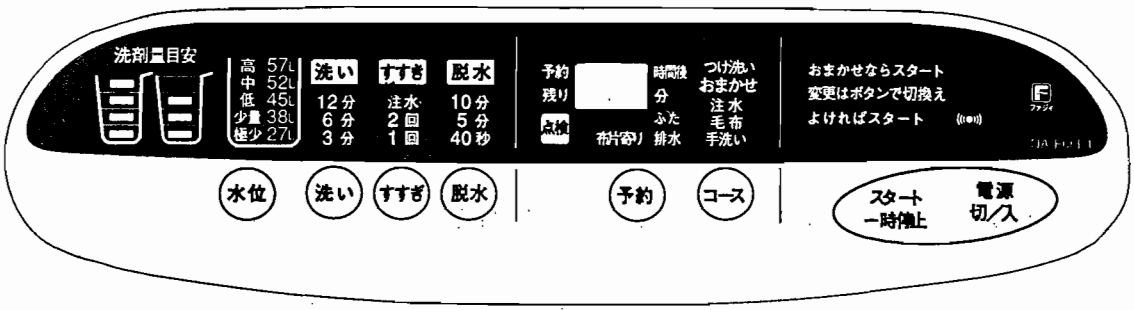


図 3.23: 全自動洗濯機の操作パネル

とを示す状態変化イベント「時間設定:9」を送信する．また，遷移が不可能ならば，受信したイベントを無視する．

出力方式部品は，制御部品が状態の変化を行なったことを受信する．そして，受信したイベントを解析し出力部品に対して表示の命令を送る．

図3.22に，図3.17に示す例の場合に作成する時間出力方式部品のインタフェースを示す．時間出力方式部品は，状態変化イベント「時間設定:9」を予約ON制御部品より受信する．つぎに，9を表示するLEDのパターンを作成し，時間出力部品に対して物理的なイベント「表示:0,1,1,0,...」を送信する．

適用例

この節では，本手法を洗濯機に適用した例を示す．図3.23は，全自動洗濯機の操作パネルを表している．

部品の分類	部品数
入力部品	14
制御部品	15
出力部品	14
合計	43

方式部品を使用しない部品化

部品の分類	部品数
入力部品	14
入力方式部品	8
制御部品	15
出力方式部品	14
出力部品	14
合計	65

本手法による部品化

図 3.24: 検討開始時の部品数

この洗濯機の仕様の部品化に対して本手法を適用した．比較のために，方式部品を使用しない

部品の分類	部品数
入力部品	2
制御部品	104
出力部品	24
合計	130

方式部品を使用しない部品化

部品の分類	部品数
入力部品	2
入力方式部品	5
制御部品	3
出力方式部品	9
出力部品	5
合計	24

本手法による部品化

図 3.25: 検討中に追加した部品数

方法で同じ製品の部品化を行なった.

仕様検討開始時に作成した部品の数を図3.24に示す. 次に, 約100の仕様検討を行なった後に, 追加した部品数を比較した. この結果を, 図3.25に示す.

本手法による部品化では, 方式部品を使用しない部品化に比べて検討中に作成する部品数が1/5以下となる. この要因としては, 方式部品を使用しない部品化では, 制御部品が操作・表示方式を実現する. このため, 入力方式の変更を行なう度に制御部品の作成が必要となる. また, 一つの方式が複数の制御部品で実現されている場合は, その影響範囲を特定することが困難になる.

ただし図3.24に示す様に, 本手法による部品化を適用すると, 方式部品を使用しない場合に比べて, 検討開始時に作成する部品の数が多くなる. しかし, これは検討中に追加する部品を作成するコストと比べてはるかに小さい.

3.5 結言

本論文では, 機器のインタラクションデザインモデルとしてFICモデルを提案し, FICモデルに基づく家電機器インタラクションデザイン支援システム *Visual CASE* の機能を示した. また, 家電ソフトウェアの機能の部品化手法を提案し, *Visual CASE* で支援することが可能であることを示した. また, 実際の家電製品の開発に適用し, システムの有効性を検証した.

最後に, *Visual CASE* の取組みを3.1節に示した家電機器インタラクションデザインの支援環境に求められる要件に沿ってまとめる.

1. 可視性

   製品フレーム, ボタン, 表示パネルなどを実際の製品に近い形と色で表現したビジュアル部の枠組みを用意し, 読む仕様書でなく「見る仕様書」として使えるようにした.

2. 操作性

部品として定義したボタンの動作，表示パネル，および機器の可動部の動作を表示することにより，実際の製品の操作感に近い感覚の「動く仕様書」として使えるようにした．また，ユーザ評価時に，マウスに抵抗のあるユーザにはワークステーションのディスプレイにタッチパネルを取りつけて，ダイレクトマニピュレーション感覚を確保するようにした．

3. SUIのサポート
   実際の製品では，ボタンやパネル表示は物理的に固定されたり，表示内容を限定した蛍光表示管を用いたりするものが多い．また，機器や分野専用の用語表示が必要である．このため，デザイナーの設計したデザイン図をベースとして，ビジュアル部を作成する仕組みを用意した．またビジュアル部の大きさは実寸値を基本とし，他のCADツールとの連携性を持たせた．

4. 製品分野に特化した部品ライブラリ
   家電機器の特性に合わせて，機種展開や従来機種の有効活用に重要となるハードウェアと機器組込みプログラムの再利用性を高めることに重点を置き，ボタンや表示パネル，および製品の可動部分などが簡単にカスタマイズできる部品の充実を図った．また，部品オブジェクトの作成効率や再利用性を高めるために，専用のオブジェクト指向言語を設計した．

5. 機器組込みプログラムへの反映
   インタラクションデザインの結果を組込みプログラムに反映しやすいように，仕様書およびプログラム部品を統一的に扱う枠組みを用意した．また，家電製品の進化に対応し，設計段階での機能追加などにも対応できるように，版の進化をサポートする機構を用意した．

*Visual CASE*はインタラクションデザインのみならず，プログラムの自動合成機能などにも対応できるようにシステムに拡張性を持たせている．今後の課題として，プログラム自動合成システムとの結合が挙げられる．

§ 4

放送データのための連続問い合わせと動的構造化

4.1 緒言

現在，テレビ放送においては地上波放送・ケーブルテレビに加え，デジタル衛星放送がサービスを開始している [58]．デジタル衛星放送では，MPEG2の規格に準拠し映像および音声を送信している [45]．従って，映像・音声圧縮技術と伝送技術により，限られた伝送帯域幅のもとで多数の番組を伝送することが可能である．すなわち，放送チャンネルとしては，数十から数百チャンネルによる番組が伝送可能となる．

本章では，デジタル衛星放送システムを用いて送信された大量データの構造化について述べる：

1. 4.2節，4.3節では，一定周期ごとに連続して送信されるテキストデータに対して連続的に問い合わせを行うことにより，ユーザの所望するデータを検索する方法について検討を行う．すなわち，デジタル衛星放送の番組情報として送信されている電子番組ガイド (EPG: Electronic Program Guide) [46] を対象データとし，数百チャンネルの番組ガイドに対する番組データの検索方法について議論を行う．

2. 4.4節では，デジタル放送における，インタラクティブ・データ配信のためのカルーセル方式を用いたDVXについて述べる．DVXの伝送方式は，今後のデジタル放送の標準となるMPEG2のハードウェア処理機能を活用したものとなっている．本稿では，DVX の伝送方式と，データベース連携の観点からのサービス展開の可能性について述べる．

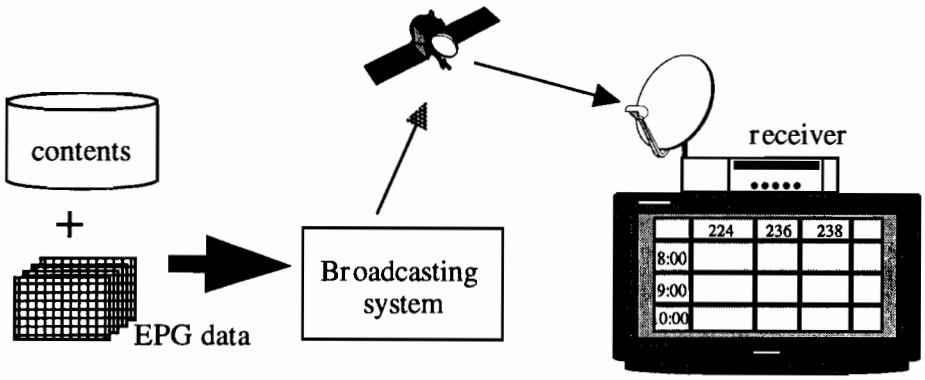


図 4.1: EPG サービス

4.2 バーチャルチャンネルの概要

番組情報は膨大なデータを含んでおり，かつ，更新される可能性を持つため，ユーザがすべての番組情報をチェックすることは不可能である．従って，すべての番組を各ユーザごとに定義されたビューにより，提示する機構が必要になる．

本研究では，システムが定義したEPGデータを，再構築する枠組みについて検討を行なう．この枠組みをバーチャル・チャンネル(VCH)と呼ぶ．VCHは以下の手順にしたがって処理が行なわれる．

- 新しいEPGデータが配送されるごとに検索を実行
- 検索結果は，その放送時間に従って時系列に配置
- それぞれの番組情報は，内容的に近いもの，あるいは関連する情報への動的リンクを生成

従来の蓄積情報しか扱えなかったハイパーメディアを拡張し，ライブ情報も扱うことが可能なライブハイパーメディアが提案されている[48]．このモデルでは，放送される情報に対してハイパーメディア技術を用い，多数のチャンネルを関係づけて視聴することが可能である．

[48] で提案されているモデルは，映像データを対象にしたものであり，画像を認識することでビデオストリームからオブジェクトを抽出するものである．一方，本方式はビデオストリームそのものではなく，関連情報に基づいたリンク生成である点で，このモデルとはアプローチを異にする．一般に，コンテンツそのものを自動認識する方式よりも，関連情報による分類・関連付けの方が情報の抽出精度は高くなる．また，[48]では，ライブデータを処理することについて議論されているが，時間依存データおよび時間経過に対するデータの管理については考慮されていない．

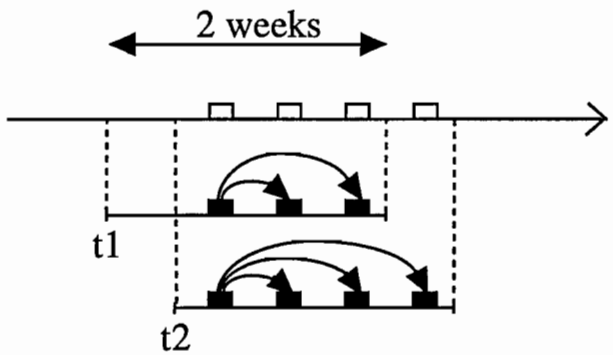


図 4.2: 連続的検索

4.2.1 EPG: 電子番組表

電子番組ガイド (EPG) は，放送内容に関する情報をサービス情報 (Service Infromation) として追加され，伝送される情報である [58]．従来，テレビ放送の番組情報は新聞，テレビ雑誌などの出版物で参照されるのが一般的であった．しかし，デジタル衛星放送では，数十から数百のチャンネルによる番組が提供されるため，ユーザの選択肢が増えると同時に，番組選択が極めて繁雑になる．EPG は，番組情報を電子的に放送データに付加して送信することでユーザの番組選択を支援するサービスである．

EPG に含まれる内容は，放送システムにより異なるが，一般に，チャンネル番号，番組名，ジャンル，放送時間，番組概要などである．番組情報は，放送中の情報に加えて，現在からある一定期間 (例えば，2 週間) の番組情報を提供する．従来の地上波放送などのように数チャンネルの場合は，新聞の番組表などで一覧することが可能であるが，多チャンネルになると膨大な情報から必要な情報を検索する枠組みが必要になる．

本研究では，データ放送サービスで伝送されてくる大量の放送データを，データサーバからの部分データの配送としてとらえ議論を行なう．大量のデータからユーザが必要な情報を検索するためには，ユーザごとのビューを提供することが有効であると考えられる．これを実現するために，ハイパーリンク構造を導入する．ハイパーリンクは，コンテンツのある部分 (アンカー:リンク元) と他の部分 (リンク先) にリンク関係を定義し，その間のナビゲーションを可能にするものである．この時，放送データの特性により，以下の課題がある：

- 放送データは永続的データではなく，時限データである
- 未配送データや削除 (expire) データがあるためにリンク構造が不完全になる

連続問い合わせと動的構造化 EPGデータは，時間が経過すると，経過したデータ(放送が終了したデータ)が削除され，新しいデータが追加される．

図4.2は，時間t_1と時間t_2にそれぞれ着信したEPGデータを示してる．時間t_2において新たにデータが追加されたことを示してる．この時，検索結果が変更されるたびに，動的リンクが生成される．

動的リンク 提案するリンク構造は，ハイパーリンク構造に基づくものである．ハイパーリンクは，静的リンクと動的リンクの2種類に大別される．静的リンクは，リンク元とリンク先が固定である．一方，動的リンクは条件によりリンク元あるいは先が動的に変更される．これまでの研究では，リンクを生成する際に，リンク元とリンク先を検索し動的にリンクを生成する方式が提案されている[68].

VCHでは検索により抽出された番組情報に対して2種類のリンクを扱う．一つは，番組属性から，関連する他の番組属性へのリンクである．例えば，複数の番組間に生成された同一出演者属性同士のリンクなどである．これらのリンクは意味的に同一，関連，派生などを表すため,「内容リンク」と呼ばれる．

もう一つは,「構造的リンク」である．検索により得られた番組情報は，優先順位情報を基に時系列にソートされ並び変えられる．ここで，再放送などのように同じ番組が何度も放送される場合，あるいはNVOD(Near video-on-demand)のように複数のチャンネルを使って，一つの番組を時間差で放送する場合がある．このような時，スケジューリングに関して，他の候補があることを示すために構造的リンクが用いられる．

4.2.2 想定する環境

想定する電子番組情報 想定するEPGデータによる番組情報は以下の属性から構成される．

1. program identifier (id)
2. series (list)
3. channel number (numeric)
4. title (character)
5. genre (code)
6. start_time (numeric)
7. program time (numeric)
8. summary (character)
9. cast (character list)
10. rate (numeric)

番組識別子は，番組を一意に識別可能な識別子である．シリーズは，ある番組シリーズに属する，他の番組のリストである．チャンネル番号は数値，番組名は文字列である．ジャンルは，{スポーツ，野球}などのように，主ジャンル，副ジャンルの組合せで表示されるのが一般的であるが，簡単化のために,「映画」,「ドラマ」,「スポーツ」などの数種類で定義する．放送開始日時は，番組放送の開始日時である．放送時間は，その番組の放送される時間である．番組概要は，あらすじなどの番組内容である．

出演者は，番組に出る出演者のリストである．視聴率は，現在のその番組の視聴率をリアルタイムに反映したもので0から100の数値とする[1]．ただし，現在，放送されている衛星放送サービスでは，出演者リストや視聴率はEPGデータに含まれていない．

EPGデータの内容については，以上にあげた文字列情報以外に，静止画，動画，音声などのサービスも考えられるが，ここでは，EPG データとして文字列情報のみを扱う．

EPG データは現在の時刻から2週間分のデータが配送される．例えば，現在の時刻が [Sep. 7 11:30] であるとすると $start$ が [Sep. 7 11:30] から [Sep. 21 11:30] までの全ての番組情報が配送される．

データ構造 議論のために，各番組をオブジェクト，各番組情報をオブジェクトの属性としてモデル化を行なう[2]．以下，番組をオブジェクトと呼ぶ．

オブジェクトは次のような組 $o(t) = (i, v(t), o_{sub})$ である．

- i はオブジェクト識別子 (oid) である．
- $v(t)$ は時刻 t の o の属するクラス C において定義されている属性の組 $[a_1 : v_1(t), \ldots, a_n : v_n(t)]$ である．
- o_{sub} はオブジェクトに含まれるサブオブジェクトのオブジェクト識別子の集合である．

クラス定義は，各番組情報に基づいて記述される．通常，ジャンルに応じてクラスを定義するのが一般的と考えられるが，本論文では特に規定しない．また，サブオブジェクトを含むオブジェクトをスーパーオブジェクトと呼ぶ．サブオブジェクトは，ニュース番組などで，一つの番組が構成要素として複数のニュースから構成されている場合，一つのニュースに相当する．

属性値については，固定ではなく時間依存であることに注意を要する[3]．

---

[1]視聴率の定義は多数考えられるが，ここでは特に規定しない．

[2]本論文では，番組情報をオブジェクトモデルを用いて定義しているが，関係モデルやその他のモデルを用いても構わない．

[3]すべての属性が時間依存である必要はない．例えば，番組名やジャンルなどは一般的には固定である．

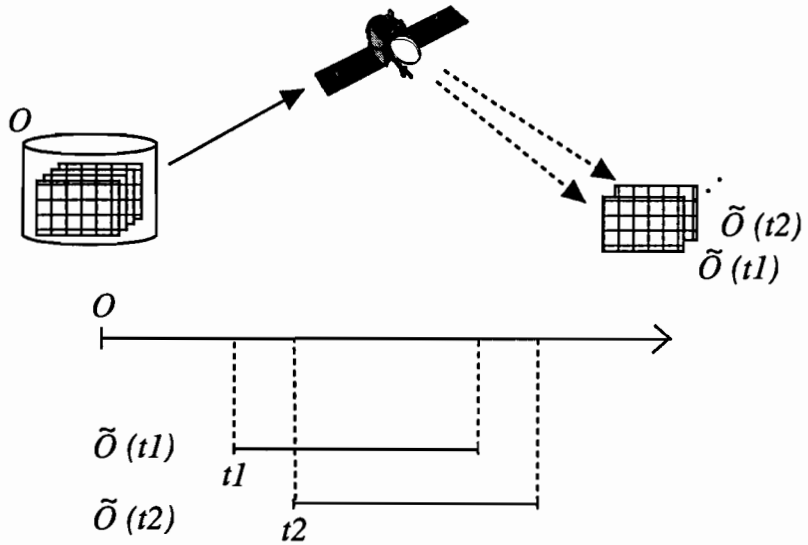


図 4.3: EPG 情報

サーバー(放送局)に存在する，すべてのオブジェクトを$\mathcal{O} = \{o_1, \ldots, o_n, \ldots\}$と表記する．$\mathcal{O}$には同じ属性値を持つオブジェクトが複数存在する場合(再放送など)があるが，これらは異なるoidを持つ別のオブジェクトと考える．

時刻tにおける，時刻tから一定期間(ここでは2週間)に放映される予定のオブジェクトは関数$broadcast(\mathcal{O}, t)$により定義され，$\tilde{\mathcal{O}}(t)$と表記される．$\tilde{\mathcal{O}}(t)$は時刻$t$でサーバにおいて配送されるすべてのEPGデータであると同時に，時刻tで端末において受信されるすべてのEPGデータである．図4.3にサーバと端末の間のデータ送信を示す．

4.3 Virtual Channel における問い合わせと構造化

4.3.1 質問言語

検索文の構文について述べる．この構文は，目的オブジェクト，変数の範囲，検索条件式の指定を，それぞれ，SELECT句，FROM句，WHERE句で指定するSQLを基にしている．ただし，便宜上この表記を基にしているだけであって，この表記に限定される必要はない．

検索文は，SELECT句，WHERE句，およびEXECUTE句からなる．このうち，SELECT句は必須であるが，WHERE句とEXECUTE句とは省略可能である．

