B-029

MDA モデル変換による GUI プロトタイプ生成手法

A Method for GUI Prototype Generation with Model Transformation in MDA

迎田 宙† 岩田 一† 白銀 純子‡ 深澤 良彰†
Hiroshi Mukaeda Hajime Iwata Junko Shirogane Yoshiaki Fukazawa

1. はじめに

近年，MDA(Model Driven Architecture)[1]と呼ばれるソフトウェア開発体系が注目を集めている．これは，UML 等に沿って記述されたモデルをもとに，自動的な変換を繰り返すことで実装を得るという技術である．開発コストの低減や保守性の向上，他のプラットフォームへの移植の容易化といった効果が期待されている．既に対応ツールも登場し始めており，実用化に向けて様々な研究が進められている．

しかし，UML には GUI(Graphical User Interface)設計に関する記述方法が用意されておらず，MDA においても，モデルを記述する段階から GUI に関する設計を組み込むことについては，指針が定まっていない．そのため，個々の MDA 対応ツールによって GUI に対する対応は異なる．従って，よりユーザビリティを意識したソフトウェア開発を行なう際には，生成されたソースコードに手動で変更を行う必要が生じてしまうため，実装の保守性や移植容易性が損なわれる恐れがある．

そこで本手法では，設計段階からユーザビリティを考慮できるよう，GUI の構築方法を盛り込んだ UML の記述方法，および MDA 体系に沿ったモデルの変換による GUI プロトタイプの生成方法を提案する．

2．MDA (Model Driven Architecture)

MDA とは，モデルを中心としてソフトウェアを開発していく方法である．UML 等の標準記法に沿って記述されたモデルに，予め用意された変換ルールを適用することにより，自動的/半自動的にソースコードを得ることができる．ここで，利用するモデルは大きく二段階に分けられ，それぞれ PIM(Platform Independent Model)，PSM(Platform Specific Model)と呼ばれる．

PIM とは，MDA の第 1 段階で描かれるモデルである．OS や言語といったプラットフォームに依存しない形で記述され，複数のプラットフォーム間で共有，再利用され，PIM を起点として様々な OS やプラットフォーム向けのモデルが生成される．

PSM とは，各プラットフォームに特化したモデルである．プラットフォームの情報をもとに，MDA ツールが個々の変換ルールを用いて，PIM を自動変換することで得られる．この変換機構をモデル・コンパイラ[2]と呼ぶ．そしてこの PSM をもとにしてプログラムが生成される．

3. 本手法の特徴

本手法では，MDA に沿って記述された UML 図を自動変換し，ソフトウェアの GUI 部分のソースコードを得ることを目指す．ソフトウェアの GUI 部分に関しては，ユーザビリティの高いものを作成する必要がある．従来の MDA 対応ツールでは，テンプレートによって GUI を決定しているものもある．この場合，対象とする使用者のスキルや使用環境といったユーザビリティに関する情報が考慮されておらず，必ずしもユーザビリティが高いとは言えない．ユーザビリティの向上のためには，GUI のプロトタイプを作成し，その試用をもとにした問題点のフィードバックを繰り返し行うこととなるため，他の開発工程に比べ，作り直しの作業にかかるコストが大きい．また，GUI の作成に対応していない MDA ツールも存在する．この場合，GUI を別個に作成する必要がある．

本手法は，ソフトウェアの GUI 部分の作成を自動化するため，各プロトタイプを作成するコストを低減し，開発全体のコスト低減に貢献することができる．特に，ユーザビリティを考慮するうえで重要となる，ウィンドウの表示順序やソフトウェアの操作手順といった，プログラム全体に影響を及ぼすような大きな修正が容易に行えるようになる．

また GUI は，プラットフォームが異なると利用可能なウィジェットが異なることも多いため，実装方法の違いにより，ユーザビリティに大きな影響を与える可能性がある．本手法で使用する PIM は OS やプログラミング言語といった特定のプラットフォームに依存せず記述可能であり，個々のプラットフォームの特性や差異を意識することなく開発が可能である．また，特定のプラットフォームに依存しない抽象度の高いモデルをもとに個々のプラットフォームに特化した詳細度の高いモデルを自動生成することから，複数のプラットフォームに対する並行開発や移植が容易になる．

一方で，本手法で用いるモデルは GUI に関する記述のみに特化しており，内部処理やデータのやり取りに関する情報は扱わない．そのため，表示部分と内部処理との結合度の低い開発が可能となり，表示部分の修正が内部処理に与える影響を小さく抑えることができる．

4. システム構成

本システムのアーキテクチャを図 1 に示す．本システムでは，ソフトウェアの操作の流れを PIM として記述し，PSM に変換し，最終的にソースコードを生成する．このとき，PSM の変換に複数段階を用意し，徐々に詳細化していく方法をとる．また，PIM および PSM は，UML のアクティビティ図で表すものとする．

4.1 PIM の記述

1 つのアクティビティは，1 つ 1 つのユーザの入力やソフトウェアの処理結果の表示などのインタラクションを表し，アクティビティ同士をアークで結ぶことで，処理の流れを表す．また，ステレオタイプでそのアクティビ

