Vol.2011-SE-172 No.7
Vol.2011-EMB-21 No.7
2011/5/18

GUI を考慮した MDA 開発手法の提案

井上尚紀† 岸知二†

近年 MDA による開発が普及し始めてきた. しかし, MDA における GUI の扱いは, 幾つかの方法が提案されてはいるが, 未洗練である. そこで, 本研究では, PIM に抽象的な GUI の情報を組み込むことにより, レイアウト指定の際の手作業の削減とプラットフォームの変化に対応できる MDA 開発手法を提案する.

The Proposal of MDA-Development Technique Considering Graphical User Interface

Naoki Inoue† and Tomoji Kishi†

Recently, MDA-Development is becoming popular. Though there proposed multiple techniques of handling GUI information in MDA, no technique is widely accepted. In this paper, we propose an approach in which abstract GUI information is merged into PIM. Our approach has advantages in the cost of layout definition and GUI platform independency.

1. 研究背景

1.1 はじめに

ソフトウェアをとりまくビジネス環境や実装技術の進歩は日々変化しており，変化に強いソフトウェア開発が求められている．そこで，近年注目されている開発手法のひとつが標準化団体 OMG(Object Management Group)が提唱している MDA(Model Driven Architecture : モデル駆動アーキテクチャ) 1)であり, 様々な分野での適用が検討されている.

一方，業務などに利用されるソフトウェアの開発では，ユーザインターフェイスがどのぐらい分かりやすく使いやすいかがソフトウェアの品質に大きな影響を与える.

しかし，MDA における GUI(Graphical User Interface)の扱いは，幾つかの方法が提案されてはいるが，未洗練である．そこで本研究は，GUI を考慮した MDA 開発方法を提案する事を目的とする. 今回は, 特にウィンドウシステムの画面構成に焦点をあて, 画面構成に関わる情報を MDA で扱うための一手法について提案する．具体的には, UML モデリングツールによって作られる通常の PIM に対して，GUI ビルダを使用して作られる抽象的な GUI（画面構成）の情報を組み込んだ GPIM（拡張 PIM）を作成することにより，レイアウト指定の際の手作業の削減とプラットフォームの変化に対応できるようにすることをねらう.

評価実験の結果，本手法は，GUI レイアウト情報を含めたプラットフォーム独立な GPIM から，ソースコードを生成することができ，複数のプラットフォームに対応できる事を確認した.

1.2 背景知識

1.2.1 MDA

MDA は OMG が提唱しているソフトウェア開発手法である．CIM(Computation Independent Model : 計算処理非依存モデル)を参考に，実装技術から独立した PIM(Platform Independent Model : プラットフォーム独立モデル)を作成し，PSM(Platform Specific Model : プラットフォーム依存モデル)へ変換し，PSM からソースコードを生成する．MDA におけるプラットフォームとは，Java, CORBA, XML のようなプログラミング言語や実現方式等の実装技術を指す.

MDA の利点のひとつは, プラットフォームに依存しないモデル(PIM)を作成する事により，プラットフォームの進化があったとしても PIM を再構築しなくてよい所にある．また，ソースコードの自動生成によるプログラミング工程の効率化や，コーディング技術の差を埋めることによるソースコード品質の底上げ等も利点である．図 1-1 に MDA の全体像を示す．(2)より引用）

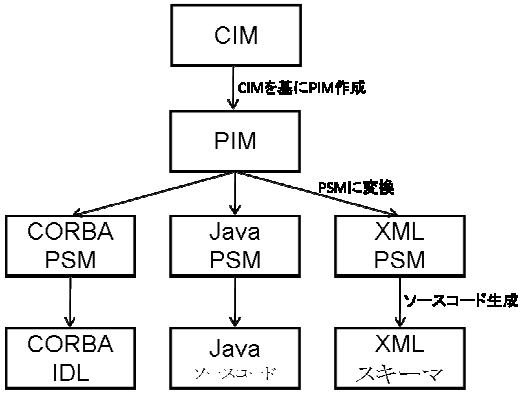

図 1-1 MDA 全体像

† 早稲田大学 経営システム工学科
Department of Industrial and Management Systems Engineering, Waseda University

Vol.2011-SE-172 No.7
Vol.2011-EMB-21 No.7
2011/5/18

以下，CIM, PIM, PSM の各モデルについて説明する．

- CIM

CIM とは，コンピュータ技術等の計算処理に依存しないモデルである．システム自体から独立したビジネスプロセスを表現する．例えば，システムの入出力やユーザからの要求を表現する．

