音声言語情報処理 12－15
（1996. 7. 27）

オブジェクト指向モデリングおよび設計を用いたラピッドプロトタイピングツールMuseの開発

神尾 広幸，雨宮 美香，松浦 博，新田 恒雄

株式会社 東芝 マルチメディア技術研究所

音声入出力や手書き文字入力などのマルチモーダル入出力を備える社会情報システムの開発では，少なからぬ評価・改良サイクルが必要となる．本稿ではラピッドプロトタイピングツールMuseのオブジェクト指向開発について述べる．Museの開発は，分析の段階から一貫してMVC-Xモデル(MVCモデルをマルチモーダルシステムに特化した分野特化フレームワーク)を基盤に進められた．MVC-Xモデルの適用によって，各種モダリティの統一的な扱いと将来の拡張容易性を保証するとともに，複雑な対話制御を仕様化することができた．

Object-Oriented Modeling and Designing of Rapid Prototyping Tool — Muse

Hiroyuki Kamio　Mika Amamiya　Hiroshi Matsu'ura　Tsuneo Nitta

Multimedia Engineering Laboratory, Toshiba Corporation

In this paper, we describe the Object-Oriented Modeling and Designing of Rapid Prototyping Tool "Muse" for social information systems equiped multimodal input and/or output facilities including voice recognition, finger-writing character recognition, and text-to-speech. The development of Muse is done based on MVC-X model, a new application-specific framework, which was given by extending MVC model to adapt multimodal UI environment. By using MVC-X model, we can integrate various types of modalities, adapt to the future extension of UI environment, and get an unification mechanism to contorol complex, multimodal dialogue.

1 はじめに

銀行の自動取引装置 (ATM) や駅の自動券売機など，我々の生活の中で数多くの社会情報システムが利用されている．これらのシステムは不特定多数のユーザが利用することから，わかりやすいユーザインタフェース (UI) を持つ必要がある．筆者らは入出力チャネルを多重化したマルチモーダル対話システム MultiksDial[1] を開発し，東京駅周辺の地理案内をタスクとした情報案内システムに適用した．そして，タスク達成時間の比較実験により，マルチモーダル UI(以下 MUI と呼ぶ) が社会情報システムのユーザビリティ向上に有効であることを確認した．一方で，複数のモダリティを有する MUI の場合，従来の GUI に比べて各モダリティの制御が複雑になるため，その評価や改良が難しいことも明らかになった．特に異なるモダリティを同時に取り扱うシステムでは，各モダリティが互いに影響を及ぼし合うため，その評価・改良は非常に困難である [2].

今回，筆者らは社会情報システムで利用する MUI を容易に作成でき，また評価・改良を通してユーザビリティを事前に評価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)[3] を開発した．本ツールを使用すると，MUI の作成，実行をすべて GUI 環境で行うことができるため，システムの仕様を作成するデザイナや営業担当者など，プログラミングに不慣れな人でも容易に UI を設計，評価し，そして改良することができる．

筆者らは Muse を開発する上で，オブジェクト指向技術を採用した．具体的には，これまでの GUI システムの開発に数多くの適用実績がある MVC(Model / View / Controller) モデルを MUI システム向けに拡張した MVC-X (eXtended-MVC) モデルを提案し，このモデルに準拠して OMT 法 [4] を適用し開発した．

2 ラピッドプロトタイピングツール Muse

Muse は，画面や対話シナリオを設計する設計モードと，MUI を実際に動作させ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でも容易に 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 を利用して行えるようになっている．画面の設計は，図 1に示すように画面 (カード) 上に部品 (UI-object) を配置することで行う．Muse には表 1に示す UI-object が用意されている．これらの UI-object 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していく．このカードの編集を行うマネジャをカードレイアウトマネジャ(CLM:Card Layout Manager)，UI-object を供給するマネジャをパーツマネジャ(PM:Parts Manager) とよぶ．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，MUI の構造の視認性を高めるため，マップビューア (MV:MapViewer) が用意されている．

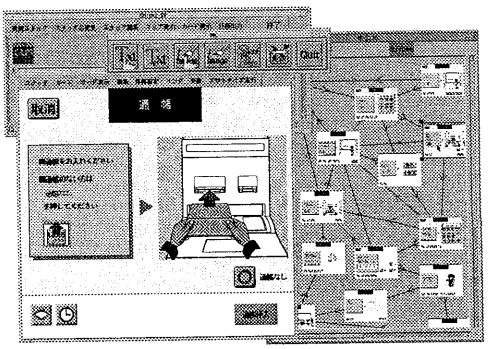

図 1: Muse の全体図

一方，対話シナリオはカード上の UI-object から他のカードや UI-object にリンクを張ることによって設定する．リンク元となる UI-object 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．