SELECT句では，検索の目的オブジェクトが指定される．\ はサブオブジェクトが検索された場合は，サブオブジェクトの粒度で取り出される．直観的には，ニュース番組の中の一つのトッピクスをとり出すのか，あるいはニュース番組自体(全体)をとり出すのかの指定に当たる．WHERE

検索文の基本構文:

$$< \text{検索文} > ::= \quad < SELECT\text{句} > [< WHERE\text{句} >] [< EXECUTE\text{句} >]$$

$$\begin{aligned}
< SELECT\text{句} > ::= &\quad SELECT < \text{式} > \\
< \text{カテゴリー} > ::= &\quad < \text{値} > | < \text{集合} > \\
< WHERE\text{句} > ::= &\quad WHERE < \text{探索条件} > \\
< \text{探索条件} > ::= &\quad < \text{式} > [< \text{時間条件} >] \\
< \text{時間条件} > ::= &\quad WHEN \ < \text{値} > \ | \ LAST < \text{値} > \\
< EXECUTE\text{句} > ::= &\quad EXECUTE < now > \ | \ < continuously >
\end{aligned}$$

句では，探索条件を記述する．この時，時間条件付きの式を指定することも可能である．時間条件には，WHENと LASTが指定可能である．WHENはその式の値を絶対時間で指定する．また，LASTは検索文を実行する時間からの相対時間で指定する．

実行条件では，検索文を一度だけ実行するのか，あるいは連続して実行するのかを指定する．continuously を指定した場合，その検索文は常時実行される．

Example 1: 検索式の例を示す．以下は，「無料チャンネルで放映される番組で，タイトルが『The X Files』というオブジェクトをとりだせ」という検索を記述したのものである．

$$\begin{aligned}
SELECT: &\quad \$X \\
WHERE: &\quad (\$X.genre = Free \ AND \\
&\quad \$X.title = "The\,X\,Files")
\end{aligned}$$

Example 2: 次に，粒度条件に関する例を示す．以下は，「有料チャンネルで放映される番組で，ジャンルが"News" で，その要約に"Hong Kong" を含むオブジェクトを検索せよ．ただし，サブオブジェクトの場合はサブオブジェクトの単位で取り出せ」という検索を示している．

$$\begin{aligned}
SELECT: &\quad \backslash\$X \\
WHERE: &\quad (\$X.genre = \text{“}News\text{”}
\end{aligned}$$

$$AND\ \$X.summary \ni \text{“}HongKong\text{”})$$

以下は,「有料チャンネルで放映される番組で，ジャンルが"News" で，その要約に"Hong Kong" を含むオブジェクトを検索せよ．ただし，サブオブジェクトの一つ上の階層のすべてのオブジェクトを取り出せ」という検索を示している．

$$\begin{aligned} SELECT: &\quad \$X/../* \\ WHERE: &\quad (\$X.genre = \text{“}News\text{”} \\ &\quad AND\ \$X.summary \ni \text{“}HongKong\text{”}) \end{aligned}$$

粒度条件は，相対指定と絶対指定の2つの方法がある．

相対： $\$X/../*$
絶対： $\$X, \backslash\$X$

時間条件の指定 ここでは，属性を指定する場合に時間条件を付加した指定について述べる．

Example 3: 以下は,「123チャンネルの番組の中で，過去1週間の視聴率が10% 以上のオブジェクトを取り出せ」という検索を示している．

$$\begin{aligned} SELECT: &\quad \$X \\ WHERE: &\quad \$X.rate >= 10\ AND \\ &\quad \$X.channel = 123 \quad as\_of\ 7\ days \end{aligned}$$

連続問い合わせ これまで述べた例では，検索の実行は一度のみ行なわれていた．ここでは，問い合わせを将来にわたり連続的に行なうことで，未着データに対しての検索を行なう機構について述べる．例として，野球のワールドシリーズをすべて予約する場合を想定する．この時，試合数は試合結果によって，最大7試合，最少4試合が行われる．第1試合が始まる前に全ての試合を録画予約することは，従来の方式では不可能である．本方式では，一度リンクの設定を行えば，これを解決することが可能である．

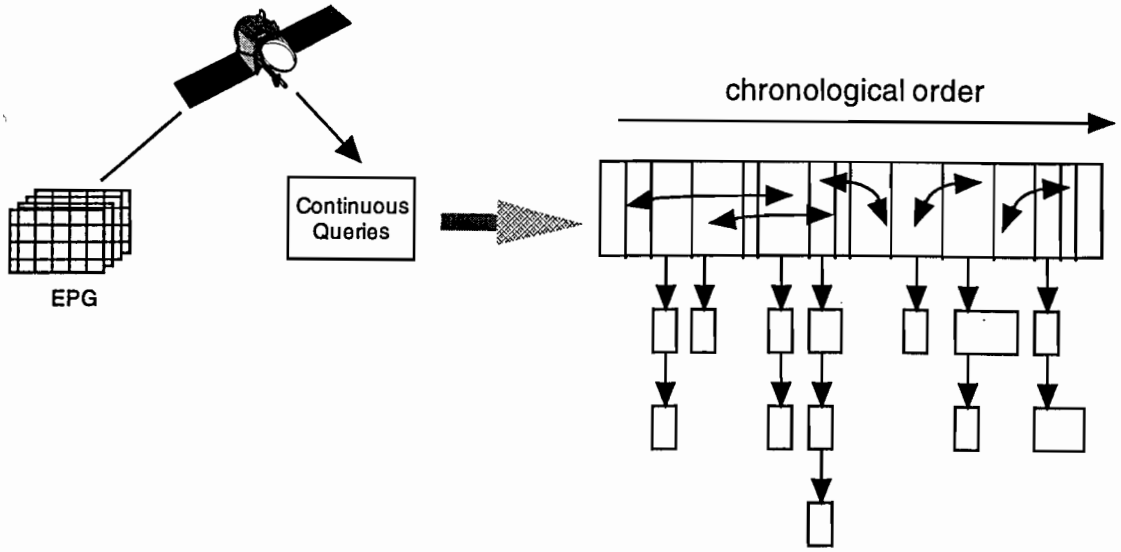


図 4.4: バーチャル・チャンネル

Example 4: 以下は,「すべてのチャンネルの中で，タイトルが"The World Series" というオブジェクトを取り出せ．ただし，この検索を連続的に実行せよ」という検索を示している．

$$\begin{aligned} SELECT: &\quad \$X \\ WHERE: &\quad \$X.title = "The\ World\ Series" \\ EXECUTE: &\quad continuously \end{aligned}$$

本節では，EXECUTE句にはcontinuouslyが指定されているため，この検索式は新着データが来る度に評価される．なお，EXECUTE句が省略された時，および，EXECUTE句にnowが指定された時は，どちらも一度限りの検索実行となる．

4.3.2 動的構造化

前節で取り出されたオブジェクトは，以下の2つのリンク構造によって動的に構造化される．VCHでは検索により抽出された番組情報に対して2種類のリンクを扱う．一つは，番組属性から，関連する他の番組属性へのリンクである．例えば，複数の番組間に生成された同一出演者属性同士のリンクなどである．このリンクは意味的に同一，関連，派生などを表すため,「内容的リンク」と呼ばれる． もう一つは，スケジュール情報を除き，番組属性が同一のオブジェクトへのリンクである．このリンクは「構造的リンク」とよばれる．ここで,『構造的』という用語は『スケジュール上の』という意味で使用している．図4.4にVCHの概念図を示す．横方向のリンクが内容的リンク，縦方向のリンクが構造的リンクを表している．

方，動的リンクは条件によりリンク元あるいは先が動的に変更される．これまでの研究では，リンクを生成する際に，リンク元とリンク先を検索し動的にリンクを生成する方式が提案されている [68].

内容的リンク VCHのリンクの一つは「内容的リンク」である．内容的リンクは，ある属性値から他の属性値へのリンクである．再放送，NVOD等もこのリンクで表現される．

内容的リンクはユーザに対して，番組選択を行なう際の補助的な情報として使用される．

$$\begin{aligned} \mathcal{CL}(t) \;=\; & (\{S.cast\},\{T.cast\},\{S.cast = T.cast\} \\ & (\{S.genre\},\{T.genre\},\{S.genre = T.genre\} \\ & (\{S.cast\},\{T.summary\},\{S.cast \in T.summary\} \end{aligned}$$

構造的リンク VCHのもう一つのリンクは,「構造的リンク」である．構造的リンクは，あるオブジェクトから他のオブジェクトへのリンクである．この場合，放送時間が同じオブジェクトにリンクが生成される．ここで，Allen の Temporal Interval として定義されている，equals, before, during, overlaps, meets, starts, finishes を用いて構造的リンクを生成する．

$$\mathcal{SL}(t) = O_i \;\theta\; O_j$$

検索により得られた番組情報は，優先順位情報を基に時系列にソートされ並び変えらることが可能である．ここで，再放送などのように同じ番組が何度も放送される場合，あるいはNVOD(Near video-on-demand)のように複数のチャンネルを使って，一つの番組を時間差で放送する場合がある．このような時，スケジューリングに関して，他の候補があることを示すために構造的リンクが用いられる．

優先順位は，どの検索式からの結果であるかにより決定される．言い換えると，検索式の優先順位が検索結果の優先順位にそのまま反映される．例えば，$Q_a \succeq Q_b \succeq Q_c$の3つの検索式が半順序関係で指定され，それぞれの結果が$\{o_{a_1}, o_{a_2}\}, \{o_{b_1}, o_{b_2}, o_{b_3}\}, \{o_{c_1}, o_{c_2}\}$であると仮定する．また，$o_{b_3}$には構造的リンクとして再放送（あるいはNVOD)であるo'_{b_3}に，o_{c_1}にはo'_{c_1}にそれぞれリンクが張られているとする．この時，o_{a_2}の放送時間にo_{b_3}が重なる部分がある場合を考える．この時，o_{b_3}は，o'_{b_3}に置き換えられる．これに伴い，o'_{b_3}とo_{c_1}に重なる部分があることになるため，o_{c_1}はo'_{c_1}に置き換えられる．これらの操作を図 4.5に示す.

$$\mathcal{SL}(t) = O_i \ \theta \ O_j$$

検索により得られた番組情報は，優先順位情報を基に時系列にソートされ並び変えらることが可能である．ここで，再放送などのように同じ番組が何度も放送される場合，あるいはNVOD(Near video-on-demand)のように複数のチャンネルを使って，一つの番組を時間差で放送する場合がある．このような時，スケジューリングに関して，他の候補があることを示すために構造的リンクが用いられる．

優先順位は，どの検索式からの結果であるかにより決定される．言い換えると，検索式の優先順位が検索結果の優先順位にそのまま反映される．例えば，$Q_a \succeq Q_b \succeq Q_c$の3つの検索式が半順序関係で指定され，それぞれの結果が$\{o_{a_1}, o_{a_2}\}, \{o_{b_1}, o_{b_2}, o_{b_3}\}, \{o_{c_1}, o_{c_2}\}$であると仮定する．また，$o_{b_3}$には構造的リンクとして再放送（あるいはNVOD)であるo'_{b_3}に，o_{c_1}にはo'_{c_1}にそれぞれリンクが張られているとする．この時，o_{a_2}の放送時間にo_{b_3}が重なる部分がある場合を考える．この時，o_{b_3}は，o'_{b_3}に置き換えられる．これに伴い，o'_{b_3}とo_{c_1}に重なる部分があることになるため，o_{c_1}はo'_{c_1}に置き換えられる．これらの操作を図4.5に示す．

4.4 インタラクティブ・データ配信のためのカルーセル型送出方式

4.4.1 デジタル放送によるインタラクティブサービス

デジタルテレビ時代のインタラクティブサービスは，ニュースや天気予報などをオンデマンドに配信するだけでなく，スポーツ番組における選手・ゲーム戦績情報配信のような番組関連情報配信，映画・イベント・番組そのもののプロモーション，ショッピング，ギャンブル，教育など様々な分野への応用が期待されている．

双方向網を含むデジタルテレビにおけるインタラクティブサービスの検討は，1994年以降，DAVIC (Digital Audio-Video Council) で詳しく行われている [51]．[52] はDAVICリファレンスモデルに基づく実装の一例である．DAVICにおけるインタラクティブサービスのリファレンスは，ISO SC29/WG12で策定されたMHEG5[53] であり，古典的なHyperCard同様のシーン単位でのナビゲーションによりインタラクティブ性を実現する．オンデマンドにシーンを取得するMHEG5の実行モデルは，DSM-CC[4]データカルーセルを用いることでMPEG2トランスポートストリーム（以下MPEG2-TS）によって伝送されるデジタル放送においても実現するものとしている [54]．

一方，デジタル放送の導入段階において最も重要なことは，視聴者にとっての導入阻害要因となる端末コストを低く押さえることである．古典的なHyperCardの実装は，少なくとも1シーンの全てのデータをCPUメモリに格納した上で実行するため，必要なRAM容量の増大によってコストの上昇を避けることができない．

また，[55] ではカルーセル伝送におけるレスポンス時間の向上について論じられているが，公共資源である帯域幅を有効活用するという観点からは，膨大な情報の中から予めフィルタリングされた情報を送ることも重要である．本節では，このような課題に着目して新たに開発された，デジタル放送向けのインタラクティブマルチメディアデータ配信のためのカルーセル方式伝送DVXについて述べる．

4.4.2 カルーセル型送出方式DVX

DVXの伝送フォーマットの基本的なアイデアは，STBが本来持っているMPEG処理機能を有効に活用して，効果的な表現機能を有するインタラクティブマルチメディアサービスを実現しようというものである．

[4]DSM-CCは，ISO SC29/WG11 (MPEG)の定める規格の一部であり，映像・音声・データに関するサーバと端末の間の伝送の手順を定めている．特にデータカルーセルは [50] にある放送型データ配信を実現する具体的手順とMPEG2-TSのプライベートセクションを用いた伝送フォーマットについて規定している．

受信端末のハード構成

図4.6は標準的な衛星デジタルSTBのハード構成のブロック図である．デジタル放送の電波はチューナー，QPSKによって復調されたMPEG2-TSはトランスポートデコーダで処理される．MPEG2-TSは複数の映像・音声・データブロックを伝送するが，それぞれのセクションのヘッダ部分によって所望のセクションが識別される．

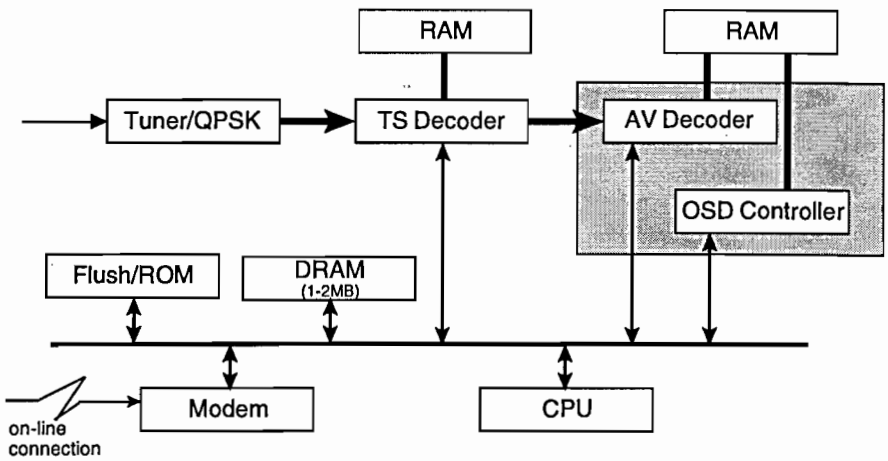


図 4.6: STBハード構成のブロック図

トランスポートデコーダは，MPEG2-TSの全てのセクションの中からヘッダ部分をみてハードウェアでフィルタリングを行うことで，受信中の映像・音声のセクションを後段のAVデコーダに送る他，所望のデータをCPU側のメモリ空間に転送する．

トランスポートデコーダからCPU側に送られる情報としては，EPGのための番組情報やインタラクティブサービスのためのデータがある．しかし，CPU側のRAMは通常1～2MBであり，主として番組情報の格納のために用いられる．

AVデコーダは，MPEG2の音声･映像（動画および静止画）を再生する機能を有する．また，STBではEPGを表示する目的で，CPUから制御して情報を表示するためのOSD（On-Screen Display）表示機能を有している．通常OSDは4bit/pel～8bit/pelのグラフィック表示能力をもつ． 図4.7に示すように，MPEG画面とOSD画面のプレーンは合成されてテレビ画面上に表示される．

DVXのカルーセルによる伝送方式

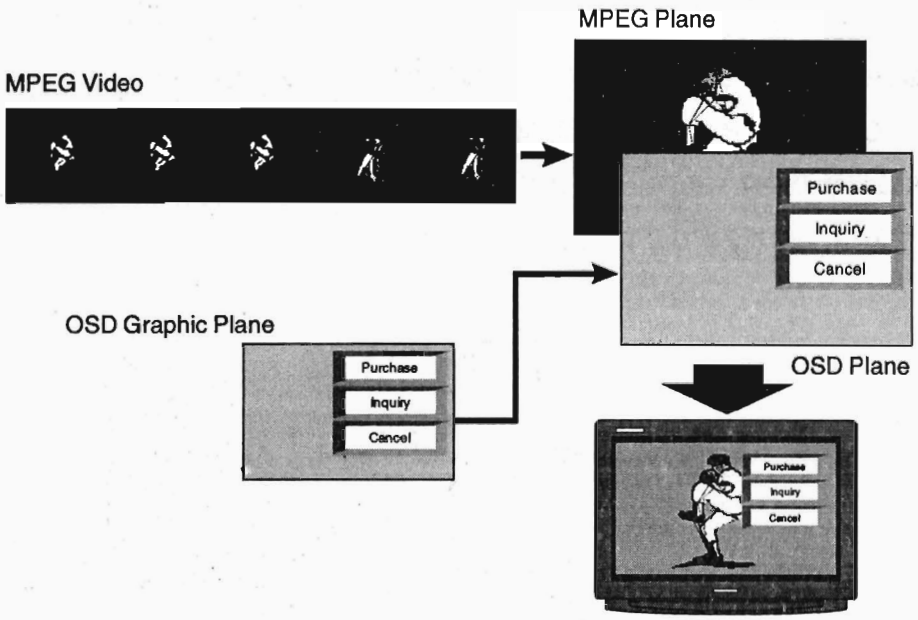


図 4.7: MPEG 画面と OSD 画面の合成

(1) 基本的な設計方針

このような安価な STB で実用的なインタラクティブサービスを行うために，我々は以下の設計方針のもとで伝送方式を定めた．

1. シーン毎のデータ構造

   画面表示の観点から，RAM 上にロードせざるを得ない最小単位は 1 画面に表示される情報である．よって，シーン単位でのデータ構造を基本とする．

2. MPEG 映像によるフルカラー映像表現

   ただし，フルカラー映像（静止画・動画）については MPEG 再生機能を活用し，CPU 側への転送や，CPU 側メモリの利用を前提としない．

3. ハードウェアフィルタリング機能の活用

   各シーンに対する ID をセクションの先頭部分に埋め込むことで，トランスポートデコーダのハードウェアフィルタリング機能によるデータ取得を行う．

4. 機種非依存のフォーマット

   様々なメーカーの STB 端末で再生できるよう，データおよびインタラクティブの制御のためのプログラム記述は機種に依存しないものとする．

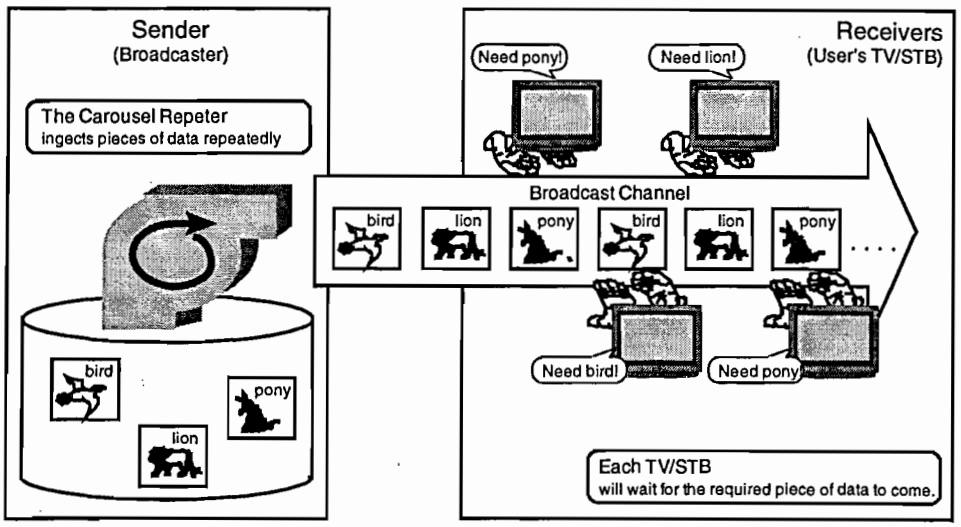


図 4.8: データカルーセルの概念図

(2) シーンの識別と伝送フォーマット

DVXでは1画面に表示されるシーンのデータをコンテンツと呼ぶ．DVXではコンテンツの構成要素のうち，オンデマンドに取得されるものをカルーセル(図4.8)で放送時間中常に送出し，映像（動画・静止画)・音声についてはCPUの介在なく再生できるMPEGエレメンタリストリームを用いることとした．