† 早稲田大学
‡ 東京女子大学

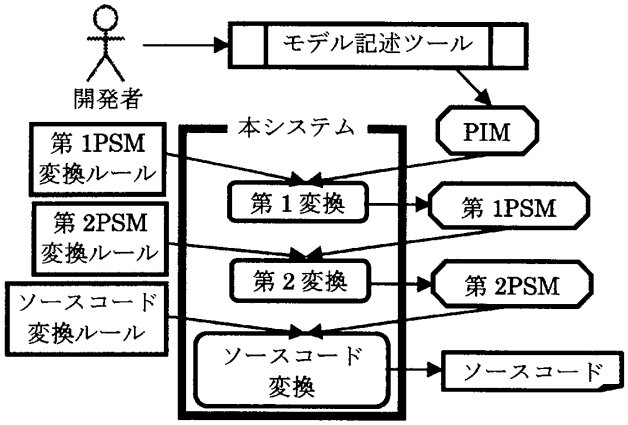

図 1：モデル変換の流れ

ティでのインタラクションの内容を示す．インタラクションの内容を示す方法として，6 種類のものを定義する．この種類と意味を表 1 に示す．この記述例を図 2 に示す．図 2 は，図書館における書籍検索のインタラクションの流れである．ソフトウェアが最初にメイン画面を表示し，ユーザが作者名を入力し，作品名を選択してソフトウェアがユーザの入力内容の確認を表示するというものである．[action]と[separate]は遷移を表すアーク上に記述する．

表 1：PIM のインタラクション名と内容

インタラクション名	内容
display	表示
input	入力
select–single	単数選択
select–free	任意数選択
action	アクション（ユーザの行動によって起こる能動的な処理）
separate	処理の区切り

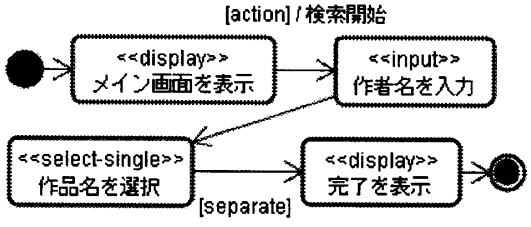

図 2：書籍検索システムのアクティビティ図

4.2 PSM の生成

PSM は，PIM を段階的に詳細化していくことにより生成する．その際，PIM での抽象的なインタラクションを各プラットフォーム固有のウィジェットに対応させる．現状では 2 段階の変換を行う．第 1 段階目では各インタラクションに対してプログラミング言語に依存しない，どのプログラミング言語でも実装可能なウィジェットを割り当てる．第 2 段階目ではプログラミング言語に依存した形でウィジェットを割り当てる．PIM での各インタラクションと第 2 段階目でのウィジェットとの対応の一部（Java Swing の場合）を表 2 に示す．1 つのインタラクションに複数種類のウィジェットが対応する場合は，本システムでデフォルトの対応付けルールも用意しているが，それ以外のものを利用したい場合には，ソフトウェア開発者が選択し，変換ルールに適用する．

表 2：インタラクションとウィジェットの対応例

インタラクション	対応するウィジェット
input	JTextArea JTextField
select–single	JRadioButton JComboBox
action	JButton

4.3 ソースコードの生成

ソースコードは第 2 段階目の PSM を自動変換することで得られ，PSM 内で指定されたウィジェットの種類およびインタラクション遷移の手順に沿って，ウィジェットの生成やウィンドウへの配置が指定される．図 2 の PIM を変換して生成されたソースコードの実行例を図 3 に示す．ウィンドウやウィジェットを表示する上で最低限必要になる値の設定は変換ルールを用いて行う．イベントハンドラやその他の内部処理に関しては自動生成されず，開発者が書き足すこととする．

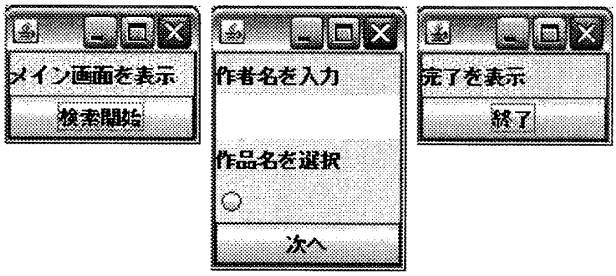

図 3：ソースコードの実行例

5. 終わりに

本手法では，モデルによる GUI の記述方法，および MDA 体系に沿って自動/半自動変換する方法について提案した．今後の課題は以下のとおりである．

- より抽象度の高い PIM 記述法の提案
- GUI レイアウトのモデルへの組み込み方法の提案
- 生成した GUI と内部処理との連携の生成手法の提案

6. 参考文献

[1] スティーブ J. メラー, ケンドール スコット, アクセル ウール, ディルク ヴァイセ 著, 二上貴夫, 長瀬嘉秀 監訳: “MDA のエッセンス”, 翔泳社 (2004)

[2] スティーブ J. メラー, マーク J. バルサー 著, 二上貴夫, 長瀬嘉秀 監訳 : “Executable UML”, 翔泳社 (2003)