Muse のメニューから「実行」を選択すると実行モードに入り，設計した MUI を動作させることができる．実行モード中にマウスやタッチパネルを用いて UI-object をポインティングすると，その UI-object 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 UI-object

表 1: Muse で用意されている UI-object の種類

画面を構成する UI-object	イメージ部品 テキスト部品 アニメーション部品
音声を出力する UI-object	サウンド部品 音声合成部品
マルチモーダル入力を行う UI-object	イメージボタン テキストボタン 音声認識部品 文字認識部品
制御を行う UI-object	タイマ部品 条件分岐部品など

は，メッセージに従った動作を行う．例えばイメージ部品やテキスト部品は「表示 (Active)」というメッセージを受けるとカード上に表示され，「非表示 (Inactive)」というメッセージを受けるとカード上から消去される．なお，Muse において設計モードと実行モードを切替えるマネジャとして，実行マネジャ(EM:Execution Manager) が用意されている．

3 Muse のオブジェクト指向開発

本章では，MVC モデルを MUI システム向けに拡張した MVC-X モデルを提案し，これに準拠した Muse のオブジェクト指向開発について説明する．

3.1 MVC-X モデル

MVC モデルは，Model / View / Controller という 3 種類のオブジェクトとその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 [5][6]．この 3 種類のオブジェクトとその関係を，図 2を用いて説明する．図には 1 つの Model オブジェクトと 3 つの View オブジェクト，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，このデータを 3 つの View オブジェクトがさまざまな形式で画面表示する．また，各 View オブジェクトは，値を変化させるための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．また 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．各 View オブジェクトは，自分の表示形式に合わせて変更されたデータを表示する．

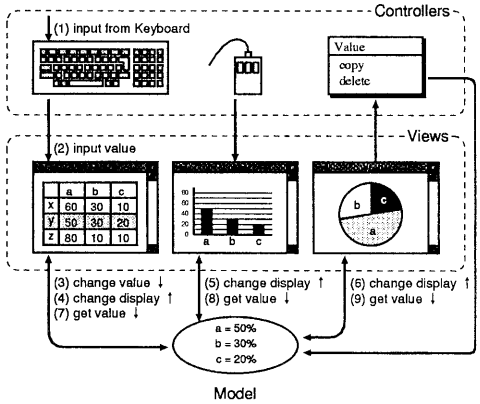

図 2: MVC モデル

この MVC モデルに見られる特徴は次の 2 点にある．

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．

Muse の開発においては，この MVC モデルを MUI 向けに拡張した MVC-X モデルを定義し，このモデルに基づくオブジェクト指向開発を行った．以下では，MVC-X モデルで Model / View / Controller がそれぞれどのような役割を担うかを説明する．

Muse では，視覚，聴覚，触覚の 3 つのモダリティを扱うことができ，これらを利用した MUI の作成・実行をすべて GUI 上で行うことができる．このようなモダリティを GUI 上に実現するには，モダリティの情報操作，モダリティの可視化，モダリティの制御という 3 つの要件が必要となる．MVC-X では，これら 3 つの要件を Model / View / Controller としてとらえる．音声というモダリティを例にとると，Model / View / Controller の関係は図3のようになる．モダリティの情報とは，音声として発せられる内容，音質，音量などの情報のことであり，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) として実現することである．またモダリティの制御とは，音声の発せられるタイミングといった実行動作や，発声内容の編集作業といったモダリティに対する操作を扱うことを意味する．図3の場合，モダリティの情報は Model クラスが保持し，モダリティの可視化は View クラスが担当していることがわかる．またモダリティの制御は，実行マネジャ(EM) やカードレイアウトマネジャ(CLM) といった Controller クラスからの依頼によって行われる．

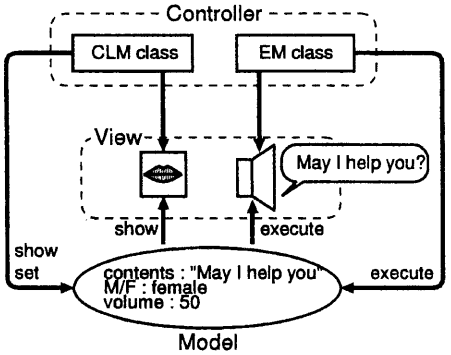

図 3: 音声合成部品における MVC-X モデル

3.2 MVC-X モデルに基づくマルチモーダル対話機能の実現

Muse では，すべてのモダリティを同一に扱うためのオブジェクトを定義し，MUI の機能拡張や評価・改良といった要求を容易に行えるようになっている．具体的には「UI-object」と呼ばれるすべてのモダリティのための抽象クラスを定義し，各モダリティは UI-object を継承した Model クラスを作成して実現した．これにより，音声や画像という異なるモダリティでも，Muse 内では GUI で全く同等に取り扱うことが可能となった．以下に UI-object によるマルチモーダル対話機能の具体的な実現方法について説明する．