具体的には，DVXコンテンツは映像・音声・ナビゲーション情報(NI)の三つの要素から構成される．ここで，NIは，シーンでのインタラクションに必要となるOSD画面に表示されるボタンやプログラムを格納したデータである．

これら3種類の情報は，図4.9に示すように1本のMPEG2-TSの中にエンコードされて伝送される．

DVXフォーマットを通常放送番組と同様の仕組みで選局・再生できるようにするために，デジタル放送方式において最も一般的な，DVB-SI(Digital Video Broadcasting - Service Information)[56]に従った番組運用が可能となるよう設計にされている．

1. **MPEG映像:** 動画については通常のMPEGのビデオエレメンタリストリームとして送出される．静止画についてはMPEG2 MP@MLの1枚のI-frameとしてカルーセルとして繰り返し送出する．しかし，CPU側への静止画の伝送を避けるため，I-frame静止画はビデオエレメンタリストリームとしてエンコードする．このため，所望のI-frameが送出されるタイミングでAVデコーダを適切に制御するための，補助的な情報であるVET(Video Element Table)もカルーセルとして送られる．

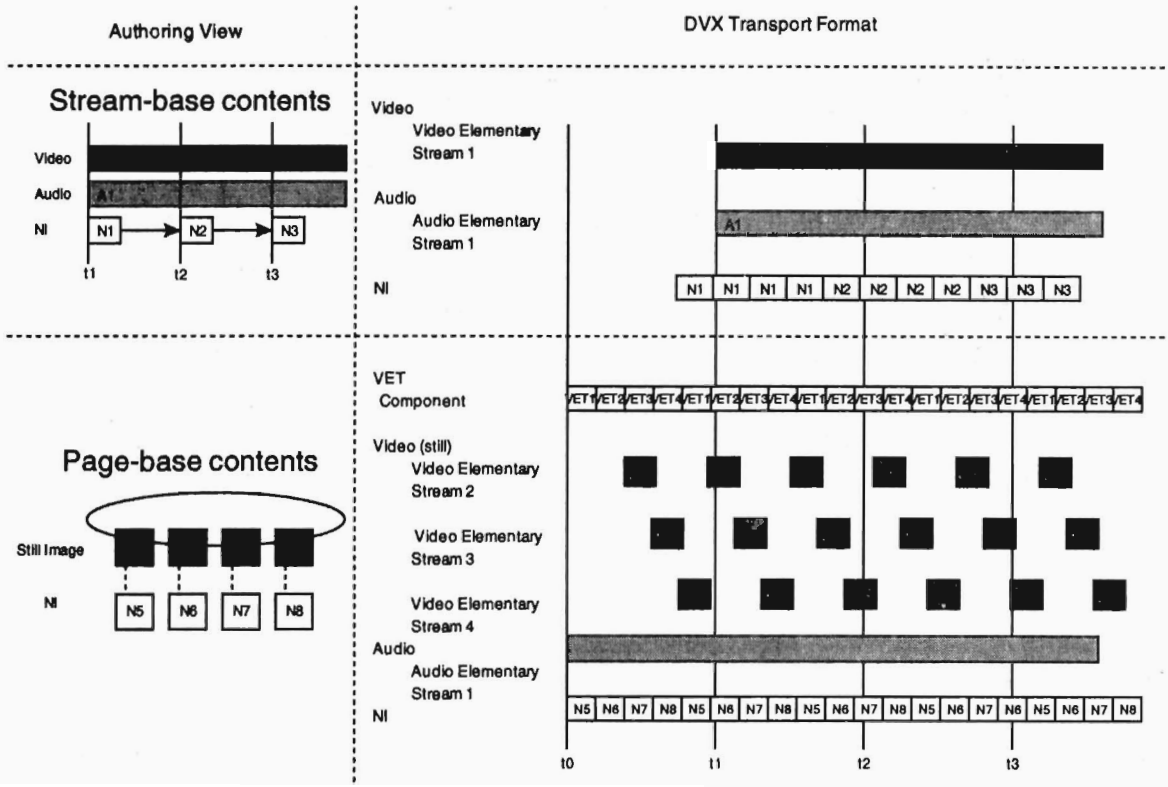


図 4.9: コンテンツの多重化

2. **MPEG音声:** 音声については，通常のMPEGのオーディオエレメンタリストリームとして送出される．

3. **NI (Navigation Information):** NIはMPEG2-TSのプライベートセクションを用いたカルーセルとして繰り返し送出される．伝送フォーマットとしてはDSM-CCデータカルーセルと互換性のあるフォーマットを用いている．

これらの三つの構成要素のうち，ビデオ・オーディオのエレメンタリストリームは，元来ハードウェアが持つ機能により直接トランスポートデコーダからAVデコーダに渡されて再生される．よって，静止画のデコードのタイミングを制御するために必要なVETと，NIのみをCPUで解釈すればよい．

コンテンツの識別はこれら三つの構成要素に対する参照によって行う．三つの構成要素に対する参照を常に独立させることによって，シーンの切り替え時に，トランスポートデコーダに対するフィルタリング条件を同時にかつ独立して設定できる．このことにより，以下の利点がある．

- NIを取得する前に，どのビデオ・オーディオが必要かがわかるので，NIの取得完了と同時に再生を開始できる．つまり，シーン間の不要な待ち時間がない．

- 同じ構成要素を共有するシーン間で，共有されている構成要素については不要な操作を行わずに済む．例えば，BGM のオーディオを共有するシーン間の切り替えで，次のシーンでも同じ BGM を使うことが予めわかるので音を途切れさせることがない．

また，NIについてはバージョンアップ機能をサポートしている．これは，一つのNIであっても，より新しいバージョンが放送されると，STB端末側でそれを取得し直す機能である．この機能によって，時々刻々変換する情報について端末側で自動的に表示を更新することができる．この機能も，MPEG2-TSのセクションヘッダの中のバージョン情報のフィールドを用いて実現しているため，やはりハードウェアフィルタリングのみを用いて実装可能である．

NIのインタラクティブ機能

NIは，シーン上のインタラクションを実現するためのデータであるが，おおまかに言えば以下の情報から構成される．

グラフィックオブジェクト： DVXで規定されたボタン・テキスト表示フィールド・テキスト入力フィールドのいずれか．各グラフィックオブジェクトは，1) 通常状態，2) 操作対象状態，3) 選択状態，4) 操作対象かつ選択状態，の4つの状態がある．例えば，ボタンオブジェクトは，OSDに表示するビットマップ，ビットマップ列によるアニメーション，指定色の矩形のいずれかを，各状態に対応づけることができる．

フォーカス制御テーブル： リモコンの操作に応じて，操作対象となるグラフィックオブジェクトの遷移を指定する．具体的には，あるオブジェクトIDのグラフィックオブジェクトが操作対象である時に，上下左右の矢印キーのそれぞれの押下に対して，次に操作対象となるグラフィックオブジェクトのオブジェクトIDが示される．

バイトコードハンドラ： シーンの初期化，グラフィックオブジェクト上での選択操作，指定時刻の到来などのイベントに対して，実行すべき処理内容を記述したプログラム．様々な機種での実行が必要になるので，プログラムは機種に依存しないバーチャルマシンのバイトコードによって記述される．記述される処理内容には，グラフィックオブジェクトの制御・シーンの切り替え・モデム経由での通信・番組予約等のEPG制御などがある．

以上から分かるように，DVXのNIの基本的な実行モデルは，システムで定義されたオブジェクトによる表現機能と，イベントドリブンのプログラム実行に基づいている．この点は，概念的にはHyperCardやMHEG-5と同様である．

コンテンツ合成モデルとデータフロー

放送における情報伝達はCD-ROM等のパッケージメディアと異なり，即時性をもつ情報の伝達に価値が見出される場合が多い．例えば，天気予報は少なくとも数時間前の情報であるし，野球番組における打者の打率は前の打席の結果を反映するものである．よって，DVXを用いたインタラクティブ番組では，即時性のあるデータからリアルタイムにコンテンツを合成して送出することが要求される．

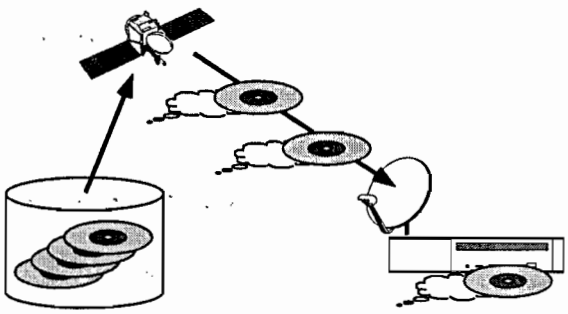


図 4.10: DVX のコンテンツ合成

即時性のあるデータは，様々なデータベースから提供されるため，既存のRDBMS等との連携が必要になる．例えば，天気予報番組なら予報データベースから，野球番組なら選手情報データベースから，視聴者にとって必要なものをNI（もしくは静止画）に埋め込んで送出する．

このようなデータ配信形態と，前節迄に述べたカルーセル伝送方式を，放送を用いたデータ配送という観点からみると以下の考察が得られる．

- データベース中の全ての情報を伝送路にのせるのではなく，送出側で視聴者にとって必要となる情報をフィルタリングしてのせる．すなわち，トータルでは送出側と端末側での2段階のフィルタリングとなる．

- 1次フィルタリングの結果として合成されるデータは，文字列や数値だけではない．多数の音声や映像の中から，視聴者に必要なものを帯域幅の制約に照らして選択し，MPEG2-TSの中に複数のストリームとして多重して送ることも可能である．また，合成の形として文字列や数値をMPEG-I静止画に画像として埋め込むことで，それらのフルカラー表現を得る場合もある．

- 1次フィルタリングの結果，コンテンツとして合成されて送られるのはデータだけではない．2次フィルタリングとしての端末側でのデータのブラウズを容易にするため，送出側では1

次フィルタリングの結果に応じて，それに適合したインタラクションのための手続きも合成する．

- データベースの情報の更新に伴い，送出側の1次フィルタリングが自動的に再実行される．コンテンツは再合成されて，新しいバージョンのNIとして伝送され，端末側での表示を自動更新することができる．

DVXでは，MPEG映像, MPEG音声, NIの3つの組$\{V, A, NI\}$でコンテンツを表現することは既に述べた．DVXに含まれるコンテンツは，直観的には「『リアルタイム情報を含むコンテンツが合成されたパッケージメディア』が配送される」と言える(図4.10)．映像，音声を素材とし合成を行なうフレームワークは[57]で提案されているが，リアルタイム，かつ，インタラクションに関する情報を扱ったものは提案されていない．

4.5 結言

1. VCHはユーザが定義したビューに基づきEPGデータを動的に再構築するための枠組である．VCHは以下の特徴を持つ：

   (a) 各々の番組情報は提案する連続的検索機構により抽出される
   (b) 番組情報は構造的リンクにより，同一および等価な番組情報へのリンクを保持する
   (c) 番組属性は内容リンクにより，関連情報へのリンクを保持する

   現在，番組情報の動的再構築の概念設計を完了している．今後は，プロトタイプを設計し検証を行なう予定である．また，リモコンを含めたユーザインタフェースの検討も行なう予定である．

2. デジタル放送における，カルーセル方式を用いたインタラクティブサービスを実現するDVXについて，その伝送方式と，データベース連携の観点からのサービス展開の可能性について述べた．DVXの伝送方式は，今後のデジタル放送の標準となるMPEG2のハードウェア処理機能を活用したものとなっている．

   今後，家庭における情報端末において，様々なインタラクティブサービスが展開されると考えられる．またインターネットのインフラの活用も求められるようになろう．DVXも，サービスのニーズ，ハード機器の進化，そしてコンテンツの互換性を念頭においてさらに発展させてゆく必要がある．

§ 5

放送型ハイパーメディアのための時間依存リンク機構

5.1 緒言

近年，インターネットやデジタル衛星放送(DVB)[58]による情報配信システムが注目を集めている[59][61]．これらのシステムの基本的な枠組みは，従来のプル型サービスによるものではなく，プッシュ型サービスに基づいている．プル型サービスとは，例えば，World Wide Web (WWW)からの情報取得のようにユーザが情報を明示的に指定して取り出す仕組みである．すなわち，ユーザが情報ソースの場所を指定するか，あるいはトピックを指定することで，サーチエンジンなどを用いて情報を取り出すことにより実現されている．一方，プッシュ型サービスでは，ユーザの要求，あるいは情報自身の更新により情報が自動的にユーザに配信されるので，ユーザが情報ソースに直接アクセスする必要はない．このサービスの仕組みは，ユーザが興味のあるチャンネルやソースをあらかじめ登録しておくことで，システムが各ユーザごとに定期的に情報を配信することにより実現されている[62]．

プッシュ型サービスで配信される情報は，テキスト情報だけでなくハイパーテキストやハイパーメディアを含むことが可能である．本論文では，対象として，単にテキストのみを扱うのではなくハイパーテキストやハイパーメディアを含めて扱う枠組みについて議論する．特に，そのリンク機構（ハイパーリンク）について述べる．一般に，ハイパーリンクは2つの種類に分類される．一つは，静的リンクと呼ばれるものであり，作成者があらかじめオーサリングを行ったリンクである．このリンクはユーザにより変更されることはない．もう一つは，動的リンクと呼ばれるものであり，作成者あるいは作成者以外が生成・更新・削除を行なうことが可能なリンクである．動的リンクのメカニズムについては，いくつかの方式が提案されている[63]．例えば，計算リンク

[64][65][66][67] や質問リンク [68][69] である．本論文では，静的リンクはもちろん，動的リンクにも適用可能なハイパーリンク機構を提案する．

実際のハイパーメディアにおいては，情報が頻繁に更新されるために，その情報間の参照情報であるリンクを管理することは大きな課題となっている．特に，時間の経過に伴って変更される情報，時系列データ，およびリアルタイム性を持つ情報では重要な問題である．なぜなら，情報の更新期間が短く，かつ頻繁であるためである．

我々は，上記問題を解決するためにハイパーリンクのための時間依存リンク機構を提案し，プロトタイプシステム *Mille-feuille*[70][71] を設計している．現在，提案する方式の基本的な機能，すなわち，情報送信／受信部，およびユーザインタフェース部については実装を完了している．また，既存のハイパーメディアの枠組みに対して，本論文で提案するリンク機構を適用する方法について考察を行う．

以下，本論文の構成を示す．まず，2.では本研究の動機であるハイパーテキストのリンク管理，および，プッシュ型サービスにおける問題点について議論する．3.では，時間依存リンクの定義について述べる．4.では3.で述べた時間依存リンクの制御方法について述べる．5.では提案する方式を用いた放送型ハイパーメディアの実装について述べる．6.では拡張可能なマーク付け言語 (XML) を用いた実装方式について議論する．7.ではまとめと今後の課題について述べる．

5.2 動機

5.2.1 ハイパーテキストにおける不完全リンク

ハイパーテキストでは，たくさんの情報の中からリンクを辿ることによって，ユーザが欲しい情報を対話的に取り出すことが可能である．実際，ハイパーテキストの概念を用いたWWWでは，大量の情報にアクセスすることが可能である．しかし，情報ソースを指定し，その情報にアクセスしたが取り出すことができない場合がある．すなわち，リンク先の情報が削除されているのにもかかわらず，その情報に対してのリンクのみが存在する場合である．この時，Hypertext Transport Protocol 1.0 (HTTP)[72] の仕様では，**"Error 404. Not Found—file doesn't exist"** のエラーメッセージを返し，リンクが見つからないことをユーザに提示する．このようにリンク元とリンク先が正しく結合されていないリンクのことを不完全リンク (dangling link)[1] と呼ぶ [73]．この不完全リンクはハイパーテキストの利用の際に大きな障害となっている．

現在，不完全リンクを排除するためにいくつかの方法が提案されている．例えば，Alexa[74] は全世界のすべてのWWW コンテンツのアーカイブを作成することによって，この問題を解決している [75]．このシステムは非常に有効であるが，大量のアーカイブ (約 8 terabytes) を管理する

[1]形式的に言えば，参照完全性 (referential integrity) を満たさないリンクを指す．

ためには非常に手間がかかるという問題がある．また，Atlas[73]では，WWW サーバーにリンクを管理するためのプロトコルを追加することで，参照が正しくなくなった場合の処理を指定することが可能である．ただし，コンテンツ自身の有効時間等の処理については考慮されていない．

従って，コンテンツやリンクの有効時間，言い換えると寿命(life span)の概念を導入することで，リンクを自動的に生成／消滅する機構により，リンクの参照完全性を実現する枠組みが必要となる．すなわち，従来のハイパーテキストのリンク機構に対し，時制を導入することにより不完全リンクや無効リンクを処理する機構が必要である．

5.2.2 プッシュ型サービスにおけるデータ管理

近年，PointCast Network[76] や Castanet 2.0[77] 等のプッシュ型サービスを開発する枠組みが提案されている．これらのシステムは，主に情報配信，ソフトウェア配布，およびドキュメント配信に用いられている．例えば，PointCast は，定期的に最新の時事ニュースをインターネットを通じてユーザに配信するサービスを実現するソフトウェアである．プッシュ型サービスでは，サーバー側からクライアント側への配信時に，どのデータをどのタイミングで送信するかは重要な論点である [62].

