マルチモーダルUIにおけるモダリティ制御統一のためのモデル化手法

神尾広幸[†] 雨宮美香[†]
松浦博[†] 新田恒雄[†,☆]

マルチモーダルシステムは，複数のモダリティを扱うことから対話制御が複雑になり，システムの変更作業が困難になりがちである．本論文では，すべてのモダリティ制御を統一的に扱うことができるモデル化手法を提案する．このモデル化手法は，個々のモダリティを抽象化するUI-objectクラス，モダリティに対する制御を統一するメッセージ処理機構，モダリティ固有の要素技術を隠蔽するControllerオブジェクトなどを導入することで，すべてのモダリティを共通の枠組みの下にモデル化する．このことにより，システムの設計・開発が簡便化され，改良作業や新たなモダリティの追加にともなう変更作業への影響を最小限にとどめることができる．本論文では同時に，提案モデル化手法の効果を確認するため，GUI操作でMUIを設計し検証するラピッドプロトタイピングツールMuseをこの手法を適用して開発し，提案手法を用いずに開発した別のツールとの比較評価を行う．評価では，両方のツールの開発者に対するヒアリングと，新たなモダリティの追加という仕様変更に対する作業量の比較により，提案手法の有効性を実証する．

Object Modeling Technique for the Management of Modalities in Multimodal User Interfaces

HIROYUKI KAMIO,[†] MIKA AMAMIYA,[†] HIROSHI MATSUURA[†] and TSUNEO NITTA[†,☆]

This paper describes an object modeling technique to manage multiple modalities in multimodal user interfaces (MUI). This technique realizes a common framework for modeling various types of modalities by introducing an abstract class named UI-object, a message-processing mechanism that standardizes the control of modalities, and a controller object that conceals modality-specific technologies. By applying the proposed modeling technique to a system, a system engineer can improve and/or add new types of modalities easily. This paper also describes an application of the proposed technique to a rapid-prototyping tool Muse that enables one to design MUIs with convenient GUI environment. The evaluation is carried out with interviews for system engineers of each tool and a task in which a system engineer adds a new modality to his/her system. The work amount in the task on Muse is compared with the work amount on the conventional tool that we used to develop.

1. はじめに

我々は日々の生活の中で数多くの社会情報システムを利用している．たとえば，駅では自動券売機で切符を購入し，自動改札機を通って入場する．また銀行ではATM（Automated Teller Machine）で現金の引出しや振込みを行う．このように不特定多数のユーザが利用する社会情報システムでは，システムの機能・性能はもちろんのこと，優れたUI（User Interface）を備えているか否かが，使いやすさを決定する重要なポイントになる．このため，一般的には対象ユーザによるユーザビリティテスト[1]，あるいは操作結果のプロトコル解析[2]を通してUIの改善が図られている．

一方，従来のGUI（Graphical User Interface）に音声認識，音声合成などを採り入れたマルチモーダルUI（MUI: Multimodal User Interface）の研究がさかんに行われており[3)~5)]，これまでに数多くの試作システムが開発されている[6)~9)]．筆者らもマルチモーダル対話システムMultiksDial[10]を開発し，駅周辺の地理案内を目的とする情報案内システムに適用した．こ

† 株式会社東芝マルチメディア技術研究所
Multimedia Engineering Laboratory, Toshiba Corporation
☆ 現在，豊橋技術科学大学知識情報工学系
Presently with Department of Knowledge-based Information Engineering, Toyohashi University of Technology

のシステムの評価では，タスク達成時間を中心に比較実験を行い，MUIが社会情報システムのユーザビリティ向上に有効なことを示した．他方，MUIでは各モダリティの制御が従来のGUIに比べ繁雑なため，評価や改良が難しいことも，この実験から明らかになった．

本論文では，制御が複雑になりがちなマルチモーダルシステムに対して，MVC（Model/View/Controller）モデルに基づいたオブジェクト指向分析を行うと同時に，個々のモダリティを抽象化し，統一的に制御するモデル化手法を提案する．このモデル化手法は，モダリティ固有の制御をシステムから隠蔽するため，開発者はMUIの評価・改良を容易に行うことが可能になる．また，今回提案するモデル化手法の適用事例として，MUIを容易に作成でき，その評価・改良を通してユーザビリティを事前に検証できるラピッドプロトタイピングツールMuse（multimodal user interface design support editor）[11]を開発した．MuseはGUI環境でMUIを作成し，実行モードに切り替えることによりMUI動作を検証する．このため，システム仕様を作成するデザイナや営業担当者なども，MUIを容易に設計・評価し，改良することができる．

本論文では，MUIを持つシステムを開発する際の課題について述べた後（2章），MVCモデルに基づくモダリティのモデル化によってこれらの課題を解決する方法を詳説する（3章）．次に，提案のモデル化手法をラピッドプロトタイピングツールMuseの開発に適用した事例を述べるとともに（4章），この手法を確立する以前に試作したマルチモーダルUI開発支援ツールとの比較評価を行う（5章）．