3.2.1 出力のマルチモーダル化

出力を行う UI-object には画面を構成するもの (イメージ部品，テキスト部品など) と，音声を出力するもの (サウンド部品，音声合成部品) が用意されている．設計モードでは GUI ですべての UI-object を操作できるようにするため，音声などの目に見えないモダリティを可視化するアイコンを用意し，カード下部にアイコン表示領域を設けた．このアイコンは対応する UI-object が所有する View クラスである．これにより，配置やリンクの設定なども他の UI-object と同様にすべて GUI で行うことができる．

実行モードでは，音声合成部品は「出力 (Active)」または「停止 (Inactive)」というメッセージをリンク元から受け取り，それに従って音声を出力／停止する．図4に示すように抽象クラス UI-object のメソッド “Active”，“Inactive” は仮想関数であり，継承された各 UI-object 独自のメソッドが呼び出される．Muse では，このようにして出力のマルチモーダル化が実現されている．

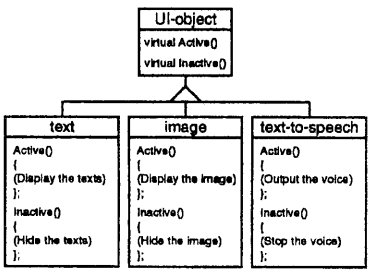

図 4: UI-object の継承関係

3.2.2 入力のマルチモーダル化

入力を行うUI-objectには，ポインティング動作を受け取るイメージ部品とテキスト部品，タッチパネルなどで画面上に描かれた文字を認識する文字認識部品，音声認識装置からの認識結果を受信する音声認識部品がある．イメージ部品やテキスト部品は，ユーザからマウスやタッチパネルでポインティングされることをきっかけとして，リンク先にメッセージを送信する．一方，文字認識部品や音声認識部品は，それぞれの認識結果が得られたことをきっかけとして，リンク先にメッセージを送信する．このようにUI-objectによってそれぞれ異なったイベントが入力されるが，各UI-object内でイベントを処理してからは，どのUI-objectもリンクに従ったメッセージ交換によって動作する．従って，リンクの設定方法はすべてのUI-objectで統一されている．このためユーザは，どのような入力モダリティでも特別な配慮を払うことなくGUIで取り扱うことができる．

また，音声認識などの認識結果を格納する手段として，Museでは変数オブジェクトが用意されている．図5は音声認識の処理を表したものである．音声認識部品は音声認識装置から認識結果を受け取ると，変数オブジェクトにその結果を格納して，条件分岐部品にメッセージを送信する．条件分岐部品は条件とリンク先を格納したテーブルを持ち，認識結果と比較して対応するリンク先にメッセージを送信する．これによって認識結果に応じた動作を記述することができる．

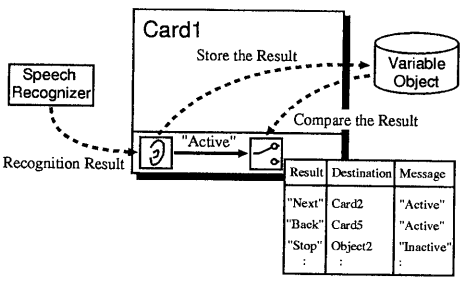

図 5: 音声認識の処理の流れ

3.3 MVC-X モデルによる効果

Museのマルチモーダル対話機能をMVC-Xモデルで構築することで，(1) 音声合成技術や文字認識技術等の要素技術をアプリケーション開発者から隠蔽し，(2) 将来新たな要素技術が開発された際にその機能を容易に追加し，(3) すべてのモダリティの制御を定式化することで統一的に取り扱うことができる，という効果が得られる．以下に各項目を詳述する．

3.3.1 要素技術の隠蔽

MVC-Xモデルの枠組みによって，すべてのUI-objectはModelクラスとそれに対応したViewクラスで構成できるようになった．編集時はViewクラスを任意の位置に配置するだけで，画面のデザインが行える．また各UI-object固有の詳細情報は，それぞれのUI-objectの情報設定ダイアログを用いて入力し，Modelクラスに格納する．このため，編集時にはすべてのモダリティをGUIで取り扱うことができる．

図6は音声合成部品の情報設定ダイアログである．この中では，出力内容，男声／女声の選択，発声スピード，音量といった音声合成特有の情報を設定することができ，実行時に出力される音声合成の内容なども簡単に変更できる．つまり，技術の詳細を知らなくても，容易にその要素技術を利用することができる．