ここで，プッシュ型サービスにおけるデータ配信の例を考えてみる．あるサーバーに2つの相互に関連したデータが存在しており，これらのデータは個々に配信されることになっている．この場合，一般的には2つのデータはクライアント側に同時に到着するとは限らない．すなわち，サーバー側においては2つのデータは関連していることが分かるが，クライアント側ではその関連が分からなくなる可能性がある．言い換えると，送信されるデータの参照一貫性は保証されない．なぜなら，(1) 明示的な関連情報が無い，(2) サーバー側での部分的なデータの更新がクライアント側で正確に反映されるとは限らないからである．従って，プッシュ型サービスに対する要件は以下の通りであると考えられる．

- 関連データに対する自動的な参照生成
- 時間経過を考慮した一貫性管理

従来のハイパーテキスト／ハイパーメディアでは，データ間に存在する参照情報はすべて手動で作成する必要があった．個々の参照情報を手動で作成する作業は非常に手間のかかる作業である．また，参照先のデータがクライアントに到着しておらず不完全リンクになっているデータは，時間が経過した後に参照先のデータが到着し，不完全リンクが解消される場合もある．従って，プッシュ型サービスには，時制に関する一貫性を管理する枠組みが必要である．

これまでのハイパーメディアにおける時制管理は，Dexterハイパーテキスト参照モデル [78] に時間概念を導入したAmsterdam ハイパーメディアモデル [79] に代表されるように，主にメディ

アの同期をとるための枠組みについて議論されてきた[80]. しかし，データの関連性や有効性を考慮した枠組みについては検討されていない. 我々は，プッシュ型サービスなどに用いられるハイパーテキスト／ハイパーメディアの枠組みを特に**放送型ハイパーメディア**と呼び，このメディアに対するデータの関連性および一貫性のための枠組みを提案する.

5.3 時間依存リンク

本節では，不完全リンクの管理，あるいは放送型ハイパーメディアのためのリンク制御を行う時間依存リンクモデル (TDL: Time Dependent Link) について述べる. TDLはリンク先ノードとリンク元ノードにより定義される.

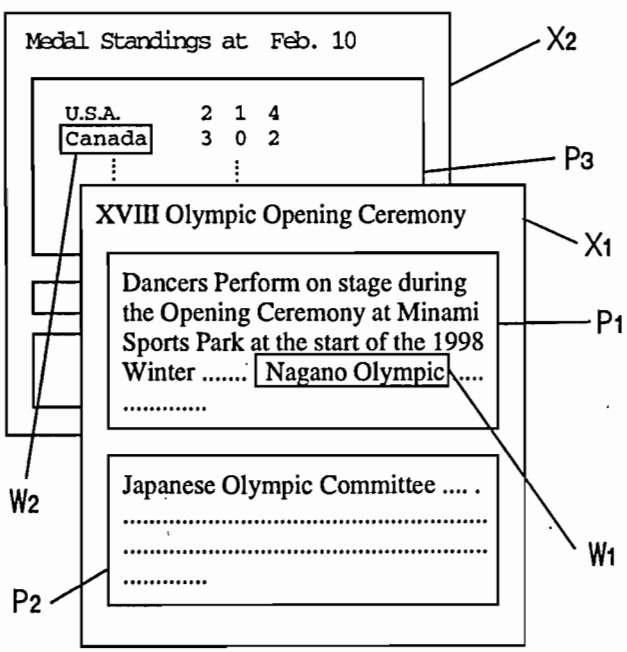


図 5.1: アンカーとコンテナの例

5.3.1 アンカーとコンテナ

提案するTDLで用いられるリンクモデルは動的リンクに基づくものである[2]. リンクとノードとは別に管理されており，リンクとノードはどちらも実行時に動的に生成・更新・削除される. リンクをノードから独立させて管理する点はDexterハイパーテキスト参照モデルと同様である[78]. また，リンクの参照ポイントであるアンカーも動的に生成・更新・削除され，それを含むオブジェ

[2]もちろん，動的リンクは静的リンクを包含するので，静的リンクも扱うことが可能である.

クト(ドキュメントやメディア)には依存しない．WWWで用いられるHTML等はアンカーがあらかじめ埋め込まれているため，本モデルとは大きく異なる．

TDLにおけるノードはアンカーと呼ばれる．また，アンカーを含むオブジェクトはコンテナと呼ばれる：

- アンカーはコンテンツに含まれるすべてのオブジェクトが対象となる．
- コンテナは複数のアンカーを含むことが可能である．

ニュース記事のコンテンツ(図 5.1)を例にすると，各ニュースのページがコンテナであり，キーワードやパラグラフがアンカーとなる[3]．ページX_1やX_2がコンテナであり，キーワードW_1やW_2，およびパラグラフP_1, P_2, P_3がアンカーとなる．もちろん，ページX_1やX_2もアンカーになることも可能である．従って，すべてのレベルのコンテンツがアンカーになる可能性がある．

5.3.2 トランザクション時間と有効時間

アンカーとコンテナがサーバーのデータベースに登録された時間をトランザクション時間(transaction time)，内容が有効である時間を有効時間(valid time)と呼ぶ．Snodgrassの時制データベース[49]の定義に従えば，トランザクション時間とはデータベースにいつその記録がされたのかを示す時間である．一方，有効時間とは実時間における時間を表し，実世界で事象が発生した時刻や，いつからいつまである状態が続いていたかという情報を表す時間である．

TDLでのトランザクション時間と有効時間について説明する．

アンカーのトランザクション時間： tr　コンテナにそのアンカーが登録された時間である．

コンテナのトランザクション時間： Tr　コンテナがデータベースに登録された時間である．

アンカーの有効時間： $[v_1, v_2]$　アンカーがナビゲーションポイントとして機能する期間を示すものである．すなわち，アンカーは時刻v_1から有効になり，時刻v_2になるとナビゲーションポイントでなくなる(消滅する)．例えば，アンカー“Nagano Olympic”は「第18回冬季オリンピック」の開催期間中[Feb.7, 08:00, Feb.22, 21:00]は有効である．

コンテナの有効時間： $[V_1, V_2]$　コンテナがアクセス可能である期間を示す．これは，コンテナの寿命ととらえることもできる．例えば，コンテナ“Medal Standings at Feb.10”は[Feb.10, 00:00, Feb.10, 24:00]の期間はアクセスすることが可能である．

[3]アンカーはテキスト・コンテンツに限らない．すなわち，音声，ビデオ，アニメーションもアンカーになる可能性がある．

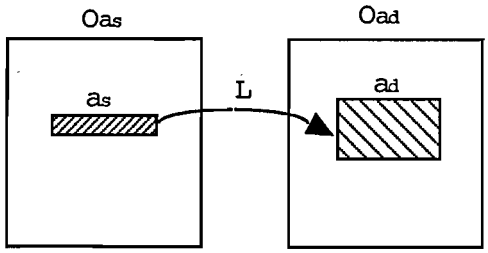


図 5.2: ノード a_s から a_d へのリンク

5.3.3 リンクの定義

TDLはリンク元とリンク先のアンカーの組 a_s, a_d で定義される：

$$L = (a_s, a_d)$$

図 5.2はコンテナ O_{a_s} に含まれるアンカー a_s から，コンテナ O_{a_d} に含まれるアンカー a_d へのリンクを示している．

5.3.4 リンクが有効である条件

リンクの有効時間はアンカーとコンテナの有効時間から計算される．リンクが有効であるためには，これから示す8つの条件をすべて満たさなければならない [70]．説明のために，表 5.1にリンク元とリンク先のそれぞれのアンカーとコンテナのトランザクション時間と有効時間とを示す．

まず，tr_s, tr_d, Tr_s, Tr_d は t_{now} より時間的に前である．ここで，t_{now} は現在時間を示す [82]．すなわち，以下の4つの条件を満たさなければならない．

$$tr_s \leq t_{now} \tag{5.1}$$

$$tr_d \leq t_{now} \tag{5.2}$$

$$Tr_s \leq t_{now} \tag{5.3}$$

$$Tr_d \leq t_{now} \tag{5.4}$$

次に，アンカーとコンテナの積は現在時間を含まなければならない．この条件はリンク元とリンク先の両方において満たされなければならない．例えば，アンカー "Nagano Olympic" が有効であるためには，それを含むコンテナ "XVIII Olympic Opening Ceremony" も有効である場合のみ，ナビゲーションポイントとして有効である．

$$max(V_{s_1}, v_{s_1}) \leq t_{now} \leq min(V_{s_2}, v_{s_2}) \tag{5.5}$$

$$max(V_{d_1}, v_{d_1}) \leq t_{now} \leq min(V_{d_2}, v_{d_2}) \tag{5.6}$$

	source		destination	
	anchor	container	anchor	container
transaction time	tr_s	Tr_s	tr_d	Tr_d
valid time	$[v_{s_1}, v_{s_2}]$	$[v_{s_1}, v_{s_2}]$	$[V_{d_1}, V_{d_2}]$	$[V_{d_1}, V_{d_2}]$

表 5.1: トランザクション時間と有効時間

図 5.3 の4つの図は上から順に，条件(5)から(8)までを示している．ここで，白抜きの四角と斜線の四角は，それぞれコンテナとアンカーを示している．

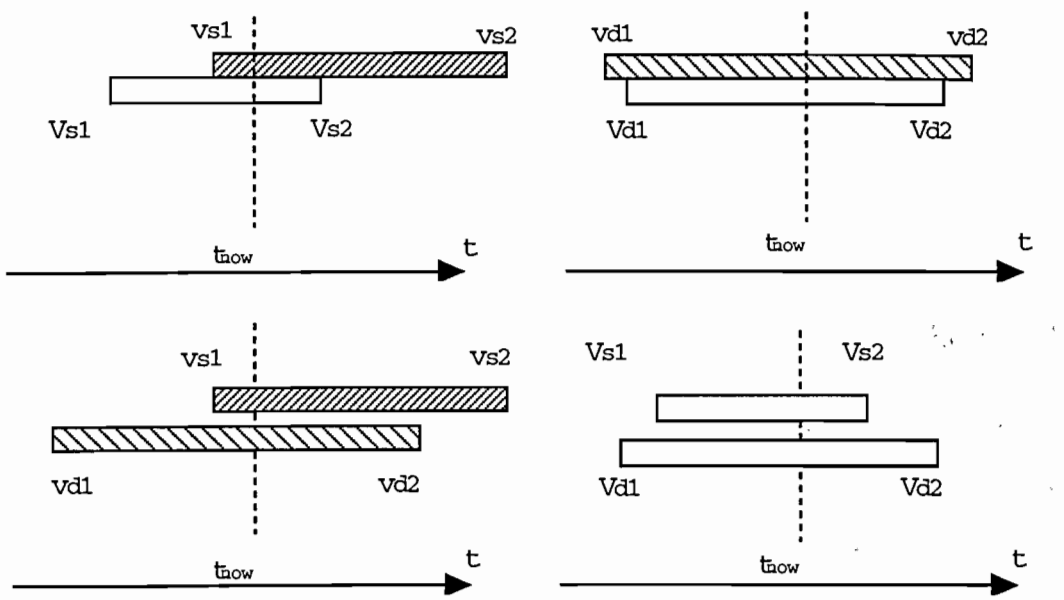


図 5.3: リンクが有効である条件

さらに，リンク元とリンク先のアンカーの有効時間の積は現在時間を含まなければならない．

$$max(v_{s_1}, v_{d_1}) \leq t_{now} \leq min(v_{s_2}, v_{d_2}) \tag{5.7}$$

最後に，リンク元とリンク先のコンテナの有効時間の積は現在時間を含まなければならない．

$$max(V_{s_1}, V_{d_1}) \leq t_{now} \leq min(V_{s_2}, V_{d_2}) \tag{5.8}$$

これらの8つの条件をすべて満たした場合のみ，時刻t_{now}においてリンクが有効である．

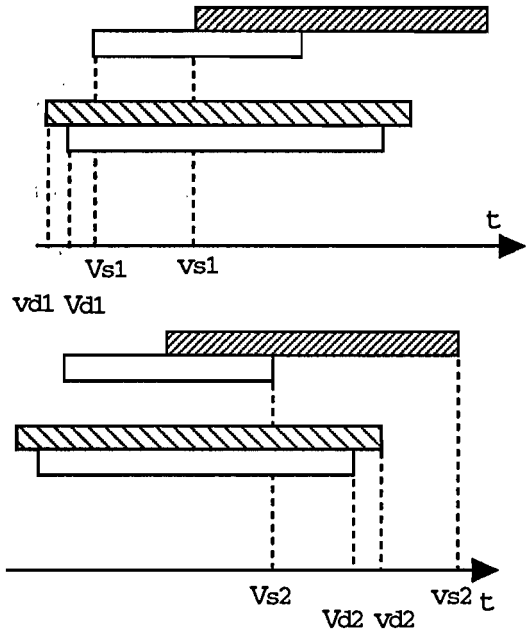


図 5.4: リンクの生成と消滅

5.3.5 時間依存リンクの生成と消滅

TDLは，リンクが有効である期間のみ存在する．すなわち，リンクが有効となった時刻がリンクの生成時間であり，リンクが無効となった時刻がリンクの消滅時間である．生成時間と消滅時間は，以下のように計算される．まず，生成時間 L_{create} は4つの時間 v_{s_1}, V_{s_1}, v_{d_1}, V_{d_1} の中で最も後の時刻であり，消滅時間 L_{expire} は4つの時間 v_{s_2}, V_{s_2}, v_{d_2}, V_{d_2} の中で最も前の時刻である (図 5.4).

$$L_{create} = max(v_{s_1}, V_{s_1}, v_{d_1}, V_{d_1})$$
$$L_{expire} = min(v_{s_2}, V_{s_2}, v_{d_2}, V_{d_2})$$

従って，リンクの有効時間は以下で表される．

$$L_{valid} = [L_{create}, L_{expire}]$$

TDLでは，このようにノードの有効時間により自動的に生成・消滅する動的リンクを実現している．

5.4 リンク制御方式

本節では，TDLの制御方式について述べる．ここでは，TDLを用いた不完全リンクの管理方法と基本操作を提案する．

5.4.1 不完全リンクの管理

ハイパーテキスト/ハイパーメディアのリンクの一貫性を保持するために，不完全リンクを排除するための管理方式を提案する．不完全リンクが起こるのは以下の場合である：

1. コンテナが削除される
2. アンカーの内容が不適当である
3. 1. と 2. の両方

1. の場合，コンテナの有効時間$[V_1, V_2]$をチェックすることにより回避することが可能である．2. の場合，アンカーの有効時間$[v_1, v_2]$をチェックすることにより回避することが可能である．すなわち，1. と 2. のチェックの方法は，どちらの場合もリンクの有効時間を調べることと同等である．もちろん3. の場合も同様である．

従って，リンクの有効時間をあからじめ計算することで，不完全リンクを排除することが可能であることが分かる．実際，コンテナをアプリケーション/ブラウザにロードする前に，各アンカーに関してリンクの有効時間を計算しておくことが実用的であると考えられる．Web のブラウザ等では，以下のような表示方法が考えられる：

- 有効なアンカーはハイライトや下線によりナビゲーション可能であることを示す
- 無効なアンカーは通常のコンテンツと同様の扱いにするか，あるいは削除する

アンカーに対する処理は各アプリケーションにより違うので，チェック方式についても各アプリケーションごとに特有の方式を設計する必要がある．

5.4.2 放送型ハイパーメディアの特性

前節で述べた不完全リンクは，パッケージ型あるいはサーバーアクセス型のハイパーテキスト/ハイパーメディアを扱う場合には障害となるために排除されなければならなかった．一方，放送型ハイパーメディアでは，リンクの不完全性を容認することが必要な場合が存在する．放送型ハイパーメディアの特性から容易に推測できることであるが，インターネットやデジタル衛星放送を通じて配送されるデータの送信順序や受信順序は必ずしも各クライアントで一定であるとは限

らない．従って，一時的に一部のデータが**未着**(upcomming)である状態が存在する．まとめると，放送型ハイパーメディアには以下のような特性がある：

- ノードとリンクはサーバ側で動的に追加・更新・削除される
- データが部分的・段階的に配送されるため，サーバー側でのノード/リンクの構造とクライアント側でのノード/リンクの構造が違う場合がある．

例えば，図 5.5は，いくつかの関連するニュースを含んだ時事ニュースの例である．ここでは，「1月の首都圏において，大雪のため警報が発令され，交通への影響が懸念されている．その後，時間経過とともに交通情報やニュースが発表される」という状況を時系列で示している．

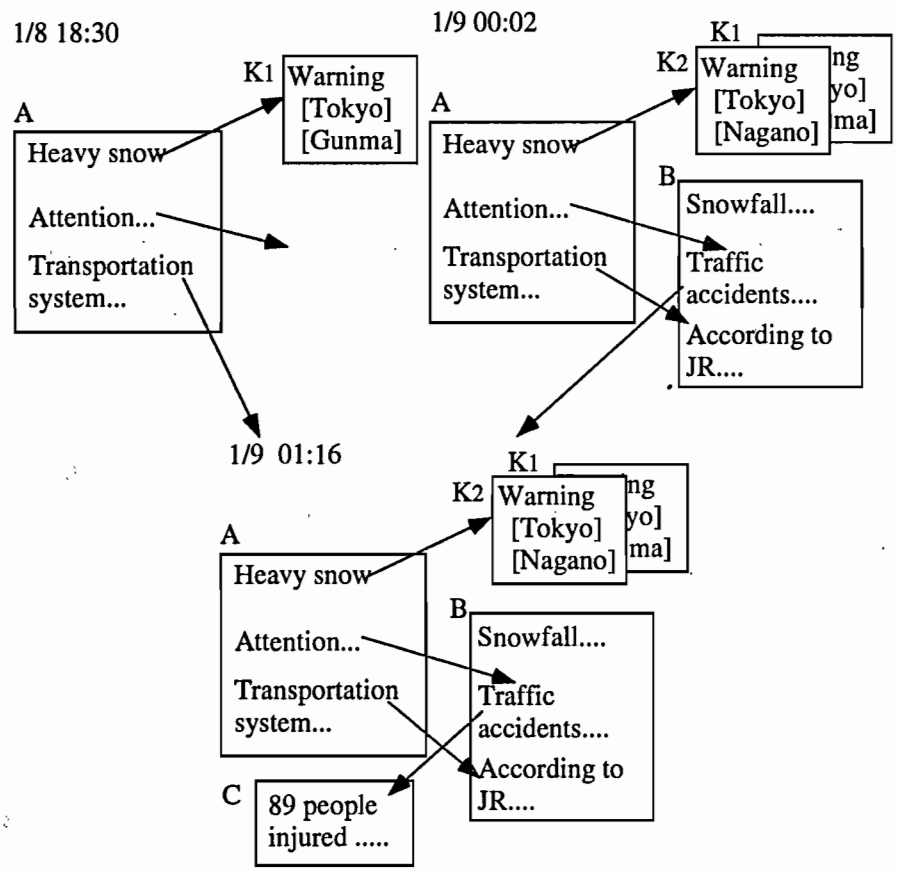


図 5.5: ハイパーメディアの更新

この例では，上から順に，それぞれ，[Jan.8, 18:30]，[Jan.9, 00:02]，および[Jan.9, 01:16]の時点で配信されたニュース記事を示している．四角はニュース記事(コンテナ)，矢印はリンクである．

1. [Jan.8, 18:30]の時点(上図)では，ニュース記事Aにおける"Heavy snow"から警報情報K_1へのリンクが存在することを示している．一方，"Transportation system..."はリンク

先が未着であるため，待ち状態である．すなわち，このリンクはこの時点では不完全リンクである．

2. [Jan.9, 00:02]の時点(中図)では，ニュース記事Bが到着し，この時，不完全リンクが解消したことを示している．同様に，ニュース記事Aにおける"Heavy snow"はK_1の新しいバージョンであるK_2が到着しているために，リンク先が変更されていることを示している．