2. MUIとその開発における課題

マルチモーダルはモダリティとモードという2つの概念を継承している．モダリティ（modality；知覚的様相）はもともと心理学で使われた用語で，情報が授受されるチャネル（たとえば音声，ジェスチャなど）の種類を指し，また情報が表現・知覚される様相までを含んでいる．たとえば，音声チャネルは音声自体がモダリティであり，また言語，感情，個人性といった異なるモダリティを内包している．一方，コンピュータUIの分野ではモード（mode）という用語が使われてきた．キーボード入力を例にとると，Shiftキーや Controlキーなどのモード指定キーによって，1打鍵に複数の意味を付与している．つまりモードは情報を解釈して意味を抽出したり，意味を担わせるためのチャネルの組合せを指して使用される．このことから，MUIとは複数のモダリティやモードを利用し対話するUIを指している[4]．

MUIでは，モダリティの組合せやモードの適切な利用によって，ユーザ側の意図をコンピュータ側にうまく伝えたり，コンピュータ側の状態・意図をユーザに分りやすく示すことが期待されている．たとえば，MITメディアラボのPut-That-There[6]では，磁気センサを用いた手によるポインティング入力と音声認識を併用し，「あれを，そこに，置け」といった操作を行う．このシステムは，対象や位置の指定に適したポインティング入力と，多数のコマンドから1つを選択するのに適した音声入力を組み合わせることで，個々のモダリティの特長を活かした新しいUIを実現した．また筆者らの開発したMultiksDialでは，音声入力に受話器を使用し，ユーザに音声入力を促す際にも画面表示と送話器を併用して操作ガイダンスを行う．操作ガイダンスは，ユーザが受話器を耳に当てているか否かのモードを利用し，適切なタイミングと内容で出力される．これは電話対話の比喩を利用したものであるが，音声入力が一般的になっていない現状では，ユーザに発話方法・タイミングを暗黙的に示唆する効果があり，初心者も容易に音声入力を利用できることが確認された．

一方，MUIを持つシステムを開発する側にとっては，複数のモダリティを組み合わせることにともなう以下の課題が残されている．

(1) モダリティ毎に要素技術の習熟が必要

MUIを構築するためには，個々のモダリティに関する技術に習熟する必要がある．たとえば，音声認識に関する技術には，認識方式や認識エンジンの実装方法などの要素技術と，認識語いの設定やAPI（Application Program Interface）の使用方法といった利用技術がある．システム開発者にとって利用技術の習得は避けられないにしても，モダリティの部品化を行うことにより，要素技術をシステム開発者から隠蔽し，各種モダリティを簡単に利用できるようにする必要がある．

(2) モダリティ追加作業に大規模な改造が必要

既存のアプリケーションに新たなモダリティを追加することは，かなり大変な作業である．特に複数のモダリティからの情報が複雑に絡み合ったMUIの場合には，システムの広範な部分にその影響が及ぶことがある．このため，異なるモダリティを統一的に扱うことのできる枠組みを考案し，モダリティの追加といった将来の拡張に対して柔軟に対応できるようにする必要がある．

(3) UIの評価・改良に多大の時間が必要

一般に，UIはユーザビリティテストなどの評価に

基づいて改良を繰り返すことで，より良いものに近付いていく．しかし，MUI は GUI に比べて対話制御が複雑なため，ユーザビリティテストによって UI 上の問題点が発見されても，改良を施すことは容易でない．複数のモダリティを扱うシステムの場合，個々のモダリティに対する制御方法を標準化することで，MUI の改良作業を定式化し簡便にする必要がある．

3. MUI におけるモダリティのモデル化

2 章にまとめた課題を解決するため，MUI に含まれるモダリティをオブジェクト指向技術を用いてモデル化することを試みる．モダリティのモデル化には，GUI システムのモデル化に数多くの適用実績がある MVC モデルを用いた．以下に MVC モデルの概要と，これをモダリティのモデル化に適用する方法を述べる．

なお，2 章では MUI の持つ 2 つの側面，モダリティとモードに言及したが，このうちモードはアプリケーションで規定される性格のものである．そこで，以下ではモダリティに限定して議論を進める．

3.1 MVC モデル

MVC モデルは，Model/View/Controller で表される 3 種類のオブジェクトとその相互関係を示したもので，Smalltalk-80 において利用された枠組みである[14),15)]．この 3 種類のオブジェクトと相互の関係を，図 1 を用いて説明する．図は 1 つの Model オブジェクト，3 つの View オブジェクト，および 3 つの Controller オブジェクトを示している．Model オブジェクトはあるデータ値を有し，3 つの View オブジェクトはこのデータを様々な形式で画面表示する．また各 View オブジェクトは，ユーザ入力方式を定義する Controller オブジェクトと関連を持つ．Controller オブジェクトはユーザからの入力を受け付けると，View オブジェクトを通して Model オブジェクトに値の変更を通知する．他方，Model オブジェクトは所有しているデータに変更が生じると，依存関係にある View オブジェクトすべてにデータの変更を通知する．この際，各 View オブジェクトは，自分の表示形式に合わせてデータを変更し表示する．

MVC モデルの特徴は次の 2 点にあると考えられる．

(1) Model オブジェクトに割り付けた複数の View オブジェクトを矛盾なく動作させることができる（Model と View の依存関係）．

(2) ユーザ入力方式を Controller オブジェクトとして抽出することで，入力方式を容易に変更できる（View と Controller の独立関係）．

これらの特徴により，MVC モデルを利用した GUI アプリケーションは，画面仕様や入力方式の変更に強い拡張性のあるものとなる．

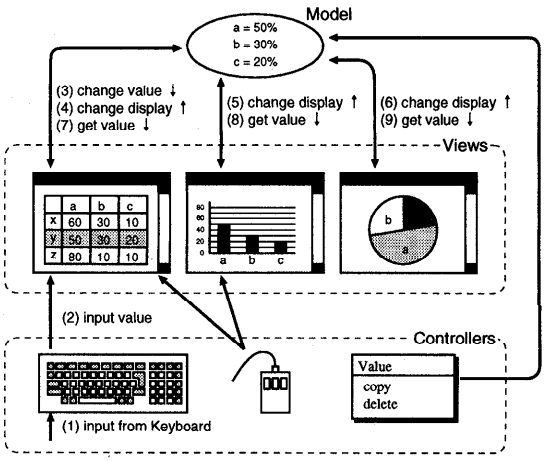


図 1 MVC モデル
Fig. 1 MVC model.

