2K-4

MVC を用いたコンポーネント連携型 UIMS の提案

久原　政彦†　　伊藤　誠‡

中京大学†‡

1．はじめに

本稿では、アプリケーション（以下 App）のユーザインタフェース（以下 UI）について、従来のシステムにあった様々な問題を解決するための、新しいシステムの提案を行い、それにより App 開発がどのように変わるかを提示することを目的とする。

2．従来型の App 開発の問題

a) 専門性を生かした開発や拡張

アルゴリズム、UI、フロー管理などを、開発者の専門性を生かして分業するための、明確な基準に乏しい。また、App の修正や拡張を行う際、App の全体を一度再構成(再コンパイル等)しなければならない。

b) UI の自由な差替え

UI 定義が App 内に実装されているため、App の UI を変更するためには、App 自体を書き換える必要がある。

c) デバイス透過性

GUI 以外の UI、例えばハードウェアデバイスを用いる際には、特別な処理が必要である。

d) 位置透過性

操作中の App に対し、プロセス外やネットワーク外にある UI を用いる際には、特別な処理が必要である。

3．問題を解決するモデル

a) MVC―App 設計モデル

MVC は App を Model,View,Controller の 3 元に分割し、互いがメッセージ通信することによって挙動し App を構成する、App 設計モデルである。次表に基づいて分割された機能をコンポーネント（以下 Comp）と呼ぶ。

表 1. MVC の 3 元分離

C	Task
M	アルゴリズム・データ管理担当。C からの通知に応じてデータの処理を行い、V へ変更を通知。
V	UI 担当。M のデータや状態を表示。M からデータの変更が通知された場合、自身の表示を変更。
C	フロー制御担当。ユーザ入力を解釈し、M へデータ操作を、V へ状態変更を、それぞれ通知する。

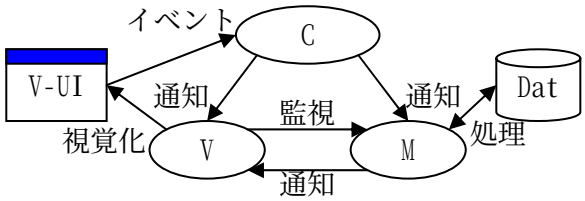

図 1. MVC 基本構成

前表の分類により、App の開発は Comp 単位で明確に分離でき、デザイン専門なら V、アルゴリズム専門なら M と、開発者の専門性を生かした分業が可能となる。

また本システムでは自由な開発・修正・拡張を可能にするため、MVC-Comp を Comp 単位で独立して構築・挙動できるようなシステムをとっている。次図に MVC-Comp の挙動を示すメッセージフローの実例を示す。

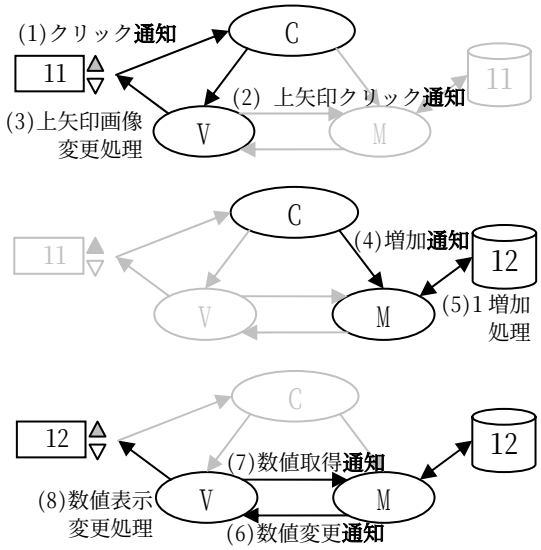

図 2. MVC のメッセージフロー例(括弧数字は処理順)

強調する点として、Comp 間の矢印がメッセージの通知のみという点がある。Comp は他者を直接操作するのではなく、メッセージ通知で挙動依頼をする。受信側はメッセージを介してのみ挙動し、直接的な操作は受けない。結果 Comp は他 Comp との強依存性を取り払われ、メッセージに対応して挙動を行うことを満たせば、その具象実装は問われなくなる。これにより、Comp の自由な開発・修正・拡張が可能となる。

b) UIMS―UI 管理システム

UIMS は、UI を管理するための専用のシステムのことで、App から UI を切り離し、独立して自由な実装をすることを可能とするシステムである。

Proposal of Component Relational UIMS with MVC
†Masahiko Kubara, Chukyo University
‡Makoto Ito, Chukyo University

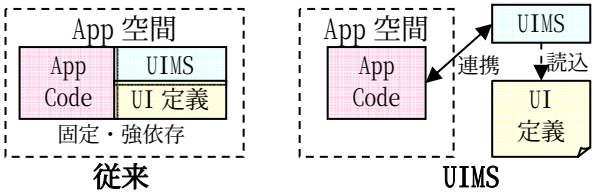

図3. 従来の App と、UIMS を用いた App の違い

UIMS を用いると、App から、UI を管理するシステム、UI を定義する部分が切り離され、独立に開発できる。UIMS は MVC の V を担当し、App の M,C とメッセージ通信を用いて連携しながら、App の UI を管理する。