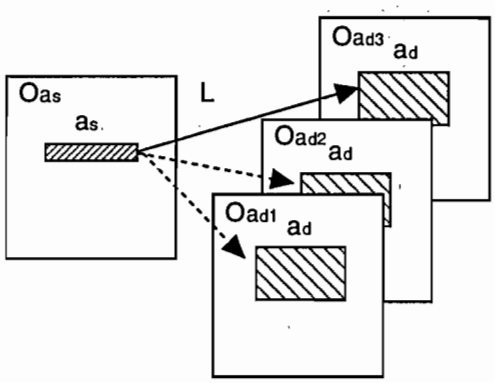
3. [Jan.9, 01:12]の時点(下図)では，ニュース記事Cが新たに到着し，''Traffic accidents...''からの不完全リンクが解消したことを示している．

5.4.3 放送型ハイパーメディアの制御

未着リンクの制御

放送型ハイパーメディアの場合は，不完全リンクは容認されるべきものであることは前節で述べた．すなわち，リンク先が未着のリンクでは，リンク元のアンカーは**待ち状態**(waiting state)にあるとみなされ，不完全リンクではない．これを**未着リンク**と呼ぶ．

待ち状態であるアンカーを指定してリンク先へのナビゲーションを行う場合は[4]，そのアンカーに対する指定は一時的に保留状態になる．



	trans. time	valid time
O_{as}	Jan. 09, 05:00	[Jan.09, 05:00, Jan.10, 05:00]
O_{ad1}	Jan. 09, 06:00	[Jan.09, 06:00, Jan.09, 12:00]
O_{ad2}	Jan. 09, 08:30	[Jan.09, 08:30, Jan.09, 14:30]
O_{ad3}	Jan. 09, 08:45	[Jan.09, 08:45, Jan.09, 14:45]

図 5.6: 最新リンクの制御例

待ち状態のアンカーはリストに登録され，リンク先が到着するまで待ち状態となる．複数の待ち

[4]Web のブラウザなどでは，アンカーポイントをクリックすることで指定を行い，リンク先へのナビゲーションを行う方式が一般的である．

状態のアンカーがある場合は，リストに登録されリンク先が到着した順にリストから外され，自動的にナビゲーションが行なわれる．

ここで，解消順序がユーザの指定順に依存する場合も考えられる．すなわち，待ちリストではなく，スタックやキューになる場合である．どのような順序により評価を行うかは，アプリケーションに依存している．

最新リンクの制御

放送型ハイパーメディアでは，情報はバージョンアップされ頻繁に更新される．従って，新しいバージョンが到着した場合に，ユーザが手動で新しい情報へナビゲーションを行なうのではなく，システムが自動的にナビゲーションを行なう機能が必要になる．言い換えると，常に最新のノードに対してリンクを生成する機能である．

前提条件としては，サーバーでは各コンテナのidが管理されていると仮定する．あるidを持つコンテナがバージョンアップされた場合，新しいバージョンは古いバージョンと同じidを付与されて送信される．この時，新しいバージョンは，古いバージョンに比べてトランザクション時間が後になっている．

クライアントでは，既に到着したバージョンのリンクが有効であるかどうかを調べ，有効である場合はリンクを生成する．すなわち，新しいバージョンが到着した場合の手順は以下の通りである：

1. 既に到着した各バージョンにおいて，リンクが有効かを調べ，有効なもののみ候補とする
2. 候補の中で最新のトランザクション時間を持つバージョンをリンク先(リンク元)として決定する

図 5.6の例では，[Jan.09, 09:00]では$O_{a_{d_3}}$が最新リンクのリンク先になっている．

最新リンクは，バージョンアップを自動的に行なう枠組として用いることが可能である．ここでは，同じidを持つコンテナの場合について述べたが，選択する範囲(ドメイン)を指定しておき，その中で最新のものを選択するなどの拡張が考えられる．また,「最新より一つ前」のような相対指定なども考えられる．

5.5 配信コンテンツのバージョン管理

提案する情報配信方式では，サーバ側でコンテナやアンカーが時々刻々と更新され，クライアントに配信される．また，アンカー，コンテナは更新されるごとにそのバージョンをサーバ側およびクライアント側に蓄積する．クライアント側では，サーバにおいて配信されるコンテナやアンカーのバージョンを高々1回しか受信しない．従って，受信できないバージョンがある可能性がある．このような場合に，コンテンツの有効性に矛盾が起きないようにする必要がある．本節では，コンテナやアンカーのバージョン管理の方法について述べる．

5.5.1 サーバ側におけるバージョン管理

バージョン木

サーバ側ではコンテナ，およびアンカーは内容を更新するごとに新しいバージョンを作成する．また，更新されたことによって古いバージョンの有効時間を変更する必要がある場合も存在する．このようなバージョン管理を行うために各コンテナ，各アンカーごとにバージョン木を生成する．バージョン木は2分木であり，ノードが各バージョンを示す．また，あるノードの左の子節点はそのノードの内容が更新された次のバージョンであり，右の子節点は有効時間の変更が行われたバージョンである(図5.7)．

バージョン木Tは以下のように定義される．

$$
\begin{aligned}
T &= (N, E) \\
N &= \{v_1, v_2, \ldots, v_n\} \\
E &= \{e_{li}, e_{rj} | i = 1, 2, \ldots, n, j = 1, 2, \ldots, m\} \\
a &= (id, version, tr, [v_1, v_2]), a \in N \\
C &= (ID, version, Tr, [V_1, V_2], \{a_i | i = 1, 2, \ldots, m\}), C \in N
\end{aligned}
$$

ここで，a, Cはそれぞれアンカー，コンテナを示すノードである．また，id, IDはアンカー，コンテナの識別子であり，$tr, Tr, [v_1, v_2], [V_1, V_2]$はそれぞれアンカーおよびコンテナのトランザクション時間と有効時間である．また，$version$はバージョン番号を表わす．バージョン番号は2つの自然数の組(i, j)で表わされる．バージョン木では，バージョン番号が$(i, 0)$のノードの左の子節点のバージョン番号は$(i+1, 0)$となり，右の子節点は$(i, 1)$となる．バージョン番号が(i, j)のコンテナをC_{ij}，アンカーをa_{ij}と表わす．

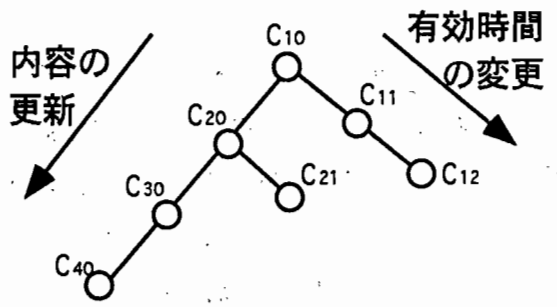


図 5.7: バージョン木

バージョン木に対する基本操作

バージョン木に対する基本操作は次の2つである．内容の更新を行った際の操作である *version_up* と，有効時間の変更を行う *propagate* である．

- *version_up(new-version)*
  あるバージョン木の最新のノード，つまり一番左の葉節点の内容を更新した新しいバージョンのノード *new-version* を左の子として追加する操作．*new-version* のバージョン番号は追加されるノードのバージョン番号を $(i,0)$ とすると $(i+1,0)$ となる．

- *propagate*$(N,[v_1,v_2])$
  C_{N0} の有効時間を $[v_1,v_2]$ に変更したものを新しいノードとして，C_{N0} から右の子を辿っていき，右の子がなくなったらそこに新しいノードを追加する操作．追加するノードのバージョン番号を (i,j) とすると，追加したノードのバージョン番号は $(i,j+1)$ となる．

この2つの操作を用いて，アンカー，コンテナの追加，更新，削除を行う．

コンテナ，アンカーの追加，削除，更新

[I] コンテナの追加　コンテナの追加は，新たなコンテナのバージョン木を作成し，コンテナを配信する．アンカーの追加に関しては5.5.1[III]で述べる．

[II] コンテナの削除　コンテナの削除に関しては，次の2つの場合がある．

1. コンテナ自体の削除
   コンテナ C 自体を時刻 t_d に削除したときは，すべてのバージョンを無効にする必要がある．したがって処理は以下のようになる．

- Cのバージョン木のすべてのバージョンNについて，操作$propagate(N, [v_1, t_d])$を行う(図5.8(a))

2. コンテナのあるバージョンの削除
   コンテナのあるバージョンC_{K0}を時刻t_dに削除する時は，以下のような処理を行う．

   (a) C_{K0}は削除されたために無効にする必要がある．したがって$propagate(K, [v_1, t_d])$を行う．

   (b) 節点C_{k-1}の有効時間の終了時間が*until-changed*[5]の場合は，次の操作を行う．

   - C_{K0}が最新のバージョンではない場合，つまりC_{K0}が存在する場合，$propagate(K-1, [v_1, Tr_{K+1}])$を行う(図5.8(b))．ただし，$Tr_{K+1}$は$C_{K+1,0}$のトランザクション時間である．
   - C_{K0}が最新のバージョンの場合，つまり左の子節点が存在しない場合，$propagate(K-1, [v_1,$ *until-changed*$])$を行う(図5.8(c))．

   図5.8はコンテナの削除の3つの場合を示している．

   (a)はコンテナ自体の削除を示している．削除により，全てのバージョンを時刻t_dで無効にするため，C_{12}, C_{22}, C_{31}を追加している．

   (b)は最新でないバージョンC_{20}の削除を示している．まず，C_{22}を追加して時刻t_dで無効にし，さらに，1つ前のバージョン(C_{10})の有効時間をC_{30}のトランザクション時間T_3に変更している．

   (c)は最新のバージョンC_{20}の削除を示している．まず，C_{21}を追加して時刻t_dで無効にし，さらに，1つ前のバージョン(C_{10})の有効時間を*until-changed*に変更したノードを追加している．

アンカーの削除に関しては5.5.1[III]で述べる．

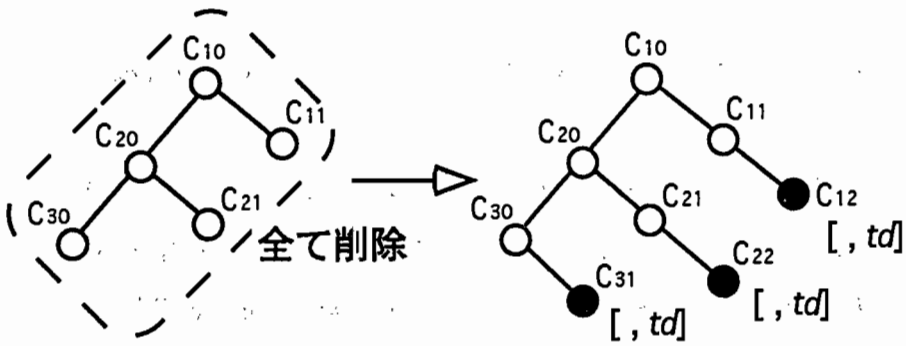
[III] コンテナの更新

コンテナの内容が時刻t_cにC_{N0}から$C_{N+1,0}$に更新された場合は，以下の処理を行う(図5.9(a))．

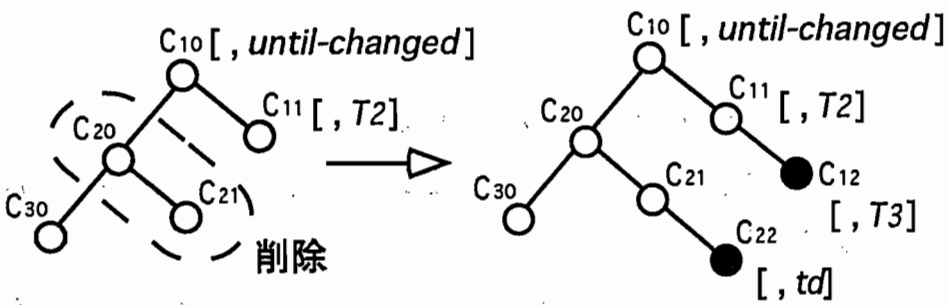
1. 更新するコンテナのバージョン木に対し，*version-up*$(C_{N+1,0})$を行う．ただし，C_{N+1}のトランザクション時間はt_cである．

2. C_Nの有効時間が*until-changed*の場合，$propagate(N, [v_1, t_c])$を行う(図5.9(a))．

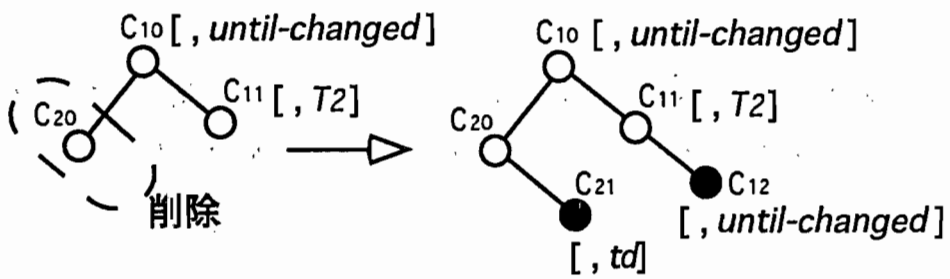
[5] *until-changed*という指定は，次のバージョンが生成されるまで有効であるという，時間変数による指定である．



(a)コンテナ自体の削除



(b)C20の削除（最新のバージョンではない場合）



(c)C20の削除（最新バージョンの場合）

●：追加されたノード

図 5.8: 時刻t_dにおけるコンテナの削除

また，C_{N0} の有効時間を $[v'_1, v'_2]$ に変更する場合は次の操作を行う (図5.9(b))．

1. $propagate(N, [v'_1, v'_2])$ を行い，C_{N0} の有効時間を変更する．
2. C_{N0} の有効時間を変更したことにより，過去のバージョンのすべての有効時間を変更しなければならない場合は，$N < K$ である全ての C_{K0} に対して $propagate(K, [v'_1, v'_2])$ を行う．

図5.9はコンテナの更新の2つの場合を示している．(a) は *until-changed* の場合であり，C_{30} が更新された C_{40} が追加され，また C_{31} を追加し，C_{30} を更新時刻 t_c に無効にしている．
(b) は C_{30} 有効時間の変更を示している．C_{31} を追加することで有効時間の変更を行い，さらに，C_{11}, C_{21} を追加し，古いバージョン全ての有効時間を変更している．

さらに，コンテナの更新の際，コンテナ内のアンカーが追加，削除，更新された場合は，そのアンカーに対して，以下の処理を行う．

- コンテナに新しいアンカーを追加する場合
  新しいアンカーのバージョン木を作成する．
- コンテナからアンカー a_n を削除する場合
  削除されたアンカーのバージョン木に対して，5.5.1のコンテナ自体の削除と同様の処理を行う．
- コンテナ内のアンカー a を更新する場合
  a のバージョン木に対して，コンテナの更新と同様の処理を行う．

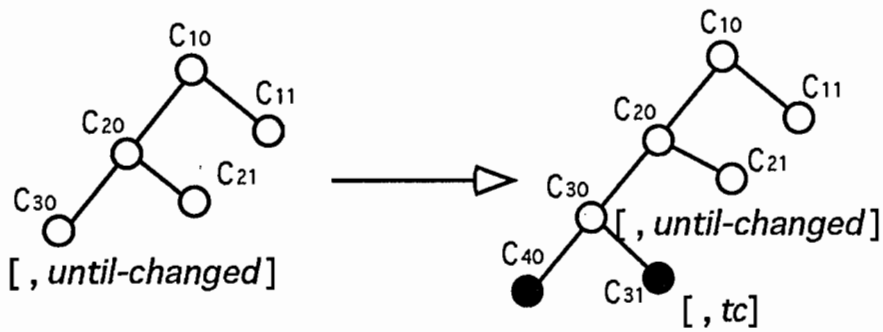
バージョンの配信

サーバは各クライアントにコンテナ，アンカーのバージョン木のノードのうち，クライアントに配信されているノードと，サーバ側でのバージョン木の葉節点との差分のノードを配信する．例えば，図5.11において，サーバは，3番目におけるクライアントに配信済のノード $N_C(t) = \{C_{11}, C_{30}\}$ と，4番目のサーバ側における葉のノード $N_S(t) = \{C_{11}, C_{21}, C_{31}, C_{40}\}$ との差分である $N_S(t) - N_C(t) = \{C_{21}, C_{31}, C_{40}\}$ をクライアントに配信する．

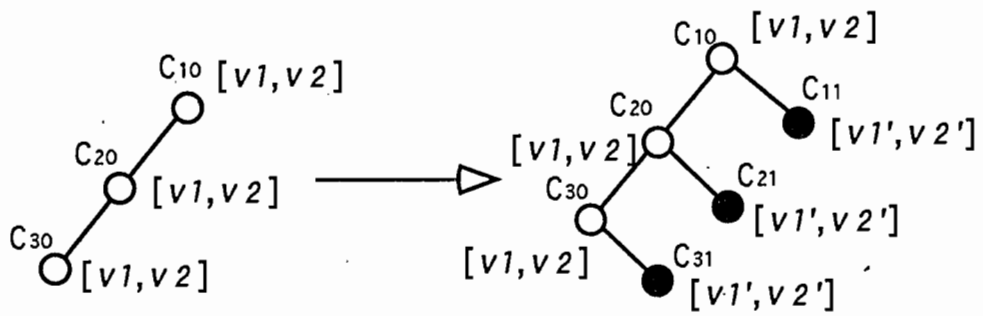
5.5.2 クライアント側におけるバージョン管理

バージョンリスト

クライアント側でのバージョン管理は，各アンカーと各コンテナのバージョンリストを生成することで行う．バージョンリストは線形リストであり，ノードが各バージョンを表す (図5.10)．ク



(a)C30からC40にVersion up



(b)C30の有効時間を変更
(全てのバージョンの有効時間を遡及する場合)

図 5.9: 時刻t_cにおけるコンテナの更新

ライアントは起動中に，サーバ側のコンテナ，アンカーのバージョン木の葉のバージョンを1回だけ受信することができる．

アンカーのバージョンリストl_aは以下のように定義される．

$$\begin{aligned} l_a &= [a_1, a_2, \ldots, a_n] \\ a &= (id, version, tr, [v_1, v_2], ar) \end{aligned}$$

コンテナのバージョンリストl_Cは以下のように定義される．

$$\begin{aligned} l_C &= [C_1, C_2, \ldots, C_n] \\ C &= (ID, version, Tr, [V_1, V_2], Ar, \{a_i | i = 1, 2, \ldots, m\}) \end{aligned}$$

ここで，id，ID，version，トランザクション時間，有効時間はサーバ側のノードと同じものである．ar, Arはそのノードの到着時間である．バージョンリストでは，バージョン番号(i,j)のiは必ずしも連続しているとは限らない．なぜなら，クライアントがシステムを起動していない間に，受信できないノードがあるからである．

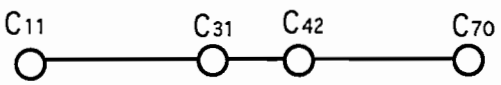


図 5.10: バージョンリスト

バージョンリストに対する操作

クライアント側では，サーバから配信された新しいコンテナ，アンカーのバージョンに対して，以下のいずれかの処理を行う(図5.11)．ここで，配信された新しいノードのバージョン番号を(i, j)とする．

- バージョンリストの最後尾にノードを追加
  現在のバージョンリストのバージョン番号(k, l)のkに対して，$i > k$が成り立つ場合，そのノードをバージョンリストの最後尾に追加する．