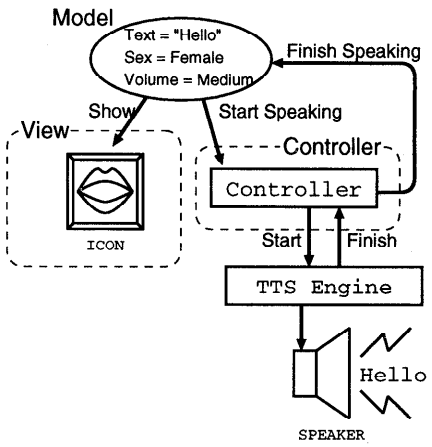


図 2 音声合成オブジェクトの MVC モデル化
Fig. 2 MVC model representation of text-to-speech object.

3.2 MVC に基づくモダリティのモデル化

今回対象とした MUI システムは，視覚（画面表示），聴覚（音声入出力），触覚（ポインティング）の 3 つのモダリティを利用している．これらのモダリティのモデル化に際して，モダリティの情報，モダリティの可視化，モダリティの制御という 3 つの要件に注目し，これを Model/View/Controller として MVC モデルに当てはめた．音声合成を例にとると，Model/View/Controller の関係は図 2 のようになる．モダリティの情報とは発話内容，音質，音量などの情報を指し，モダリティの可視化とは音声という目に見えないモダリティを GUI 部品（アイコン）として実

class UI-object		
int	m_X, m_Y;	// UI-objectの位置
int	m_W, m_H;	// UI-objectのサイズ
View*	m_pView;	// View
Controller*	m_pController;	// Controller
char	m_Name[256];	// 名前
int	m_Type;	// 種類（ボタン，音声認識など）
virtual void	Message(int);	// メッセージ処理用関数
virtual void	Show();	// 表示
virtual void	Hide();	// 非表示
void	Move(int, int)	// 移動
void	Resize(int, int)	// サイズ変更

図 3 UI-object の構造
Fig. 3 Structure of UI-object.

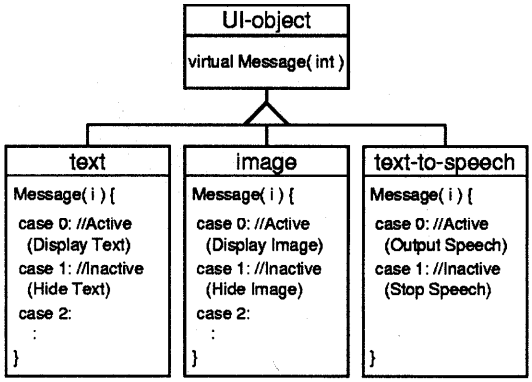


図 4 モダリティオブジェクトにおけるメッセージ処理機構の例
Fig. 4 Message processing mechanism in modality objects derived from UI-object.

現することを指している．またモダリティの制御とは，発声のタイミングといった実行動作や，発話内容の編集作業といったモダリティに対する操作を意味する．

また以下では，2 章に述べた課題を解決するため，MVC モデルで表現された異なるモダリティを統一的に扱うことのできる枠組みを提案する．

(1) Model の抽象化

アプリケーションから個々のモダリティを制御する方法を統一するため，抽象クラス “UI-object” を導入した．UI-object はモダリティの情報を保持する Model を抽象化したものであり，メンバとして View，Controller を所有している．個々のモダリティに対応するオブジェクト（以下，モダリティオブジェクトと呼ぶ）は UI-object を継承して作成する．**図 3** に UI-object が持つメンバ変数，メソッドの一例を示す．UI-object は位置やサイズ，名前など，モダリティオブジェクトが共通に持つ属性をメンバ変数として保持している．一方，UI-object クラスが持つメソッドは，表示，非表示や移動などの関数が定義されている．これらの中で，モダリティの種類によって異なる実装を必要とする関数については，仮想関数として定義し，継承先のモダリティオブジェクトにおいて実装する．これにより，すべてのモダリティを UI-object という同一の手段で管理することが可能になり，アプリケーションと各モダリティの関係が明確になる．

なお，後述するように，MUI を対象としたラピッドプロトタイピングツールの開発においては，音のように目に見えないモダリティに対しても，画面に表示されるオブジェクト同様，画面上で選択・削除を行うことが必要とされる．この場合には，図 2 に示した音声合成オブジェクトのように，モダリティを可視化するアイコンを用意して，View に割り当てた．