- PIM

PIM とは，OS やプログラミング言語といったプラットフォームに依存しない業務用件モデルである．CIM を基に手動で作られる．

- PSM

PSM とは，Java 等の特定のプラットフォームに特化したモデルであり，プラットフォームの特性を加味したモデルである．PSM は PIM から変換されて作成され，PSM への変換ルールに従い，一つまたは複数の PSM に変換される．一度変換ルールを定義すれば，そのルールを繰り返し使用することができる．

更に，変換されて作成された PSM からソースコードを自動生成する．この自動生成の際も変換ルールを定義する必要がある．

1.2.2 GUI

GUI(Graphical User Interface)は，電子化された視覚表示媒体により，仕事を達成するのに必要な操作情報をグラフィカルにユーザに提供するインターフェイスを意味する．現在のソフトウェアの一般的なユーザインターフェイスである 3)．様々なユーザに利用されるソフトウェアの開発では，ユーザインターフェイスがどのぐらい分かりやすく使いやすいかがソフトウェアの品質に大きな影響を与える 4)．

2. 従来手法と研究アプローチ

2.1 現状分析

近年，MDA 開発が普及し始めてきた．しかし，MDA 開発における GUI の扱いは，幾つかの方法が提案されてはいるが，未洗練である．一般的な MDA 開発では，GUI 部分は MDA とは別で作られる．その理由の一つとして，GUI 開発はプラットフォームに依存しやすいことが挙げられる．通常 GUI 開発は GUI ビルダを使用することが多いが，GUI ビルダを使用する時点でプログラミング言語が限定されてしまうことも多い．また，GUI のレイアウト情報をモデルに表現することが難しい事も，その理由の一つである．

2.2 従来手法とその問題点

Stefan Sauer et al. 5) は，UML モデリングツールで記述されたステートチャート図を PIM とし，PSM への変換の過程で GUI ビルダを用いて GUI の情報をモデルに組み込むことによって GUI 開発を効率化している．対応するプログラミング言語は Java のみである．しかし，本手法は，PIM ではなく PSM に GUI の情報を組み込んでいるので，Java プラットフォームにのみの対応であり，プラットフォームの変化に対応することができない．

一方，Stefan Link et al. 6) は，アクティビティ図とクラス図を UML プロファイルによって GUI の情報を扱えるように軽量拡張している．アクティビティ図に手入力で GUI の情報を加え，それをクラス図へ自動変換し，手入力で PIM に GUI の情報を加え，そのクラス図を PIM とし，PSM へのモデル変換後，手動でモデルを洗練し，ソースコードを生成し，手動でコーディングしている．本手法は，MDA に則していて手法として分かりやすいが，手作業が非常に多く，レイアウトの情報をひとつひとつの GUI エレメント毎に指定しなければならなく，開発効率が良くないことが問題点として挙げられる．

2.3 今回取り扱う問題と研究アプローチ

本研究は MDA 開発をより実用的にする為に，GUI を考慮した MDA 開発方法を提案する．特に今回は従来手法の問題点を踏まえ，以下の２つを改善することを目標とする．

1. プラットフォームの変化に対応できない
2. レイアウト指定の負担が大きい

これらの問題を改善する為に，本研究では以下のアプローチを用いた GUI の MDA 開発手法を提案する．

- PIM への GUI 情報の組込み

問題１を改善するには，PSM でなく PIM に GUI 情報を組み込む必要がある．その為に，プラットフォームに依存した GUI の情報でなく，抽象的な GUI の情報をモデルに組み込む．本研究が指す GUI 情報とは，GUI レイアウトを指定する為に必要な情報である．

- 抽象 GUI ビルダを使用

問題２を改善する為に，抽象 GUI ビルダを利用し，レイアウト指定の負担を減らす．通常，GUI ビルダは，座標の表現方法等をプログラミング言語に依存した形で出力するが，抽象 GUI ビルダは，言語に依存しない形で抽象 GUI 情報を出力する．

3. 提案手法

前章で述べた研究アプローチをもとに，手法の提案を行う．

3.1 前提条件

本手法は，MDA 開発において GUI のアプリケーションソフトウェアを開発する為の手法である．今回はプラットフォーム依存性の高い画面設計をいかに MDA で扱うかという問題にフォーカスする為，画面遷移の問題は扱わないこととする．

Vol.2011-SE-172 No.7
Vol.2011-EMB-21 No.7
2011/5/18

3.2 提案手法概要

提案手法の概要を図 3-1 に示す．アプリケーション開発者は最初に作成した CIM を基に抽象 GUI ビルダにより抽象 GUI 情報を作成し，並行して UML モデリングツールにより PIM（クラス図）を描画する．次にそれぞれを統合したものを GPIM（拡張クラス図）とし，PSM へモデル変換し，ソースコードを生成する．拡張クラス図とは，GUI 情報を扱えるように拡張したクラス図である．