- バージョンリストにノードを挿入
  現在のバージョンリストのどのノードのバージョン番号(k, l)のkに対して，$i \neq k$かつ$i < k$である場合は，そのノードをバージョンリストの適切な位置に挿入する．

- バージョンリストのノードを交換

  現在のバージョンリストのノード(バージョン番号(k, l))に対して，$i = k$となるノードがすでに存在する場合は，すでにあるノードと配信されたノードを交換する．交換したノードは，有効時間が正しくないノードなので，破棄する．

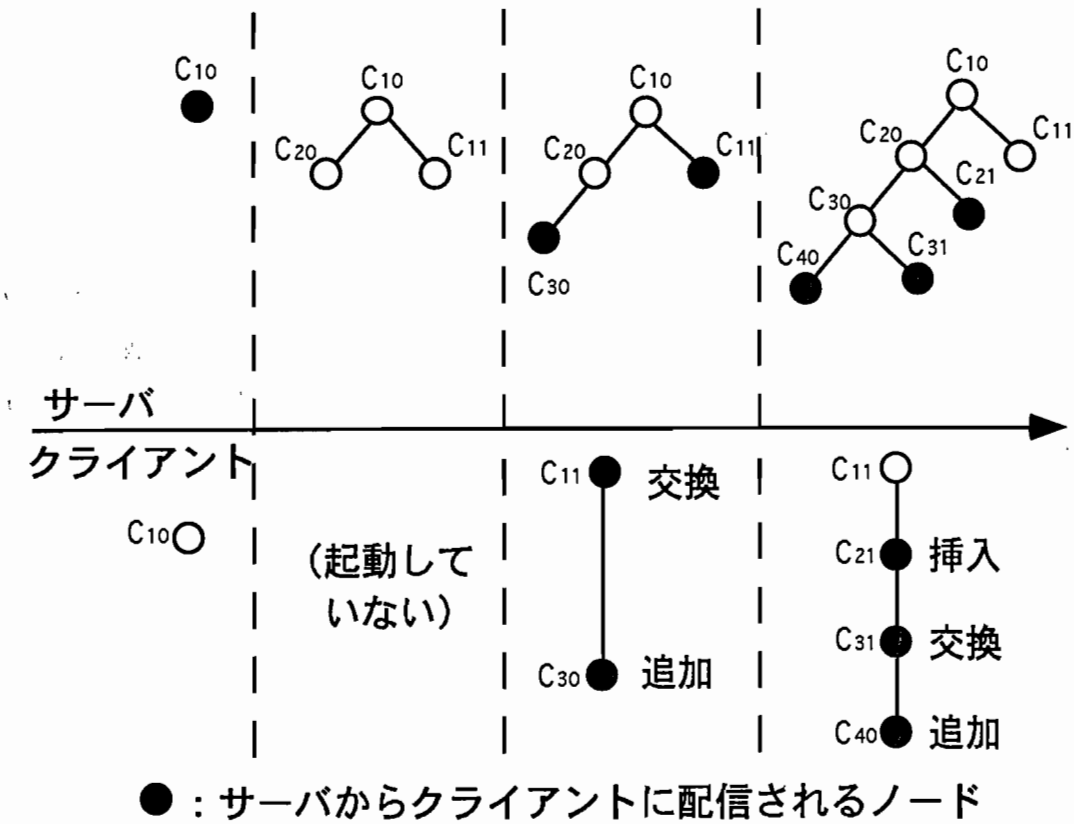


図 5.11: クライアントにおける処理

クライアント側におけるコンテンツの一貫性

クライアント側では，システムを起動していないと受信できない．しかし，サーバ側において有効時間が変更されたノードは，クライアントがシステムを起動したときに，必ずクライアント側に配信される．なぜならば，サーバは，クライアント側のノードとサーバ側での葉のノードとの差分を配信するためである．クライアント側では，そのノードと配信済みのノードを交換することによって，有効時間が正しいノードのみを蓄積する．したがって，受信できないノードはあるが，蓄積されるノードの有効時間は正しい．

5.5.3 リンクの制御方式

サーバにおけるリンク情報の配信

提案する情報配信システムでは，時間依存リンクを，サーバ側でコンテンツとは別に作成し，配信する．この時間依存リンクの情報をリンク情報と呼ぶ．1つの時間依存リンクにつき，1つのリンク情報が作成されるとする．

クライアントは，このリンク情報を受信し，それに基づいてコンテンツ間に時間依存リンクを生成する．サーバ側で作成されるリンク情報$Link$は以下のようになる．

$$Link = (a_s, a_d, v_l, mode)$$

ここで，a_s, a_dはそれぞれソースアンカー，デスティネーションアンカーを表わし，v_lはリンク自体の有効時間を表わす．ただし，リンク自体の有効時間の指定は必ずしも行う必要はない．リンク自体の有効時間の指定を行わなかった場合は，a_s, a_d及びこれらを含むコンテナの有効時間から，リンクの有効時間が計算される．また，$mode$はリンクの制御モードの指定を行う．これについては後述する．

クライアントにおけるリンクの制御

クライアント側では，サーバから配信されたリンク情報を参照し，リンクを生成する．本節では，クライアント側におけるリンクの制御について述べる．

リンクの動的生成 放送型情報提供サービスでは，配信されるデータの送信順序や受信順序が必ずしも各クライアントにおいて一定であるとは限らない．従って，一時的に一部のデータが未着(upcoming)であるような状態が存在する[71]．リンク先のアンカーが未着の場合，リンク元のアンカーは待ち状態になる．待ち状態であるアンカーを指定してリンク先へのナビゲーションを行う場合，そのアンカーに対する指定は一時的に保留状態となる．ここで複数のアンカーが保留状態となっている場合，そのアンカーを評価する順序として，キュー，リスト，スタックのいづれかに格納して評価することが考えられる．この3つのうちどの方法で評価するかはユーザによって指定することが可能であり，アプリケーションに依存すると考えられる．

また，提案する情報配信システムでは，アンカー，コンテナの更新をバージョンとして蓄積するため，リンクのアンカーとして適切な複数のバージョンのアンカーが存在する場合がある．これらのアンカーのうちどれをリンクのアンカーとして決定するかは，リンクの制御モードによって決まる．

リンクの制御モード 提案する情報配信システムでは，以下のようなモードにより，リンクの制御を行う．

- 未着リンクを許すかどうかをきめるモード
  - *upcoming*：デスティネーションアンカーが未着の場合，保留状態とし，デスティネーションアンカーが到着後にリンクを生成する．
- アンカーを決定するためのモード
  - *newest*：アンカーとして，最新バージョンのアンカーをリンクのアンカーとするモード．最新バージョンのアンカーが有効でない場合はリンクを生成しない．
  - *newer*：アンカーとして，有効であるもののうち，最新バージョンのアンカーをリンクのアンカーとしてリンクを生成するモード．

リンクの生成手順 クライアント側におけるリンクの生成手順は以下のようになる．

1. 表示するページの中に該当するソースアンカーの指定があるリンクのうち，その時点で有効であるリンク，および有効時間が付加されていないリンクを選ぶ．
2. 選ばれたリンクのデスティネーションアンカーを計算する．その結果によって，以下のような処理を行う．
   - 該当するデスティネーションアンカーが無かった場合
     リンクの *mode* に *upcoming* が指定されている場合はソースアンカーを待ち状態とし，そのリンクを生成する．指定されていない場合はリンクを生成しない．
   - 該当するデスティネーションアンカーが存在した場合
     リンクに有効時間が付加されているリンクを生成する．リンクに有効時間が付加されていないリンクについては，アンカーとコンテナの有効時間から有効であるかを判定し，有効であるリンクを生成する．

図5.12は，サーバ側におけるコンテンツの生成，更新と，リンク情報の配信，および，クライアント側におけるコンテンツの受信と，リンクの生成，破棄の処理の流れを示している．ここで，図5.12における Link は以下の通りである．

- リンク元, リンク先のアンカーが，それぞれ a_s, a_d である．
- リンク自体の有効時間の指定がない．

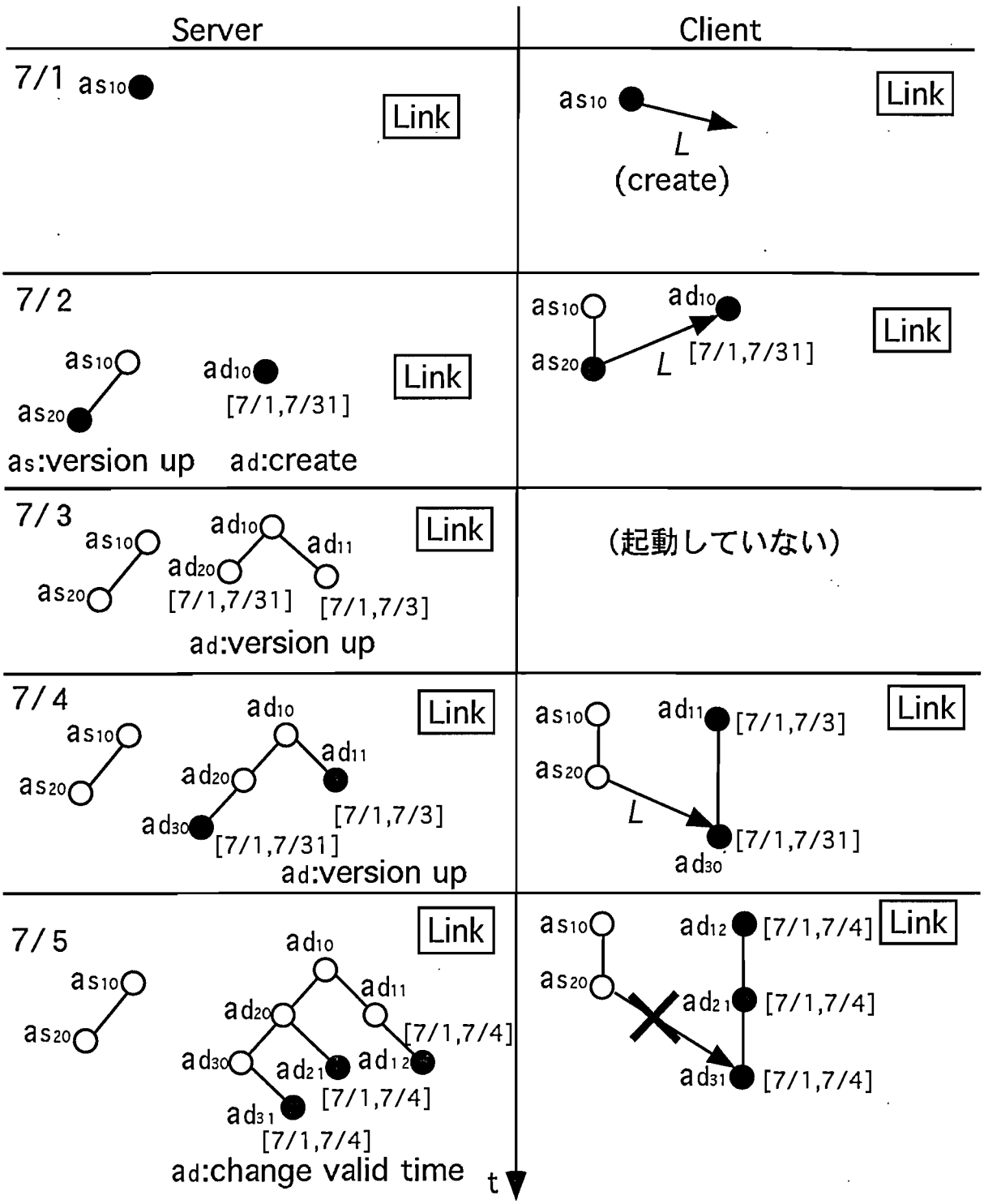


図 5.12: リンクの制御モードの例 (*newest*)

- 制御モードが *upcoming* かつ *newest* である

図5.12の7/1の時点において，サーバ側ではリンク先のアンカーが生成，配信されていないが，クライアント側では *upcomming* モードにより未着リンクとしてリンクが生成される．

7/2には，新しく更新されたノードが配信され，クライアント側では *newest* モードにより，それぞれ最新のバージョンである $a_{s_{20}}, a_{d_{10}}$ 間にリンクが生成される．

7/4において，サーバは，7/2の時点でのクライアントとの差分の葉のノードを配信し，クライアントはそれを受信し処理を行って，最新のバージョンである $a_{s_{20}}, a_{d_{30}}$ にリンクをつなぎかえる．

7/5に，サーバ側で $a_{d_{30}}$ の有効時間の変更と遡及が行われ，変更されたノードを受け取ったクライアントは，$a_{d_{31}}$ が有効でなくなったために，リンクを破棄する．

5.6 プロトタイプシステム *Mille-feuille*

我々は，提案する時間依存リンク機構を用いて情報配信システム *Mille-feuille* [6]を実装している．本システムは情報配信ソフトウェアCastanet 2.0[77]を用いて実装されている．Castanetはプログラムやデータをインターネットを通じて配送するソフトウェアである．このソフトウェアは，サーバー側ではトランスミッタと呼ばれる機能を用いて配信し，クライアント側ではチューナと呼ばれる機能をもちいて受信を行う．この時，サーバーからクライアントに定期的に送信するように設定することで，クライアント側のプログラムやデータを更新することが可能である．

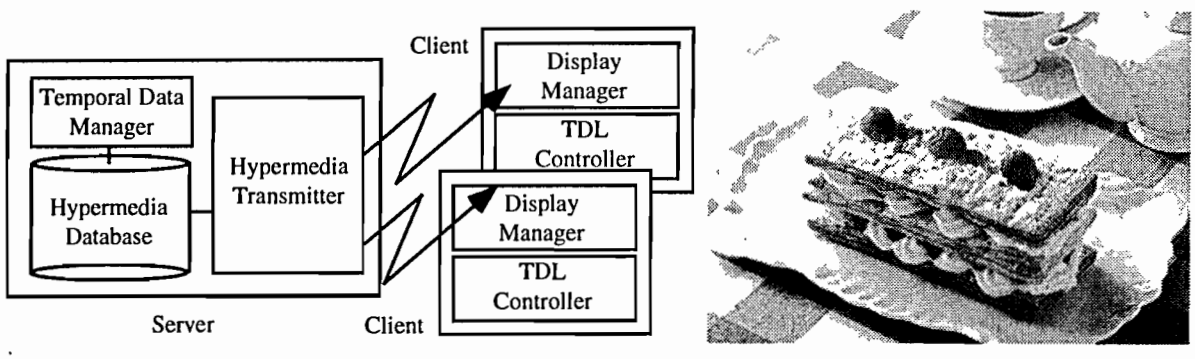


図 5.13: *Mille-feuille*のシステム構成図

本プロトタイプシステムは，サーバーからクライアントへの配信には，Castanetのトランスミッタとチューナの機能を用いている．クライアント側のアプリケーションはJavaのアプレットにより実装されている．また，ユーザインタフェースはビジュアル・インタフェース・ビルダBongo 1.1[83]を用いて実装されている．図 5.13に*Mille-feuille*のシステム構成を示す．サーバーはハイパーメディア・コンテンツ・データベース，時間属性を管理する時間情報管理部，およびハイパーメディア送信部から構成されている．また，各クライアントは時間属性を管理するためのTDL管理部，および表示部から構成されている．本プロトタイプシステムの現バージョンでは，前節で述べたすべての機能は実現されていないが，配信機構，表示機能等の基本機能については実装を完了している．

図 5.14は，*Mille-feuille*を用いてF1レースの実時間情報配信システムを構築した画面例である．画面中，ウインドウはコンテナ，ウィンドウ中の下線部はアンカーを示している．もちろん，アンカーを指示することによりリンク先にナビゲートすることが可能であり，また，URLが指定されている場合は，インターネットを通じてWebのページにアクセスすることも可能である．

ユーザがどのように本システムを使用するかの例を図 5.16 の4つの画面を用いて説明する．

[6]mille-feuille(ミルフィーユ)とは，小さいパイとその間にカスタードクリームが層状に重なった洋菓子のこと．本来の意味は「数千の木の葉」．本研究では，パイをノード，カスタード・クリームをその間のリンクにみたてている．

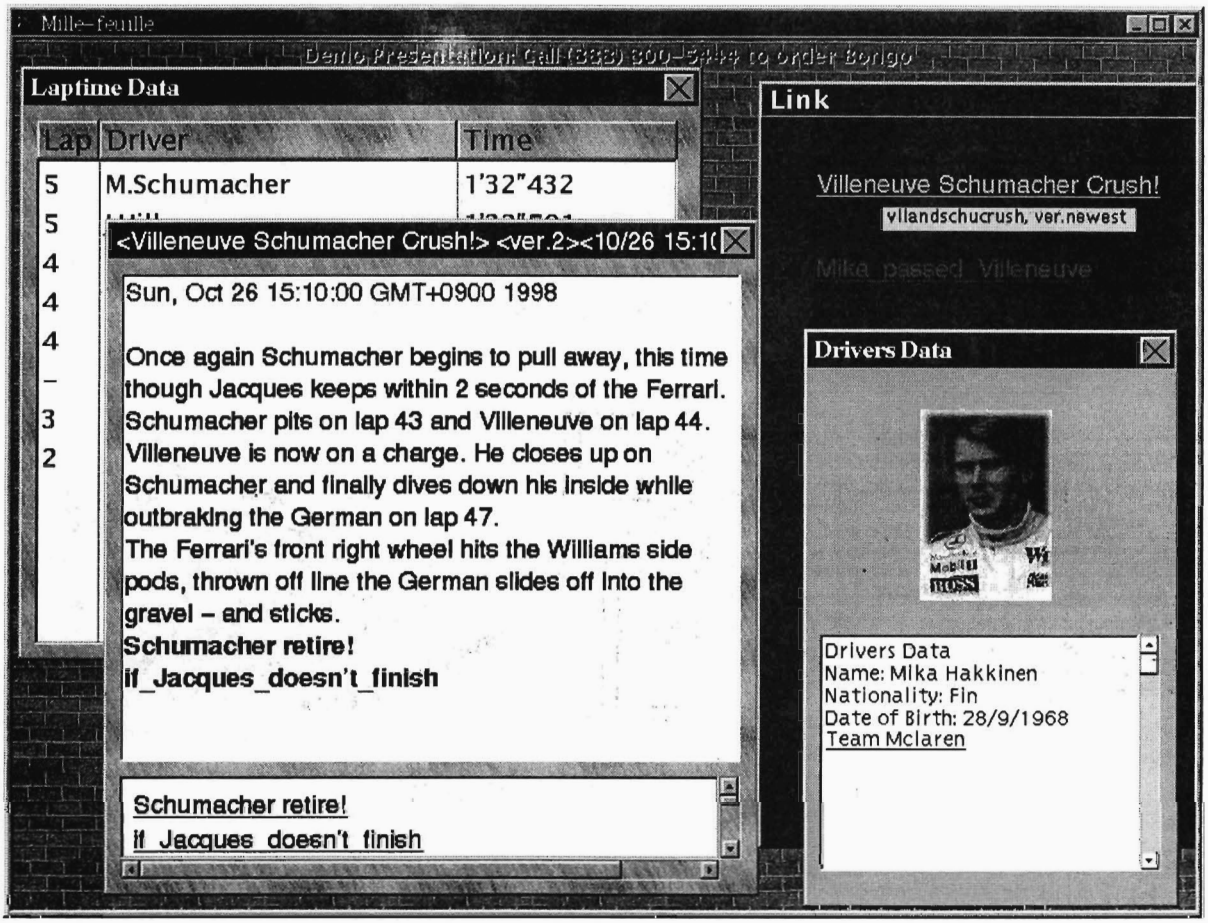


図 5.14: F1レースの情報配信システムの画面

1. 左側と中側のウィンドウは関連情報を示すウィンドウである．右側のウィンドウは時間経過とともに自動的に最新情報を表示する最新情報ウィンドウである．ここでは,「最新のクラッシュ情報」が到着した状態を示している．この時，情報が到着したことを下線部を表示することで示している．このシステムの場合，通常のアンカーは黄色で表示される．

2. ユーザがアンカーを指定(このシステムの場合はクリック)した状態を示している．ただし，この時，リンク先のデータが未着であるため，アンカーの色が変化(赤色)し，待ち状態になるだけであり，ナビゲーションは行われない．

3. リンク先のデータが到着し，同時にアンカーが待ち状態から解放され，ナビゲーションが自動的に起こった状態である．この例では,「クラッシュの詳細情報」へリンクが張られているため，詳細情報ウインドウが自動的に表示される．

4. 時間経過の後,「クラッシュの詳細情報」が更新されたために，自動的に新しい内容が再表

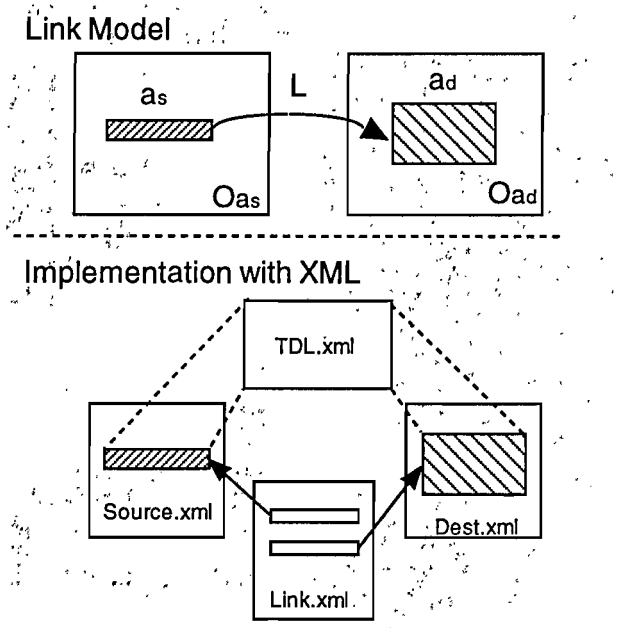


図 5.15: XMLによる時間依存リンクの実現

示されている状態を示す.