一方，目に見えないモダリティを表示する必要がない一般の MUI システムでは，そのモダリティに対応する View を作成しないことで対処する．オブジェクト指向分析方法によっては，音声入出力を画面表示と同様に View として取り扱うことも考えられるが，提案手法においては View はあくまで画面表示のみを扱うものとして定義した．この理由は，X-Window などのウインドウシステムにおいて MUI システムを実装する場合，View を Widget[12]（小さなウインドウオブジェクト）として実装できるように配慮したためである．本来，オブジェクト指向分析においては，実装時の制約を排除して分析を進めるのが理想であるが，分析から設計・実装へと作業を進める際の後戻り工程を少なくするために，上記の制約を設けた．

UI-object を導入したことによって，将来においてモダリティの追加作業が発生した場合にも，システム開発者は新たなモダリティオブジェクトを UI-object から継承して作成することにより，容易に追加することが可能になる．

(2) モダリティ制御方法の標準化

モダリティ制御方法を標準化するため，UI-object の動作は独自のメッセージ処理機構により制御される仕組みになっている．たとえば，音声合成オブジェクトでは，音声合成エンジンが持つ「読み上げ開始（Output Speech）」や「読み上げ停止（Stop Speech）」といった動作を行う関数を外部に公開するのではなく，**図 4** に示す Message という仮想関数を公開している．この関数は引数でメッセージ番号が渡される仕様になっており，UI-object を継承したモダリティオブジェクトはこの番号に従って動作する．メッセージ番号は各オブジェクトごとに固有の意味を有する．たとえば，テキストやイメージでは 0 が表示（Display Text/Image），1 が非表示（Hide Text/Image）であり，音声合成では 0 が読み上げ開始，1 が読み上げ停止である．これ

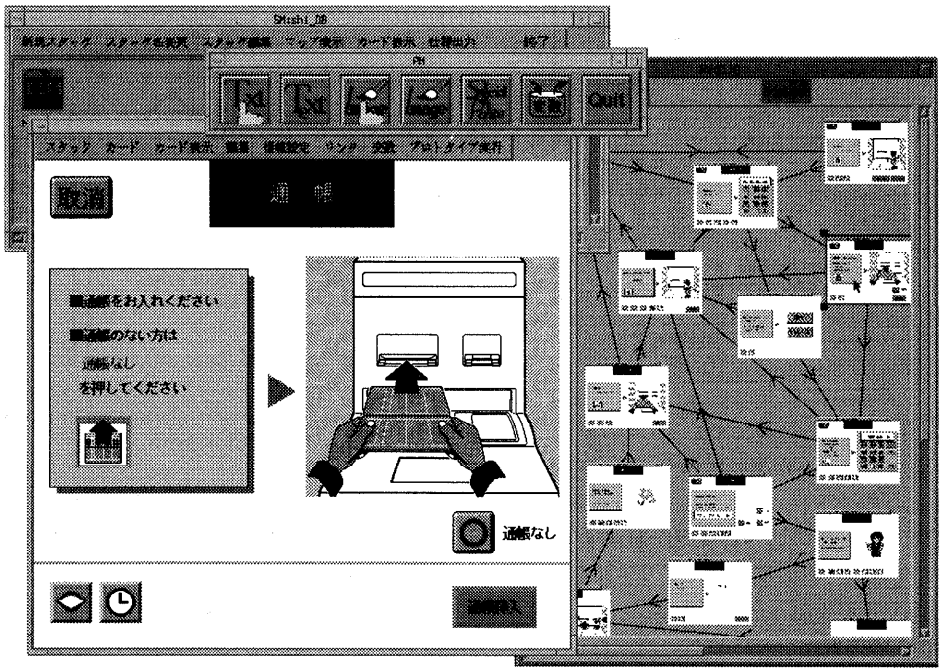


図5 ラピッドプロトタイピングツール Muse
Fig. 5 Muse, a rapid prototyping tool.

により，個々のモダリティを制御することが，モダリティの実装部が持つ独自の関数を呼び出すことの代わりに，モダリティオブジェクトに対してメッセージを送信するという共通の手続きで置き換えられる．この結果，すべてのモダリティの制御を標準化することができる．

このメッセージ処理機構により，MUI 動作の記述はメッセージ番号やメッセージの送信先，送信順序を決定するという定式化された作業で行われる．また，たとえば，テキスト表示と音声出力の順番を変更するといった MUI の改良作業も，メッセージの送信順序を変更することにより簡便に行うことが可能になる．

(3) Controller による要素技術の隠蔽

モダリティオブジェクトがメッセージを受信すると，オブジェクト内部でメッセージ番号に従った処理が行われる．たとえば，図2に示したように，音声合成オブジェクトが「読み上げ開始」メッセージを受け取ると，Controller は音声合成エンジンと通信して，対応する音声を出力させる．また音声出力が終了すると，Controller は音声合成エンジンが発生する終了イベントを Model（UI-object）に伝える．一方，音声認識オブジェクトの場合，Controller は音声認識エンジンの状態を監視し，認識結果が得られたことをきっかけとして，そのイベントを Model に伝える．すなわち，Controller は UI-object とモダリティの実装部（音声合成エンジンや音声認識エンジン）との間を結ぶ，UI-object のメッセージ処理機構とモダリティに固有の制御方法との仲立ちの役割を果たす．また，システム開発者が Controller を開発する際には，モダリティの利用技術に習熟する必要がある．しかし，要素技術開発者が利用技術を提供することにより，モダリティ実装部をシステム開発者から隠蔽することができる．したがって，Controller を介在させることによって，作成した UI-object を用いてシステム開発者が MUI システムを構築する際には，モダリティ利用技術より簡便なメッセージ処理機構に基づく方法で，動作制御を記述することが可能になる．