PSM への変換とソースコード生成は従来技術を用いる．

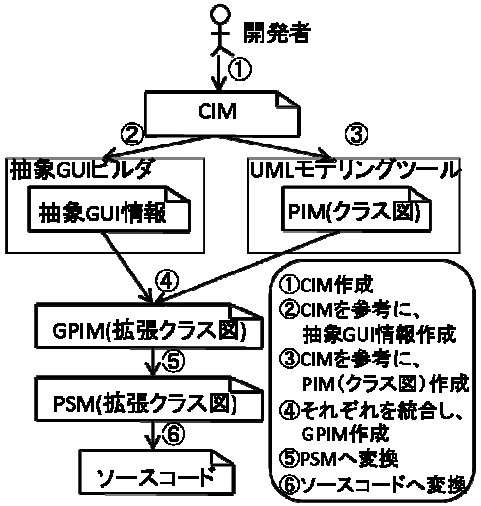

図 3-1 提案手法の概要図

3.3 各ステップ詳細

3.3.1 ①CIM 作成

CIM にはシステムによって扱われる情報のフローを記述し，特に，入力させる情報や選択させる情報を記述する．また本手法では，CIM はアクティビティ図やユースケース図を用いることを想定している．処理フロー上の複数の選択肢から，ユーザに何かを選択させる場合は，その選択肢もここで記述する．

3.3.2 ②抽象 GUI ビルダによる抽象 GUI 情報作成

CIM で示された業務フロー中のコンピュータと人とのやりとりを，具体的にどのような画面構成で実現するかを決定し，抽象 GUI 情報として出力する．XML を用いた XMI がモデリングツールとソフトウェア生成ツールの間のモデル交換媒体として一般に使われている為，抽象 GUI 情報は XML 形式とした．

幾つかの言語の GUI 機能を比較した結果，多くの言語で共通の機能や属性があった．そこでそれらを含んだものを抽象 GUI 情報とした．具体的には以下 A)～E)の情報を含む．

A) GUI エレメントの種類

GUI エレメントは，入力エレメント，出力エレメント，選択エレメント，動作エレメント，コンテナエレメントの５タイプに分けられ，それぞれ表 3-1 のように数種類のエレメントがある．この分類は 6)を参考にした．

表 3-1 GUI エレメントの種類

種類	入力エレメント	出力エレメント	選択エレメント	動作エレメント	コンテナエレメント
要素	テキストフィールド，パスワードフィールド，整数型フィールド，少数点型フィールド，テキストエリア	ラベル，ツールチップ，プログレスバー	ラジオボタンリスト，チェックボックスリスト，コンボボックス，リストボックス	メニュー，コマンドボタン	フレーム，パネル，メニューバー

各タイプのエレメントは次のような機能を持つ．

入力エレメント：キーボードによって入力させるエレメント
出力エレメント：画面上にテキスト等を表示するエレメント
選択エレメント：一つ或いは複数の選択肢から選択させるエレメント
動作エレメント：動作（振る舞い）のきっかけとなるエレメント
コンテナエレメント：他のエレメントを所有するエレメント

チェックボックス，ラジオボタンは，複数のエレメントをグループ化し，それぞれチェックボックスリスト・ラジオボタンリストとする．

B) GUI エレメントの位置と大きさ（座標情報）

座標系は GUI 描画では一般的と思われる，左上を原点としたピクセル座標系とする．位置に関してはローカル座標系を利用し，フレームの左上を基準としてそれぞれピクセル単位で，GUI エレメントの左上の点の，上からの位置を PositionTop，左からの位置を PositionLeft とする．同様に大きさに関しても GUI エレメントの左上の点から，下までの長さを Height，右までの長さを Width とする．図 3-2 に位置と大きさについての図を示す．なお今回は作成時と実行時の画面の解像度は同一とし

Vol.2011-SE-172 No.7
Vol.2011-EMB-21 No.7
2011/5/18

たが，この部分の柔軟性を上げることは今後の課題とした．

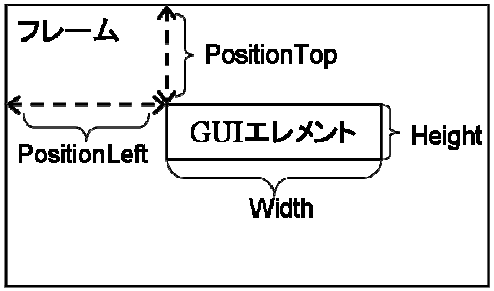

図 3-2 GUI エレメントの位置と大きさ

C) GUI エレメントが持つテキスト

出力エレメント（プログレスバー以外）や，選択エレメント，動作エレメントはテキストを持っている．

D) ラベルと他の GUI エレメントの対応関係

入力エレメントと選択エレメントは基本的にラベルと対応を持っている．例えば，入力エレメントの場合，何を入力させるのかを説明するラベルと対応する．抽象 GUI ビルダにより，どの GUI エレメントとラベルが対応関係を持っているか指定する．