5.7 XMLによる時間依存リンクの実現

本節では，提案する時間依存リンクが特定のプラットフォームで実現されるのではなく，従来のハイパーメディアにおいても実現可能であることを示す．ここでは，拡張可能なマーク付け言語(XML: eXtensible Markup Language)[86] による実現について考察する．XML はSGML のサブセットであり，かつ，HTML に拡張機能を追加した言語である．時間依存リンクは，XMLの拡張リンク機構[87]により実現可能である．

図5.15はXMLによるTDLの実現例である．上図はTDLのモデルであり，下図はXML による実現例を示している．ここでは，4つのファイルを用いている．TDLのための拡張情報を定義しているファイル*TDL.dtd*，リンク元を含むファイル*Source.xml*，リンク先を含むファイル*Dest.xml*，リンク情報を含むファイル*Link.xml*である．

*TDL.dtd*では，TDLに必要な属性であるトランザクション時間tt，および有効時間$[vt1, vt2]$をファイルの拡張属性として追加している．以下に記述例を示す．

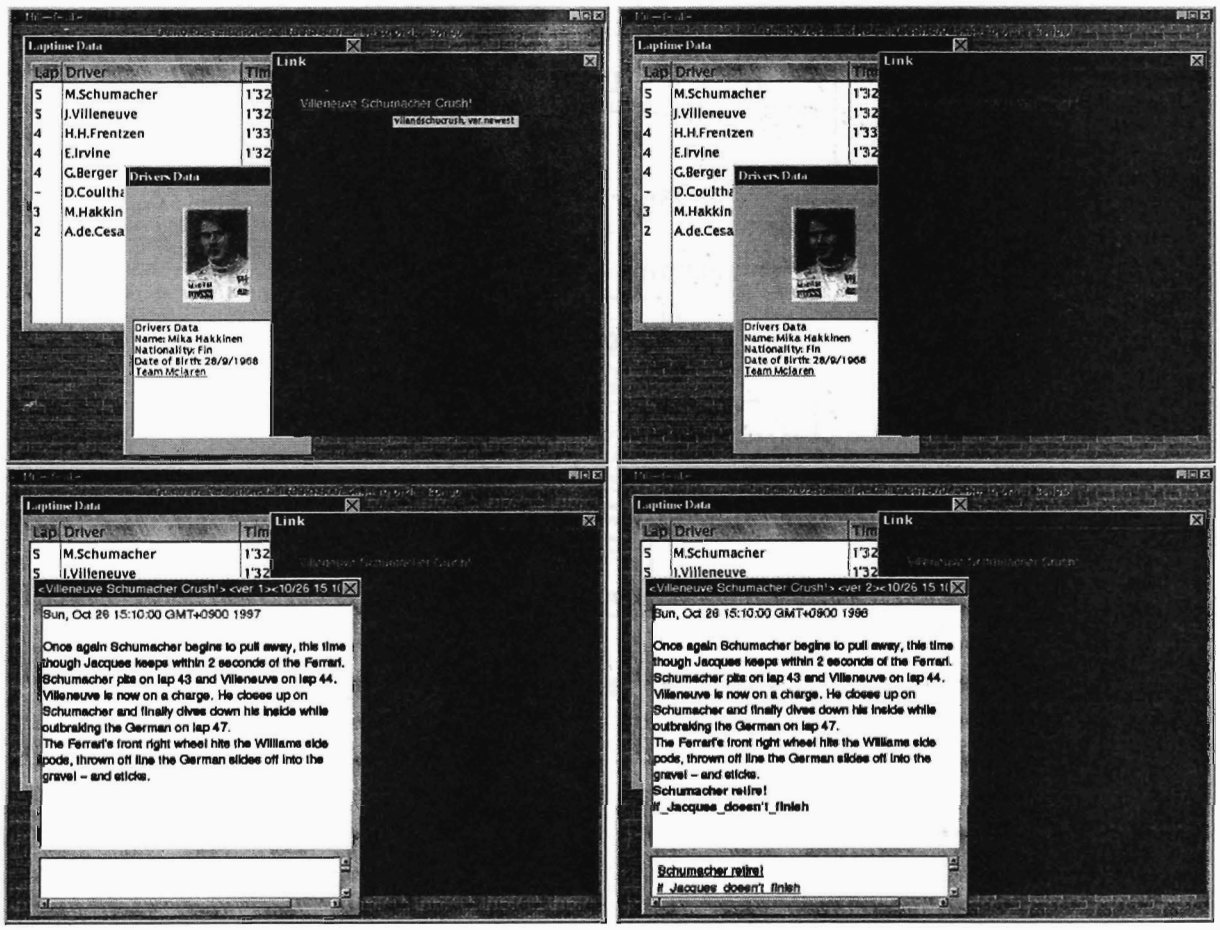


図 5.16: 放送型ハイパーメディアの画面例

```
<!ELEMENT tt PCDATA>
<!ATTLIST tt dt:dt "dataTime.iso8601" #FIXED>
<!ELEMENT vt1 PCDATA>
<!ATTLIST vt1 dt:dt "dataTime.iso8601" #FIXED>
<!ELEMENT vt2 PCDATA>
<!ATTLIST vt2 dt:dt "dataTime.iso8601" #FIXED>
```

*Source.xml*では，この拡張属性の値を設定する．*Dest.xml*でも同様に値を設定する．

```
<?xml version="1.0"? encoding="utf-8"?>
<!DOCTYPE MODEL SYSTEM "file:///TDL.dtd">
    <tt>19980430T20:30:00</tt>
    <vt1>19980501T00:00:00</vt1>
    <vt2>19980525T00:00:00</vt2>
```

Link.xml では，XMLの行外リンクによりリンク元とリンク先の記述を行なう．

```
<TDL-content  XML-LINK="EXTENTED">
   <content1  XML-LINK="LOCATOR"
     HREF="http://www.source.com/Source.xml>
   <content2  XML-LINK="LOCATOR"
     HREF="http://www.dest.com/Dest.xml">
</TDL-content>
```

各属性値の付与はユーザが行なうことも可能であるが，システムにより自動的に付与することも考えられる．トランザクション時間については，データ登録時にシステムがタイムスタンプを付与することで実現可能である．また，有効時間については，デフォルトではすべての時間に対して有効であるという設定にしておくことで，ユーザの設定に関する手間を軽減できる．

5.8 結言

本論文では，ハイパーメディアのための時間依存リンク機構について述べた．本機構はリンクの動的生成・消滅と自動リンク制御機能を実現するものである．従って，内容が動的に変化する放送型ハイパーメディアの実現に適している．また，時間依存リンク方式を用いて，プロトタイプシステム *Mille-feuille* の設計を行い，本方式の有効性を確認した．さらに，本方式がXMLを用いて実現可能であることを示した．

本論文で提案した時間依存リンク方式では，ノードの有効時間からリンクの有効時間を決定したが，リンク自身に有効時間を設定することも考えられる．これについては，本モデルを拡張することで，容易に実現可能である．

今後の課題として，放送型ハイパーメディアにおいて，データの配信時間と到着時間について，その間の遅延や配送順序による一貫性をさらに精密に制御することが挙げられる．また，配信された情報を蓄積しておくことで，時間が経過した後に過去のデータに対する遡及機能について検討する必要がある．

§ 6

結　論

本論文では，マルチメディア・コンピューティング環境におけるコンテンツ管理の課題を解決するために，マルチメディア・データの持つ特性に着目し解決方法を提案した．一般に，マルチメディア・コンテンツの特性として，以下の4つが挙げられる:

1. スキーマを持たず，構造が完全でない (半構造) データが多い
2. 多レベルを持ち，ハイブリッド構造である
3. 時間的に連続性をもつ
4. 分散環境下で用いられるため，有効性が変化する場合がある

本研究では，これらの特性を考慮しマルチメディア・データベースの要件について検討を行った．以下，本研究の総括を行う:

第2章では，ビジュアルプロトタイピングにおけるオブジェクトモデルの構築と検索について述べた．プロトタイピングにおいては，大量の仕様が試行錯誤的に作成されるため効率的な管理機構が必要である．また，スキーマを定期することが困難であり，データ構造を柔軟に扱う枠組みが必要である．本研究では，オブジェクトモデルのクラス階層をスキーマとして，クラスから生成されたインスタンスの集合を仕様のバージョンとしてモデル化を行う方式を提案した．また，記述言語*LAUSIV*の設計について述べた．さらに，大量のバージョンから検索を行う方式を提案した．これらの方式により，スキーマ進化と仕様のバージョンを明確に分離することが可能となるため，版管理における一貫性の保持が実現可能であることを確認した．

第3章では，機器のインタラクション・デザインの枠組みについて述べた．まず，家電機器インタラクション・デザインの要件について考察を行った．本研究では，インタラクション・デザインモデルとして，ユーザインタフェースと主機能を分離した機能操作制約モデル (FICモデル) を

提案した．また，このモデルに基づいて家電製品インタラクションデザイン支援システム *Visual CASE*の設計について述べた．このシステムにより，ビジュアルシミュレーションの枠組みを提供することが可能である．さらに，本システムを実開発へ適用した結果，設計工程を約1/5に短縮することが可能であることが分かった．最後に，家電ソフトウェアをインタラクション・デザインの観点から部品化を行う手法，および，実製品を用いた分析・検討について述べた．

第4章では，デジタル衛星放送システムを用いて送信された大量データの構造化について述べた．まず，一定周期ごとに連続して送信されるテキストデータに対して，連続的に問い合わせを行うことにより、ユーザの所望するデータを検索する方法について議論を行った．本研究では，電子番組ガイド (EPG: Electronic Program Guide) を対象データとし，Virtual Channel の設計について述べた．また，デジタル放送における，インタラクティブ・データ配信のためのカルーセル方式による配信について議論を行った．

第5章では，放送型のハイパーメディアにおいて，情報間の参照情報であるリンクを管理する方式について述べた．本研究では，時間の経過に伴って変更される情報，時系列データ，およびリアルタイム性を持つ情報について時間依存であるリンクに対して時間的一貫性を保証する枠組みについて議論を行った．また，サーバー側とクライアント側のバージョン管理について述べた．さらに，提案する方式に基づいて設計したプロトタイプシステム *Mille-feuille*の実装について述べた．また，本方式は，拡張可能なマーク付け言語(XML)を用いても実装可能であることを確認した．

参考文献

[1] V. Scott Gordon and James M. Bieman, "Rapid prototyping: Lessons learned," IEEE Software, vol.12, no.1, pp.85–95, January 1995.

[2] Y. Imai, K. Sumiya, K. Yasutake, and S. Haruna, "Visual CASE: A Software Development System for Home Appliances," Proc. of the IEEE 17th Int'l Computer Software and Applications Conference(COMPSAC'93), Phoenix, AZ, U.S.A, pp.11–18, November 1993.

[3] K. Itoh, Y. Tamura, and S. Honiden, "TransObj: Software prototyping environment for real-time transaction-based software system applications," Int'l Journal of Software Engineering and Knowledge Engineering, vol.2, no.1, pp.5–30, March 1992.

[4] Halskov Kim and H. Peter Aiken, "Experiences Using Cooperative Interactive Storyboard Prototyping," Communications of the ACM, vol.36, no.4, pp57–64, 1993.

[5] R. H. Katz, "Toward a Unified Framework for Version Modeling in Engineering Databases," ACM Computing Surveys, vol.22, no.4, pp.375–408, December, 1990.

[6] 飯島正，永田守男, "実世界と形式的記述の接点としてのオブジェクト指向モデル," 情報処理, vol.35, no.5, pp.412–422, 1994.

[7] 本位田真一，山城明宏，"オブジェクト指向分析・設計," 情報処理, vol.35, no.5, pp.392-401, 1994.

[8] 貫名康之，内山亘，藤井裕幸，大村優子，岩本恵子，田中裕彦, "W滝洗い洗濯機," National Technical Report, vol.41, no.1, pp.3-9, February, 1995

[9] Sylvia L. Osborn, "The Role of Polymorphism in Schema Evolution in an Object-Oriented Database," IEEE trans. of Knowledge and Data Engineerings, vo.1, no.3, pp.310–317, September 1989.

[10] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, "Object-Oriented Modeling and Design," Prentice Hall, 1991(羽生田栄一監修訳: オブジェクト指向方法論 OMT, プレンティスホールトッパン, 1992).

[11] E. Sciore, "Multidimensional Versioning for Object-Oriented Databases," Proc. of Second International Conference on Deductive and Object-Oriented Databases, pp.355-370, December 1991.

[12] Nan C. Shu, "Visual Programming," Van Nostrand Reinhold, 1988(西川博昭訳: ビジュアル・プログラミング, 日経BP社, 1991).

[13] 今井良彦, 角谷和俊, 安武剛一, 田中裕彦, 春名修介, "機能操作制約モデルに基づく家電機器インタラクションデザイン支援システム," 情処学論, vol.36, no.8, pp.1938-1950,1995.

[14] K. Sumiya, K. Yasutake, H. Tanaka, N. Sanada, and Y. Imai, "A Product Specification Database for Visual Prototyping," Proc. of 21st Very Large Data Bases(VLDB'95), pp.666-676, September 1995.

[15] Sybase, "Gaim Momentum User's Guide," Sybase, Inc, 1994.

[16] 田中裕彦, 安部秀二, 内山亘, 石崎恵美子, 縄田毅史, 今井良彦, "家電操作仕様プロトタイピングシステム –操作パネル設計における適用事例 –", 第14回ソフトウェア生産における品質管理シンポジウム, pp.9–16, 日本科学技術連盟, 1994.

[17] K. Tanaka, S. Nishio, M. Yoshikawa, S. Shimojo, J. Morishita, and T. Jozen, "Obase object database model: towards a more flexible object-oriented database system," Proc. of the Int'l. Sympo. on Next Generation Database Systems and Their Applications, pp.159–166, September 1993.

[18] R. Zicari, "A framework for schema updates in an object-oriented database system," Proc. of Int'l. Conf. on Data Engineering, pp.2–13, 1991.

[19] 青木淳: オブジェクト指向システム分析設計入門, SRC, 1993.

[20] Coutaz, J.: *The Construction of User Interfaces and the Object Paradigm*, Proc. E-COOP87, 1987.

[21] Austin,D.Henderson Jr: *The Trillium User Interface Design Environment*, Proc. of CHI 86, pp.221–227, 1986.

[22] Hatley, D. J., Pirbhai, I. A.: リアルタイム・システムの構造化分析, 日経BP社, 1989.

[23] Hirakawa, M., Tanaka,M., and Ichikawa,T.: *An Iconic Programming System, HI-VISUAL*, IEEE Trans.Softw.Eng., Vol. 16, No. 10, pp. 1178–1184, 1990.

[24] 伊藤潔, 本位田真一, 内平直志: プロトタイピングツール, 啓学出版, 1987.

[25] Imai, Y., Sumiya, K. , and Yasutake, K., Haruna,S.: *Visual CASE: A Software Development System for Home Appliances*, Proc. 17th COMPSAC, pp.11–18, 1993.

[26] 平成4年度家電製品の操作性向上に関する調査研究, 社団法人 日本機械工業連合会, 財団法人 家電製品協会, 1992.

[27] 黒須正明: ヒューマンインタフェースのデザイン, 情報処理, Vol. 34, No. 8, pp.1063–1072, 1993.

[28] Madsen,K.H.,Akiken,P.H.: *Experiences using cooperative interactive storyboard prototyping*, Comm.ACM, Vol. 36, No. 6, pp.57–64, June 1993.

[29] MacQueen,D.B.: *Using dependent types to express modular structure*: Proc. ACM Symposium of Principles of Programming Languages, pp.277–286, 1986.

[30] 垣内隆志, 宮部義幸, 南方郁夫, 東昭和: オブジェクト指向アプリケーション開発環境ActivePage, 情報処理学会研究会報告, 94-DBS-100, pp.177–184, 1994.

[31] Moggridge,B.: インタラクションデザインの出発点 - メタファを持つインターフェイス, AXIS, Vol. 37, pp.58–61, 1990.

[32] Ohori,A. and Buneman,P.: *Type Inference in a database programming language*, Proc. ACM Conference on LISP and Functional Programming, pp.174–183, 1988.

[33] Ohnishi,A.: *A Visual Software Requirements Definition Method*, Proc. of 1st ICRE, pp.194–201, 1994.

[34] ランボー,J., ブラハ,M., プレメラニ,W., エディ,F., ローレンセン,W.: オブジェクト指向方法論OMT, トッパン, 1992.

[35] 角谷和俊, 眞田紀男, 春名修介, 今井良彦: 家電ソフトウェア設計支援システムVisual CASEの開発, 情報処理学会研究会報告, 93-SE-91, pp.9–16, 1993.

[36] 田中裕彦，安部秀二，内山亘，石崎恵美子，縄田毅史，今井良彦: 家電操作仕様プロトタイピングシステム –操作パネル設計における適用事例 –, 第14回ソフトウェア生産における品質管理シンポジウム, pp.9–16, 日本科学技術連盟, September 1994.