このように，Controller によって，システム開発者は各種モダリティを容易に利用できるようになる．

4. 提案モデル化手法の適用事例

ここでは提案手法の適用事例として，ラピッドプロトタイピングツール Muse の開発について述べる．

4.1 ラピッドプロトタイピングツール Muse

Muse は，MUI の設計－試作－改良を迅速に行うことを目的としたラピッドプロトタイピングツールである（図5）．

Muse は，画面や対話シナリオを設計する設計モー

表 1 Muse で用意されている部品の種類
Table 1 Modality objects offered to Muse.

画面を構成する オブジェクト	イメージ部品 テキスト部品
音声を出力する オブジェクト	サウンド部品 音声合成部品
マルチモーダル入力 を行うオブジェクト	イメージボタン テキストボタン 音声認識部品 指書き文字認識部品
制御を行う オブジェクト	タイマ部品 条件分岐部品など

ドと，MUI を実際に動作させてユーザビリティテストを行う実行モードを備えている．設計モードでは，プログラミングに不慣れな人も容易に MUI を作成できるように，画面設計と対話シナリオ設定のすべてを GUI で行えるようになっている．

設計モードにおける画面設計は，図 5 に示すように画面（カード）上に部品を配置することで行う．Muse には**表 1** に示す部品が用意されており，これらの部品をマウス操作で任意の位置に配置することで，カードの外観を設計する．カード上の編集処理を行うマネジャをカードレイアウトマネジャ（CLM: Card Layout Manager），部品供給を行うマネジャをパーツマネジャ（PM: Parts Manager）と呼ぶ．カードの集合はスタックに格納され，スタックマネジャ（SM: Stack Manager）がこれを管理する．また，スタック内の複数のカードの一覧と，それら相互の接続関係を表示して MUI の構造の視認性を高めるため，図 5 の右側に示すようなマップビューア（MV: Map Viewer）も用意されている．

設計モードでは，設計者はまず対話シナリオを設定する．これは前述の画面設計において，カード上に配置した部品から同じカード上の部品や他のカードにリンクを張ることによって行われる．このとき，リンク元の部品はリンク先とリンク先に送信するメッセージを保持する．

次に実行モードでは，設計した MUI を動作させて検証が行われる．実行モード中に，たとえば，マウスやタッチパネルを用いて部品をポインティングすると，その部品は設定されたリンク先にメッセージを送信する．メッセージを受けたカードや部品は，メッセージに従った動作を行う．たとえば，サウンド部品や音声合成部品は「再生（Active）」というメッセージを受けると音声を出力し，「停止（Inactive）」というメッセージを受けると出力を停止する．なお，Muse には設計モードと実行モードを切り替え，実行時の制御を行う実行マネジャ（EM: Execution Manager）が用意されている．

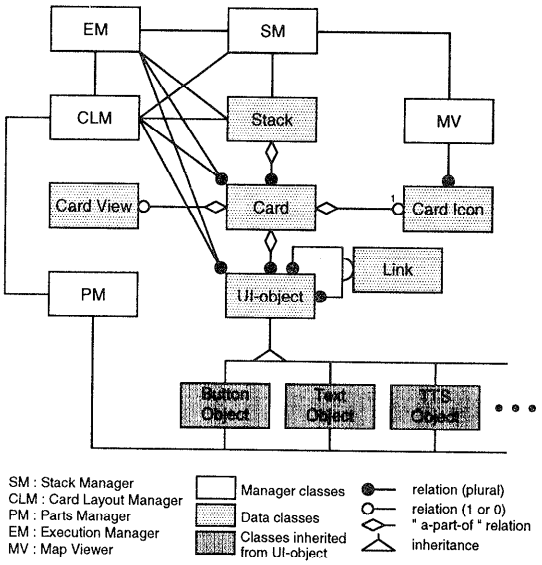


図 6 Muse におけるクラス関連図
Fig. 6 Class structure in Muse.

4.2 提案モデル化手法に基づく Muse の開発

Muse の開発では，オブジェクト指向分析/設計の分野で適用事例が豊富な OMT[13] 法を採用した．

Muse のオブジェクト指向分析では，まず内部構成要素であるオブジェクトの抽出作業を行った．その結果，4.1 節で述べた 5 つのマネジャ（スタックマネジャ，カードレイアウトマネジャ，パーツマネジャ，マップビューア，実行マネジャ）と，Muse で取り扱われるデータを表すクラスが抽出された．以下に抽出されたデータを表すクラスを列挙する．

- **Stack**： 複数の Card を管理するクラス
- **Card**： 画面を表すクラス
- **CardView**： Card の View クラス
- **UI-object**： Card 上に配置される部品のクラス
- **Link**： 部品間に設定されるリンク情報のクラス

これらのクラスのうち，UI-object は 3.2 節で述べたモダリティを抽象化したクラスである．Muse で取り扱われる様々なモダリティを表す部品（ボタン，テキスト，音声合成など）は，すべてこのクラスを継承して作成した．

次に，これらのクラス間の関連図を作成した（**図 6**）．クラス関連図はクラス間の参照・利用関係を表しており，関連を持つクラスは実線で結ばれている．図から，UI-object はマネジャや Card と関連を持つが，UI-object を継承したボタンなどのクラスは他のクラスと