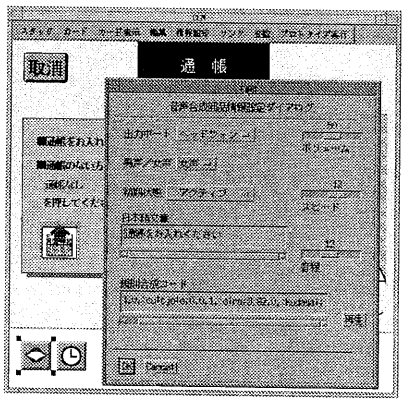

図 6: 音声合成部品の情報設定ダイアログ

このように，MVC-X モデルの枠組みを利用することで，Muse では要素技術の詳細部分をアプリケーション開発者から隠蔽し，容易に利用できる形で提供することができた．

3.3.2 将来の拡張への対応

すべての UI-object は，生成や配置，移動，コピーなど編集時に用いるメソッドを抽象クラスから継承して利用できる．このため，各 UI-object で異なる部分は，実行時の動作に関する記述のみである．すなわち，新たなモダリティの追加は，実行時の動作を記述することで行うことができる．このように Muse では，従来の MUI システムに比較して，より容易にモダリティの追加に対応することができる．

また，例えば音声認識や文字認識の辞書登録を行う機能を Muse に付加するといった Muse 自身の機能拡張を行う場合は，MVC-X モデルの手順に基づき，その機能を担当する Controller クラスを持つサブシステムを追加して実現できる．このとき，既存システムの Controller クラスへの修正は，新しい Controller クラスへのインタフェースの追加を行うだけである．つまり，Muse に新しい機能を追加する場合でも，元々のクラス構造は維持されている．

このように，MVC-X モデルを利用することによって，UI-object および Muse 自身の双方の機能拡張が容易に実現できた．

3.3.3 制御の定式化

Muse で扱うモダリティは，すべて同じ抽象クラスを継承して実現されている．このため，リンクを介したメッセージの伝達方法は，すべての UI-object で共通化されている．メッセージを受け取った UI-object は，メッセージに従った動作を行う．例えば "Active" というメッセージを受け取った場合，イメージ部品は View クラスを表示させ，音声合成部品は合成音声を出力する．このように，同じメッセージを受信しても UI-object の種類によって異なった動作が行われる．Muse では，すべての UI-object で共通なメッセージのやりとりを仕様化し，メッセージに対応した個々の動作を各 UI-object 内で定義することにより，上記の動作を実現する．すなわち，MUI の制御はリンクとメッセージで定式化されている．このため，修正はリンクの貼り替えやメッセージの変更で実現でき，MUI の評価・改良に柔軟に対応することができる．

4 むすび

本論文では，オブジェクト指向モデリングおよび設計を用いたラピッドプロトタイピングツール Muse の開発における，マルチモーダル入出力の取り扱いについて説明した．

Muse では，マルチモーダル入出力を GUI で統一的に取り扱うことで，簡単な操作によって MUI システムを作成できるようになった．特にすべてのモダリティを UI-object を継承して実現しているため，各モダリティの情報設定やリンク設定を，すべてのモダリティに対して共通な方法で行うことができる．このため，デザイナーなどプログラミングに精通していない人でも MUI の評価・改良が容易に行え，社会情報システムの使い勝手向上に大きく寄与するものと思われる．

また，MVC-X モデルによるパターン化技術を利用することで，(1) 機能分析時に適切なオブジェクトとその役割を抽出できる，(2) 分析／設計／実装の各フェーズにおいてクラス構造を維持できる，(3) 機能拡張にも容易に対応できるという，頑強なソフトウェア構造を構築できた．

参考文献

[1] 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, August 1994.

[2] 新田恒雄, "GUI からマルチモーダル UI(MUI) に向けて", 情処誌, vol36, No.11, pp.1039-1046 (1995).

[3] 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, no.NLC95-50, no.SP95-85, December 1995.

[4] J.Rumbaugh, M.Blaha, W.Premerlani, F.Eddy, W.Lorensen, Object-Oriented modeling and design, Englewood Cliffs, NJ: Prentice-Hall, 1991 (日本語訳：羽生田栄一, オブジェクト指向方法論OMT, トッパン).

[5] G.E.Krasner, S.T.Pope, "A cookbook for using the Model-View-Controller user interface paradigm in Smalltalk-80," Journal of Object-Oriented Programming, August/September, pp.26-49, August 1988.

[6] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Design patterns, Addison-Wesley Publish Company, 1995 (日本語訳：本位田真一, 吉田和樹, デザインパターン, ソフトバンク社).