[37] 田中克己，西尾章治郎，吉川正俊，下條真司，上善恒雄: Obase: 動的継承とアクティブルール機構を有するインスタンス指向オブジェクトデータベースシステム, 情報処理学会研究会報告, 94-DBS-100, pp.87–96, 1994.

[38] 田中譲: 次世代情報処理のプラットホームとしてのシンセティック・メディア・アーキテクチャ, 情報処理学会研究会報告, HI-41-4, pp.25–31, 1992.

[39] David Harel. On visual fromalisms. *Communications of the ACM*, Vol. 31, No. 5, pp. 514–530, May 1988.

[40] Yoshihiko Imai, Kazutoshi Sumiya, Kouichi Yasutake, et al. Visual CASE: An Software Development System for Home Appliances. In *Proc. of Compsac'93*, pp. 11–18. IEEE Computer Society, November 1993.

[41] Kim Halskov Madsen and Peter H. Akiken. Experiences using cooperative interactive storyboard prototyping. *Communications of the ACM*, Vol. 36, No. 6, pp. 57–64, June 1993.

[42] 眞田紀男, 角谷和俊, 今井良彦. Visual CASE:家電ソフトウェア設計支援システム. 電子情報通信学会秋期大会, pp. 369–370. 電子情報通信学会, September 1993.

[43] 田中克己. Obaseオブジェクトモデルの基本設計. Obaseプロジェクト第二期研究成果報告書, pp. 165–184. Obaseコンソーシアム, 1992.

[44] European Broadcasting Union. Digital video broadcasting (DVB) systems. *http://www.dvb.org/dvb_index.html.*

[45] Hirohi Fujiwara. *MPEG Textbook.* ASCII, 1995.

[46] Jorgen Rosengren. Electronic programme guides and service information. *Philips Journal of Research*, 50(1/2):253–265, 1996.

[47] K. Tanaka, N. Nishikawa, S. Hirayama, and K. Nanba. Query pairs as hypertext links. In *proceedings of IEEE International Conference on Data Engineering(ICDE'91)*, pages 456–463. IEEE, May 1991.

[48] Takashi Satou and Masao Sakauchi. Video acquisition on live hypermedia. In *proceedings of IEEE International Conference on Multimedia Computing and Systems(ICMCS '95)*, pages 175–181. IEEE, May 1995.

[49] Richard Snodgrass. Temporal databases. *IEEE Computer*, 19(9):35–42, 1986.

[50] T. Imielinski, S. Viswandathan, and B. Badrinath. Energy efficient indexing on air. In *proceedings of the ACM SIGMOD Annual International Conference on Management of Data*, pp. 25–36, May 1994.

[51] DAVIC 1.2 Specification. Digital Audio Video Council, 1996. http://www.davic.org.

[52] A. Antoniazzi and G. Schapeler. An open software architecture for multimedia consumer terminal. In *proceedings of the Second European Conference on Multimedia Applications, Services and Techniques (ECMAST'97)*, pp. 621–634, May 1997.

[53] ISO/IEC 13522. Information technology - coding of multimedia and hypermedia information: Part 5 - support for base level interactive applications. Technical report, IS, 1996. (MHEG5).

[54] ISO/IEC 13818-6. Generic coding of moving pictures and associated audio information - part 6: Extension for digital storage media command and control(DSM-CC), 1997.

[55] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. Broadcast disks: Data management for asymmetric communication environment. In *proceedings of the ACM SIGMOD Annual International Conference on Management of Data*, pp. 199–210, May 1995.

[56] DVB-SI. Digital broadcasting systems for television, sound and data services: Specification for service information (SI) in digital video broadcasting (DVB) system. Technical report, ETSI ETS 300 468, TM1217, October 1995.

[57] S. Gibbs, C. Breitendeder, and D. Tsichritzis. Audio/video databases: An object-oriented approach. In *proceedings of the International Conference on Data Engineering*, pp. 381–390, April 1993.

[58] European Broadcasting Union, "Digital Video Broadcasting (DVB) Systems," *http://www.dvb.org/dvb_index.html.*

[59] 日経BP社, "プッシュ技術最前線," 日経インターネットテクノロジー, no.10, pp.95–111, Oct. 1997.

[60] 上林弥彦(編著), 有川正俊, 國島丈生, 木寛新一, 高田秀志, 宮部義幸, "ハイパーメディアとオブジェクトベース", 分散協調メディアシリーズ -4-, 共立出版, 1995.

[61] Kate Gerwig, "The push technology rage...so what's next?," ACM netWorker: The Craft of Network Computing, vol.1, no.2, pp.13—17, July/Aug. 1997

[62] Calton Pu and Ling Liu, "Update Monitoring: The CQ project," Proc. of the 2nd International Conference on Worldwide Computing and Its Applications - WWCQ'98, Tsukuba, Japan, Lecture Notes in Computer Science 1368, Springer, pp.396-411, 1998.

[63] Jemes Mayfield, "Two-level models of hypertext," Lecture Notes in Computer Science, 1326, Springer, pp.90–108, 1997.

[64] Michael Biever, "Issues in modeling a "dynamic" hypertext interface for non-hypertext systems," Proc. of ACM Hypertext '91, pp.203–217, Dec. 1991.

[65] Frank Wm. Tompa, G. Elizabeth Blake, and Darrell R. Raymond, "Hypertext by link-resolving components," Proc. of ACM Hypertext '93, pp.118–130, Nov. 1993.

[66] Daniel T. Chang, "HieNet: A user-centered approach for automatic link generation," Proc. of ACM Hypertext '93, pp.145–158, Nov. 1993.

[67] Nipon Charenkitkarn, Jim Tam, Mark H. Chignell, and Gene Golovchinsky, "Browsing through querying: Designing for electronic books," Proc. of ACM Hypertext '93, pp.206–216, Nov. 1993.

[68] K. Tanaka, N. Nishikawa, S. Hirayama, and K. Nanba, "Query pairs as hypertext links," Proc. of the 7th IEEE Int' Conf. on Data Engineering, pp.456–463, Apr. 1991.

[69] Daniel Cunliffe, Carl Taylor, and Douglas Tudhope, "Query-based navigation in semantically indexed hypermedia," Proc. of ACM Hypertext '97, pp.87–95, Nov. 1997.

[70] 角谷和俊, 野田玲子, 田中克己, "時間依存リンクを用いた情報配信システム Mille-Feuille の設計," 電子情報通信学会第9回データ工学ワークショップ (DEWS'98), 1998

[71] Kazutoshi Sumiya, Reiko Noda, Katsumi Takana, "Hypermedia Broadcasting with Temporal Links," 9th International Conference on Database and Expert Systems Applications (DEXA), Aug. 1998 (to appear)

[72] Network Working Group, "Hypertext Transfer Protocol – HTTP/1.0 RFC(Request for Comments)", 1945.

[73] J. E. Pitkow and R. K. Jones, "Supporting The Web: A Distributed Hyperlink Database System," Proc. of 5th Int'l World Wide Web Conference, May. 1996.

[74] Alexa, "Alexa version 1.3", *http://www.alexa.com/.*

[75] Jim Rapoza, "Alexa's theory of relativity - filtering, analytical algorithms link to Web sites – relevant or not," PC Week on line, Aug. 1997. http://www.zdnet.com/pcweek/reviews /0818/18alexa.html.

[76] PointCast, "PointCast Network," *http://www.pointcast.com/ main.html.*

[77] Laura Lemay, "Official Marimba Guide to Castanet," Sams.net, Indianapolis, IN 46268, USA, 1997(松田晃一ほか訳：Marimba オフィシャルガイド Castanet, トッパン, 1997).

[78] Rfank Halasz and Mayer Schwartz, "The Dexter Hypertext Reference Model," Communications of the ACM, vol.37, no.2, pp.30–39, Feb. 1995.

[79] L. Hardman, D.C.A. Bulterman, and G. V. Rossum, "The Amsterdam Hypermedia Model: Adding time and context to the Dexter model," Communications of the ACM, vol.37, no.2, pp.50-62, Feb. 1995.

[80] Nitin N. Sawhney, David Balcom, and Ian Smith, "HyperCafe: Narrative and Aesthetic Properties of hypervideo," Proc. of ACM Hypertext '96, pp.1–10, Mar. 1996.

[81] Abdullah Uz Tansel, James Clifford, Shashi Gadia, Sushil Jajodia, Arie Segev, and Richard Snodgrass, "Temporal databases - Theory, Design, and Implementation" The Benjamin/Cummings Publishing, 1993.

[82] James Clifford, Curtis E. Dyreson, Tomás Isakowitz, Christian S. Jensen, and Richard T. Snodgrass, "On the semantics of "now" in databases," ACM Trans. on Database Systems, vol.22, no.2, pp.171–214, Jun. 1997.

[83] Danny Goodman, "Official Marimba Guide to Bongo," Sams.net, Indianapolis, IN 46268, USA, 1997(石川和也訳：Marimba オフィシャルガイド Bongo, トッパン, 1997).

[84] Richard Thomas Snodgrass, "The TSQL2 temporal query language" Kluwer Academic Publishers, 1995.

[85] Wendy Hall, Hugh Davis, and Gerard Hutchings, "Rethinking Hypermedia" Kluwer Academic Publishers, 1996.

[86] World Wide Web Consortium (W3C), "EXtensible Markup Language (XML)," *http://www.w3.org/TR/REC-xml.*

[87] World Wide Web Consortium (W3C), "XML Linking Language (XLink) Draft 3," *http://www.w3.org/TR/WD-xlink.*

謝辞

本研究の遂行ならびに論文の作成にあたり，懇切なる御指導を賜りました神戸大学大学院自然科学研究科(情報メディア科学専攻) 田中克己教授に謹んで感謝の意を表します．本論文をまとめるにあたり，有益な御助言と御教示を賜りました神戸大学工学部 北村新三学部長，金田悠紀夫教授，瀧和男教授に謝意を申し上げます．

本論文の作成にあたり，御配慮を賜わった松下電器産業株式会社 三木弼一取締役，マルチメディア開発センター 櫛木好明所長，坂尾隆技監，宮部義幸チームリーダーに深く感謝致します．

本研究の遂行にあたり，御助言ならびに御協力を頂きました松下電器産業株式会社 AVC商品開発研究所 今井良彦部長，AVCソフト開発推進センター 大津隆史チームリーダー，半導体開発本部 安武剛一主任技師，マルチメディア開発センター 中島不二雄主担当，津賀一宏チームリーダー，春名修介チームリーダー，岡村和男チームリーダー，喜納久行主任技師，垣内隆志主任技師，楠見雄規主任技師に感謝致します．

本研究の実用化推進に際してシステムの開発や製品への適用に御協力いただいた安倍秀二氏，伊藤謙次氏，岩本恵子氏，内山亘氏，平石輝彦氏，石崎恵美子氏，佐野雅章氏，高坂桂子氏，田中裕彦氏，眞田紀男氏，縄田毅史氏，關口卓也氏，菱田利浩氏，川端智氏に謝意を申し上げます．また，論文の作成にあたり，御協力いただいた松下電器産業株式会社 マルチメディア開発センター情報第2チーム（旧情報第7チーム，旧基盤技術第2開発室）の皆様，浜田優子氏，Timothy Cornish氏，および，神戸大学工学部 情報知能工学科 田中研究室の皆様に感謝致します．

最後に，日頃より研究活動を支えてくれた家族，父母 角谷和勇・育子，義父母 佐藤克朗・千枝子，祖母 片岡梅代・佐藤礼子・大塩末子，および，角谷昌美・真歩に感謝します．

研究業績

主要論文

1. Kazutoshi Sumiya, Kouichi Yasutake, Takashi Ohtsu, and Yoshihiko Imai, Visual CASE: An Object-Oriented Software Development System for Home Appliances", Proc. of Int'l Conferece on Technology of Object-Oriented Languages and Systems (TOOLS-USA'93), August 1993, pp. 97-107.

2. 角谷和俊, 安武剛一, 眞田紀男, 春名修介, 今井良彦, 製品仕様データベースにおける操作に基づく検索機能, 情報処理学会アドバンスト・データベース・シンポジウム, 1992年12月, pp. 53–61

3. Yoshihiko Imai, Kazutoshi Sumiya, Kouichi Yasutake, and Shusuke Haruna, "Visual CASE: A Software Development System for Home Appliances", Proc. of the IEEE 17th Int'l Computer Software and Applications Conference (COMPSAC'93), November 1993, pp. 11–18.

4. 今井良彦，角谷和俊，安武剛一，田中裕彦，春名修介，機能操作制約モデルに基づく家電機器インタラクションデザイン支援システム, 情報処理学会論文誌, vol. 36, no. 8, 1995年8月, pp. 1938-1950.

5. Kazutoshi Sumiya, Kouichi Yasutake, Hirohiko Tanaka, Norio Sanada, and Yoshihiko Imai,"A Product Specification Database for Visual Prototyping", Proc. of 21st Very Large Data Bases (VLDB'95), August 1995, pp. 666-676.

6. Kazutoshi Sumiya, Yoshihiko Imai, Yoshiyuki Miyabe, and Yahiko Kambayashi, "A Visual Prototyping and Software Development System for Home Appliances", Workshop on New Paradigms in Information Visualization and Manipulation (in conjunction with the ACM CIKM95), November 1995.

7. 角谷和俊，安武剛一，田中裕彦，宮部義幸，今井良彦, ビジュアルプロトタイピングのための製品仕様オブジェクトモデル, 電子情報通信学会論文誌 vol. J79-D-I, no. 10, pp. 884-894, 1996年10月.

8. Kazutoshi Sumiya and Katsumi Tanaka, "Virtual Channel: Dynamic Structuring and Continuous Queries for Data on the Air", Proc. of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing(Pacrim'97), August 1997, pp. 715-720.

9. 角谷和俊, 楠見雄規, 岡村和男, ディジタル放送インタラクティブ・データ配信のためのカルーセル型送出方式DVXとその応用, 情報処理学会アドバンスト・データベース・シンポジウム, 1997年12月, pp. 23–30

10. Kazutoshi Sumiya, Reiko Noda, and Katsumi Takana, "Hypermedia Broadcasting with Temporal Links", Proc. of 9th International Conference on Database and Expert Systems Applications (DEXA'98), August 1998, pp. 176–185

11. Shinji Nabeshima, Kazuo Okamura, Takeshi Kakiuchi, Kazutoshi Sumiya, Naoya Takao, Yoshiyuki Miyabe, "Extended Digital Video Broadcasting with Time-lined Hypermedia", 1st Int'l Conference on Advanced Multimedia Content Processing(AMCP'98), November 1998 (to appear)

12. 角谷和俊，野田玲子，田中克己, 放送型ハイパーメディアのための時間依存リンク機構, 電子情報通信学会論文誌D-I, 1999年1月 (掲載予定)

学術講演

1. 角谷和俊, 家電ソフトウェア設計支援システムVisual CASEの開発, 京都大学大型計算機センター研究セミナー「ビジュアルなソフトウェア開発」第36回研究セミナー報告, 1993年3月, pp. 5–13

2. 角谷和俊, 安武剛一, 向井雅樹, 眞田紀男, 田中裕彦, 春名修介, 今井良彦. 家電ソフトウェア設計支援システム Visual CASE. 電気関係学会 関西支部連合大会講演論文集, 1992年11月, p. S4

3. 眞田紀男, 角谷和俊, 今井良彦, Visual CASE: 家電ソフトウェア設計支援システム, 電子情報通信学会秋期大会, 1993年9月, pp. 369–370

4. 縄田毅史, 石崎恵美子, 角谷和俊, 田中裕彦, 家電製品操作仕様プロトタイピングシステム Visual CASE, 95年度システム制御情報学会研究発表講演会, 1995年4月

5. 菱田利浩, 福宮英二, 縄田毅史, 角谷和俊, 田中裕彦, 家電ソフトウェア設計支援システム「VisualCASE」における仕様検証機構, 第49回情報処理学会全国大会, 1994年9月

学術報告

1. 角谷和俊, 眞田紀男, 春名修介, 今井良彦, 家電ソフトウェア設計支援システム Visual CASE の開発, 情報処理学会ソフトウェア工学研究会, vol. 91 pp. 1993年3月, pp.9–16

2. 眞田紀男, 關口卓也, 角谷和俊, 安武剛一, 今井良彦, 伊藤謙次, 家電ソフトウェアの部品化手法, 電子情報通信学会ソフトウェアサイエンス研究会, vol. SS94–11, 1994年5月, pp.81-88

3. 角谷和俊, 安武剛一, 田中裕彦, 縄田毅史, 今井良彦. ビジュアル・プロトタイピングのための製品仕様データベース. 情報処理学会データベースシステム研究会, 1994年5月

4. 角谷和俊, 野田玲子, 田中克己, 時間依存リンクを用いた放送型情報提供システム Mille-feuille の設計, 電子情報通信学会 第9回データ工学ワークショップ, 1998年3月

5. 野田玲子, 馬 強, 角谷和俊, 田中克己, 放送型情報提供システム Mille-feuille における時間依存情報の配信, 情報処理学会データベースシステム研究会, vol. 98, no. 57, 98-DBS-116(1), 1998年7月, pp.103–110

6. 川口知昭, 土居明弘, 角谷和俊, 田中克己, 放送型ネットワークにおけるライブ映像ストリームの編集と配信, 情報処理学会データベースシステム研究会, vol. 98, no. 57, 98-DBS-116(1), 1998年7月, pp.111–118

主要特許

1. ソースファイル管理装置[出願 特 平成 01-283268 (H01.10.30)] [公開 特 平成 03-144726 (H03.06.20)]

2. プログラム自動生成装置及び方法[出願 特 平成 01-283269 (H01.10.30)] [公開 特 平成 03-144728 (H03.06.20)]

3. ソフトウエア性能測定装置及びその測定方法[出願 特 平成 01-344446 (H01.12.27)] [公開 特 平成 03-201044 (H03.09.02)]

4. テスト仕様生成装置および方法[出願 特 平成 04-169048 (H04.06.26)] [公開 特 平成 06-12285 (H06.01.21)]

5. データベース検索処理方法とその装置[出願 特 平成 04-298290 (H04.11.09)] [公開 特 平成 06-149892 (H06.05.31)]

6. テスト支援装置およびその方法[出願 特 平成 04-298291 (H04.11.09)] [公開 特 平成 06-149620 (H06.05.31)]

7. 自動実行装置および自動実行方法[出願 特 平成 04-300166 (H04.11.10)] [公開 特 平成 06-149618 (H06.05.31)]

8. プログラム部品実行方式とプログラム部品検査装置[出願 特 平成 05-259871 (H05.10.18)] [公開 特 平成 07-114462 (H07.05.02)]

9. ソフトウェア開発支援装置及びソフトウェア開発支援方法[出願 特 平成 07-0165 (H07.06.28)] [公開 特 平成 07-283114 (H07.10.31)]

10. オブジェクト管理方法とその装置[出願 特 平成 07-179820 (H07.07.17)] [公開 特 平成 09-034708 (H09.02.07)]

11. リアルタイム制御装置及び方法[出願 特 平成 07-0166 (H07.08.23)] [公開 特 平成 08-032038 (H08.02.20)]

12. 複数機種対応プログラム自動生成装置[出願 特 平成 07-0158 (H07.08.23)] [公開 特 平成 08-012203 (H08.01.26)]

13. データ提示制御装置及びデータ提示制御情報編集装置[出願 特 平成 08-0384 (H08.09.03)] [公開 特 平成 08-240297 (H08.09.11)]