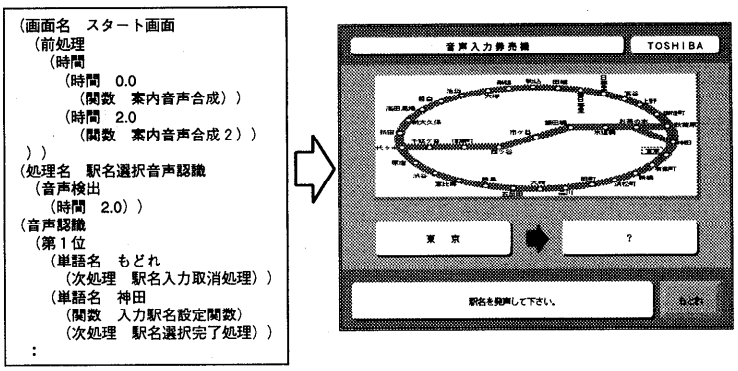


図7 UISCRIPT と作成された画面
Fig. 7 UISCRIPT and a generated picture.

関連を持たないことが分かる．これは提案のモデル化手法が，すべてのモダリティ制御を UI-object の枠組みに基づいて統一したことの効果といえる．

抽出したクラス構造を基に，各クラスを設計し Muse の実装を行った．プラットフォームは，東芝 AS4080（SUN 社製 SPARC Station10 同等）ワークステーション上で動作する OSF/Motif1.2 ウインドウシステムを採用した．実装言語は SUN C++2.0，実装規模は約 2 万ステップであった．

5. 提案モデル化手法の適用効果

ここでは，以前筆者らが提案のモデル化手法を用いずに開発したマルチモーダル UI 開発支援ツール[16]と Muse を比較することにより，本手法の効果を確認する．

5.1 マルチモーダル UI 開発支援ツール

マルチモーダル UI 開発支援ツールは，MUI の仕様を日本語のスクリプト言語で記述することにより，MultiksDial 上でプロトタイピングを行う開発環境である．この開発環境では，システム開発者は UI 仕様記述言語（UISCRIPT）を用いて MUI の仕様を記述する．続いて，これを解析プログラムでコンパイルすることにより，バイナリ形式の中間言語ファイルを作成する．このファイルを MultiksDial 上の実行管理部に読み込ませることにより，MUI の検証を行う．**図 7** に UISCRIPT と，実行管理部で動作中の MUI 画面の例を示す．

このツールは大きく分けると，UISCRIPT を解析して中間言語に翻訳するコンパイラ（Compiler）部と，中間言語を読み取り MUI の実行を行う実行管理部（Execution Manager）で構成される．実行管理部は，画面表示部（Display Manager），時間管理部（Timer Manager），音声認識処理部（Speech Recognition Manager），音声合成処理部（TTS Manager），およびタッチパネル処理部（Touch Screen Manager）と通信し，MUI の動作検証を行う．**図 8** に実行管理部のオブジェクト関連図を示した．

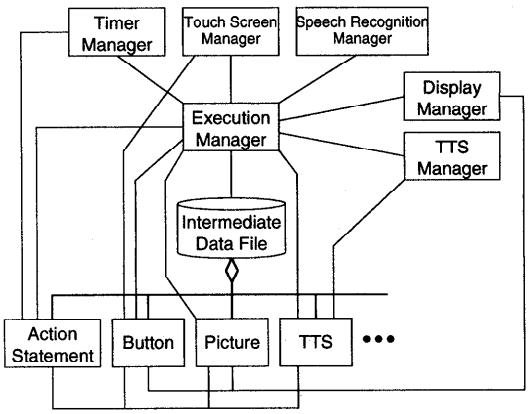


図8 マルチモーダル UI 開発支援ツールのオブジェクト関連図
Fig. 8 Class structure in multimodal UI development tool.

実行管理部によるプロトタイピング機能を用いることで，システム開発者がユーザの操作状況を観察し，操作時におけるユーザのとまどいや誤操作を発見するユーザビリティテストを実施できる．このユーザビリティテストを通して MUI の問題点を発見した場合，システム開発者は UISCRIPT を修正することで改良を図る．図 7 に示したように UISCRIPT は日本語で記述されているため，仕様の閲覧性が高く，容易に改良作業を進めることができる．

5.2 開発工数による比較の結果と考察

5.1 節に説明したマルチモーダル UI 開発支援ツールと Muse の開発工数の比較を**図 9** に示す．図を見ると，Muse の分析（Analysis）における工数が多いことが分かる．これは Muse の分析作業を，OMT 法の実習を兼ねて進めたためである．一方，Muse の設計

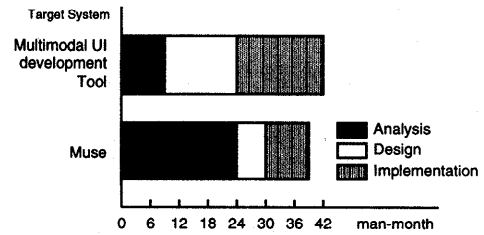


図 9 Muse とマルチモーダル UI 開発支援ツールの開発工数の比較
Fig. 9 Comparison of the man-month between Muse and Multimodal UI development tool.