UIMS が表象する UI の定義も外部化され、定義を書き換えることで、App を変更することなく、UI のみを自由に変更できるようになる。UI 定義の記述には、UI 特有の階層構造を上手く表現できる XML を利用する。

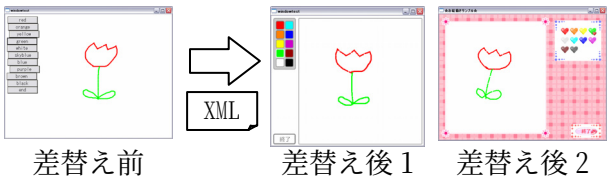

図4. UI の自由な差替え例

c) MVC によるデバイス透過

MVC によって Comp は、メッセージに対して等価に挙動すれば Comp の具象形態を問われない。例えば V の UI が、GUI であっても、ハードウェアであっても、透過的に扱うことができる。また、GUI を使うかハードウェアを使うかを、UIMS によって任意に選択できる。

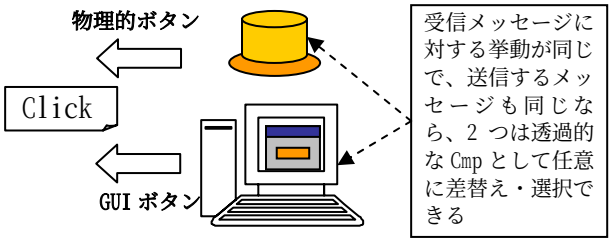

図5. デバイス透過例

d) システムのネットワーク対応による位置透過

メッセージを送信する Comp が、相手先がどこにあるかを意識せずに MVC のメッセージを通知できるシステムを導入する。これにより、Comp がどの位置にいても、透過的に扱うことができる。

4. システム構造

a) Comp・メッセージフローマネージャ

MVC-Comp はすべて、App 起動時に Comp マネージャに登録される。Comp マネージャはメッセージフローのマネージャも兼ねており、送信側 Comp がメッセージを送信した際、受信側へ通知するための仲介役となる。

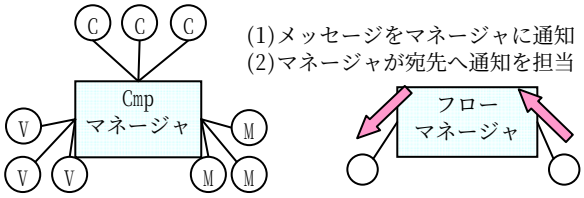

図6. Comp マネージャ、メッセージフロー例

b) 透過的メッセージ通信・シリアライズ

メッセージの宛先が、自分の管理する Comp でない場合、他のマネージャを調べ、宛先が発見できた場合は、そのマネージャ経由でメッセージを通知する。マネージャ間通信には XML シリアライズを用いる。これにより、Comp の位置透過性を実現できる。

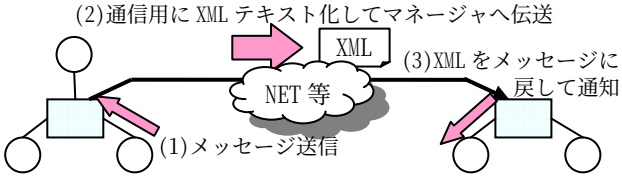

図7. 透過的メッセージ通信、XML シリアライズ例

5. 期待できる改善の例

a) 初心者向け改善―マウスボタンの混同

いずれのマウスボタンを押しても同じ動作をさせる変更を、MVC を用いて、マウスイベントを解釈する C を差替えるだけで実現できる。App 全体の変更は必要ない。通常の挙動に戻すのも、C の差戻しだけで済む。

b) 趣向に応じた改善―メール表示を葉書のように

メール内容を表示する V であるテキストボックスを、MVC と UIMS を用いて、葉書に書いたような表示を行う V に変更できる。この際 App 全体の変更は必要ない。このような UI の改善を、デザインの得意な開発者や、UI 専門のサードパーティに委譲することもできる。

c) デバイス透過性の改善―早押しボタン

クイズゲームの早押し用の GUI ボタン V を、UIMS とデバイス透過性を利用して、実際のボタン V にできる。開発者はデバイスに応じた App 変更を意識しなくてよいうえ、ニーズに応じた拡張が容易に可能となる。

6. おわりに

本稿では提案のみに留まったが、今後はシステムを実際に開発し、その実証実験を行いたい。本システムにより、App 開発がより個々の能力や専門性を生かしたものとなり、洗練されていくことを期待したい。

参考文献

[1] 久原政彦, 伊藤誠: "UIMS に関する諸提案", 平成 17 年度電気関係学会東海支部連合大会, 2005/09