(Design) 以降においては，マルチモーダル UI 開発支援ツールと比較して少ない工数で作業を終えることができた．これは，オブジェクト指向分析によって Muse のクラス構造を分析段階において確立することができ，その構造に基づいて設計，実装 (Implementation) を進めることができた効果である．

次に，Muse の開発における提案モデル化手法の効果を調査した．以下は，双方のツールのシステム開発者へのヒアリング結果を基にまとめたものである．

- Muse で利用するモダリティを UI-object として抽象化したため，分析時においてはモダリティの種類を意識せず，UI-object のみを対象として分析作業を進めることができた．
- 分析時に UI-object のみを対象としたため，アプリケーション本体と個々のモダリティ間の独立性が高く，実装時の分業が容易であった．
- すべてのモダリティを UI-object の継承クラスとして作成したため，モダリティの追加は定形的な作業によって行うことができ，工数が短縮できた．

5.3 モダリティの追加による工数比較の結果と考察

5.2 節で述べた，モダリティの追加作業における工数の短縮効果を定量的に評価するため，マルチモーダル UI 開発支援ツールに音声認識機能を追加した際の作業量と，Muse に同様の機能を追加した際の作業量とを比較した．

この作業にともない変更が加えられたクラスの一覧を**表 2** (マルチモーダル UI 開発支援ツール)，および**表 3** (Muse) に示す．

マルチモーダル UI 開発支援ツールでは，音声認識機能追加は UISCRIPT 内に音声認識処理の記述を追加することで行われる．この UISCRIPT の文法追加にともない，UISCRIPT のコンパイラと実行管理部に新文法処理のための修正を行った．また実行管理部では，音声認識処理を行う制御部分も修正する必要があった．さらに，音声認識処理部の新設と，認識処理との同期をとる時間管理部の修正も行った．図 8 のオブジェクト関連図を見ると，実行管理部や時間管理部など複数のクラスが，ボタンや TTS などのコンテンツと直接関連を持っている．このため，新たなコンテンツが追加される場合，関連を持つすべてのクラスに影響が及ぶ．

表 2 変更が発生したクラスの一覧 (マルチモーダル UI 開発支援ツール)
Table 2 Modified classes when adding speech recognition facility (Multimodal UI development tool).

クラス名	修正内容
実行管理部	中間言語ファイル読取部修正
タイマ処理部	音声認識処理用タイマ処理追加
音声認識処理部	新規作成
コンパイラ	中間言語ファイル作成部修正
UISCRIPT	文法修正

表 3 変更が発生したクラスの一覧 (Muse)
Table 3 Modified classes when adding speech recognition facility (Muse).

クラス名	修正内容
音声認識部品	新規作成
パーツマネジャ	音声認識部品作成処理追加

一方，提案モデル化手法を適用して開発した Muse でも，音声認識機能は音声認識部品というクラスとして新たに追加された．しかし，その他のクラスでは，パーツマネジャ以外に修正の必要がなかった．図 6 のオブジェクト関連図に示したように，パーツマネジャ以外のクラスは UI-object を通して部品と通信しており，各部品と直接の関連を持たない．したがって，新たなモダリティを追加しても，個々の部品供給を行うパーツマネジャ以外に追加作業が影響することはなかった．

なお，今回の追加作業では現れなかったが，新たなモダリティの追加にともない UI-object の仕様が変更になる場合も考えられる．たとえば，現在の仕様では Message 関数にメッセージ番号を引数として送ることになっているが，引数を増やしたいという要求が生じることもありうる．この場合は，UI-object に必要な関数を追加することで対応することになる．これにより，UI-object を継承したすべてのモダリティオブジェクトにこの追加が反映される．すなわち，個々のモダリティ独自の仕様を排除して UI-object による抽象関係を維持することによって，モダリティ制御の標準化を保つことができる．

両ツールに音声認識機能を追加した際の作業量を比較すると，マルチモーダル UI 開発支援ツールでは全 173 ファイル中修正ファイル数 40 ファイル，修正ステップ数 2304 ステップ，追加にともなう仕様分析か

ら設計，実装の完了に至るまでの作業工数は3人月であった．一方，Museでは全272ファイル中修正ファイル数8ファイル，修正ステップ数945ステップ，実装完了までの作業工数は1人月であった．このことから，モダリティの追加作業における提案手法の有効性が確認された．

6. おわりに

本論文では，MUIシステムに対し，MVCモデルに基づくオブジェクト指向分析を行うことで，異なるモダリティを統一的に扱うことのできるモデル化手法を提案した．

このモデル化手法は，UI-objectによる個々のモダリティの抽象化，メッセージ処理機構によるモダリティ制御方法の標準化，Controller導入による要素技術の隠蔽を実現する枠組みを持ち，すべてのモダリティをこの手法でモデル化することにより，MUI開発における課題の解決を図ることができた．

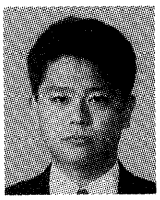
また提案手法の適用事例として，ラピッドプロトタイピングツールMuseを開発し，以前に試作したマルチモーダルUI開発支援ツールとの比較評価を行った．評価は，新たなモダリティの追加という仕様変更に対する作業量を比較することによって行った．Museとマルチモーダル UI開発支援ツールは同じMUIラピッドプロトタイピングを目的としているが，前者はGUI操作によるMUI設計の容易性を重視し，後者はUISCRIPT表記によるMUI仕様の閲覧性を重視している．このようにツールの性格が異なることから，両者を単純に比較することはできないが，本論文では仕様変更に対する柔軟性に注目し，提案モデル化手法の効果を確認することができたと考えている．

今回提案したモデル化手法は，MUIシステム全般のオブジェクト指向分析に適用できると考えている．しかし本論文においては，ラピッドプロトタイピングツールの開発への適用についてのみ言及し，一般的なMUIシステムへの適用を証明したものではない．今後はこの手法を様々なMUIシステム開発に適用することにより，提案手法の妥当性を評価していきたい．

参考文献

1) 日本認知科学会（編）：認知科学の発展 第5巻，講談社 (1992).
2) 海保博之，原田悦子：プロトコル分析入門，新曜社 (1993).
3) Nigay, L. and Coutaz, J.: A design space for multimodal systems: Concurrent processing and data fusion, *INTERCHI'93*, pp.172–178 (1993).
4) 新田恒雄：GUIからマルチモーダルUI（MUI）に向けて，情報処理学会論文誌，Vol.136, No.11, pp.1039–1046 (1995).
5) 速水 悟，竹澤寿幸：マルチモーダル情報統合システムの研究動向，人工知能誌，Vol.13, No.2, pp.206–211 (1998).
6) Bolt., R.A.: Put-That-There: Voice and gesture at the graphic interface, *Computer Graphics*, Vol.14, No.3, pp.262–270 (1980).
7) 竹林洋一：音声自由対話システムTOSBURG II—ユーザ中心のマルチモーダルインタフェースの実現に向けて，信学論，Vol.J77-D-II, No.8, pp.1417–4128 (1994).
8) 長尾 確：マルチモーダルインタフェースとエージェント，人工知能誌，Vol.11, No.1, pp.32–40 (1996).
9) 西本卓也，志田修利，小林哲則，白井克彦：マルチモーダル入力環境下における音声の協調的利用—音声作図システムS-tgifの設計と評価，信学論，Vol.J79-D-II, No.12, pp.2176–2183 (1996).
10) 神尾広幸，松浦 博，正井康之，新田恒雄：マルチモーダル対話システムMultiksDial，信学論，Vol.J77-D-II, No.8, pp.1429–1437 (1994).
11) 神尾広幸，雨宮美香，内山ありさ，松浦 博，新田恒雄：社会情報システムのためのラピッドプロトタイピングツールMuseの開発，信学技報，NLC95-50, SP95-85 (1995).
12) Young, D.A.: *The X Window System Programming and Applications with Xt OSF/Motif Edition*, pp.17–54 (1990). 川手恭輔（訳）：X Toolkitプログラミング OSF/MOTIF版，トッパン (1991).
13) Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W.: *Object-Oriented modeling and design*, Prentice-Hall, Englewood Cliffs, NJ (1991). 羽生田栄一（訳）：オブジェクト指向方法論OMT，トッパン (1992).
14) Krasner, G.E. and Pope, S.T.: A cookbook for using the Model-View-Controller user interface paradigm in Smalltalk-80, *Journal of Object-Oriented Programming*, Vol.1, No.3, pp.26–49 (1988).
15) Gamma, E., Helm, R., Johnson, R. and Vlissides, J.: *Design patterns*, Addison-Wesley (1995). 本位田ほか（訳）：デザインパターン，ソフトバンク社 (1995).
16) 松浦 博，神尾広幸，郡田美香，新田恒雄：マルチモーダル対話システムのためのラピッドプロトタイピング，音学講論，1-7-24, pp.47–48 (1994).

(平成10年10月20日受付)
(平成11年2月8日採録)



神尾 広幸（正会員）

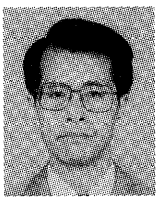
昭和 43 年生．平成 4 年上智大学大学院理工学研究科電気電子工学専攻修士課程修了．同年（株）東芝入社．現在同社マルチメディア技術研究所，開発第 6 部所属．ヒューマンインタフェース，マルチモーダルインタフェースの研究開発に従事．電子情報通信学会，日本音響学会各会員．

雨宮 美香（正会員）

昭和 44 年生．平成 5 年東京女子大学文理学部数理学科卒業．同年（株）東芝入社．現在同社マルチメディア技術研究所，開発第 6 部所属．ヒューマンインタフェース，マルチモーダルインタフェースの研究開発に従事．

松浦 博

昭和 30 年生．昭和 56 年早稲田大学大学院理工学研究科電気工学専攻修士課程修了．同年（株）東芝入社．現在同社マルチメディア技術研究所，開発第 6 部グループ長．音声認識装置，ヒューマンインタフェースの研究に従事．工学博士．電子情報通信学会，日本音響学会各会員．



新田 恒雄（正会員）

昭和 21 年生．昭和 44 年東北大学工学部電気工学科卒業．（株）東芝マルチメディア技術研究所主幹を経て，現在豊橋技術科学大学教授（大学院工学研究科知識情報工学専攻）．ディジタル信号計測，音声情報処理，およびヒューマンインタフェースの研究に従事．工学博士．平成元年電子情報通信学会論文賞受賞．著書「マルチメディアとディジタル信号処理」（共著，コロナ社）ほか．電子情報通信学会，日本音響学会，IEEE 各会員．