E) その他 GUI エレメント固有の情報

GUI エレメントの種類によってそれぞれ固有の情報を持つ場合がある．入力エレメントの場合は初期値を持ち，フレームの場合はタイトルを，チェックボックスリスト・ラジオボタンリストの場合は代表エレメント以外の位置情報を持つ．

3.3.3 ③UML モデリングツールによる PIM 作成

CIM を基に，扱いたい情報と機能のみをクラス図により PIM をモデリングする．CIM の業務フロー上に選択肢がある場合は，列挙型を使い選択肢を宣言し，属性の型とする．これは，同じく選択肢がある CIM を基に設計した抽象 GUI と対応付ける為である．また，本手法の制約として，クラスは二つでなければならなく，その内一つのクラスは属性（扱いたい情報）のみを持つ．これは，本手法が画面遷移に対応していない為と，一つのクラスが GUI エレメントに変換される為である．

3.3.4 ④PIM と抽象 GUI 情報の統合

図 3-3 に PIM と抽象 GUI 情報の統合ルールを示す．PIM と抽象 GUI 情報を統合する為に，エレメントの名称とクラスや属性の名称の対応をとる必要がある．今回の実験では名称で対応をとることにしたため，統合する前に，PIM と抽象 GUI 中のモデル要素の名前をルールで扱えるように変更する．

抽象 GUI 情報は，PIM と抽象 GUI 情報を統合し易くするように，動作エレメントの名前を“表示テキスト”にリネームし，入力・選択エレメントの名前は，関連付けされたラベルの表示テキストにリネームする．また，関連付けされたラベルも，GPIM での視認性向上の為に，「“表示テキスト”＋“Label”」にリネームする．

PIM はまず，操作を持たないクラスの属性をクラス化し，元の親クラスに所有させる．更に，列挙型で宣言された選択肢がある場合には，その選択肢をその属性クラスの属性として持たせる．

統合する際には，PIM の属性クラスを所有している，親クラスをフレームに置換し，親クラス名をフレーム名にリネームする．また，属性クラス名と同名のエレメントを置換する．次に，関連付けされていたラベルも，フレームに所有させ，関連線を引く．最後に，置換されていないエレメントもフレームに所有させ，動作エレメントがある場合は操作を持つクラスに関連付ける．

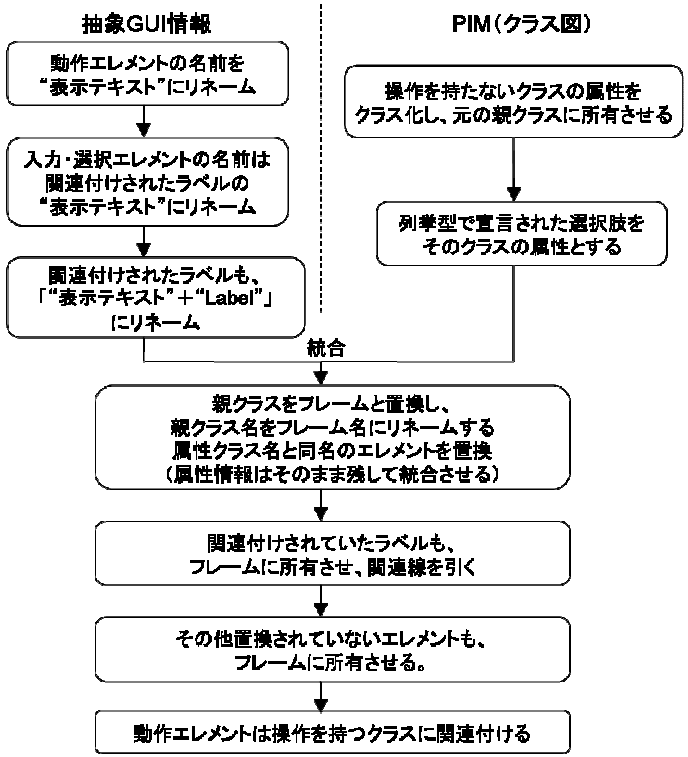

図 3-3 PIM と抽象 GUI 情報の統合ルール

便宜上，図 3-3 の操作により作られる，PIM に GUI 情報を含んだ拡張クラス図を GPIM と呼ぶことにする．GPIM のメタモデルを図 3-4 に示す．

Vol.2011-SE-172 No.7
Vol.2011-EMB-21 No.7
2011/5/18

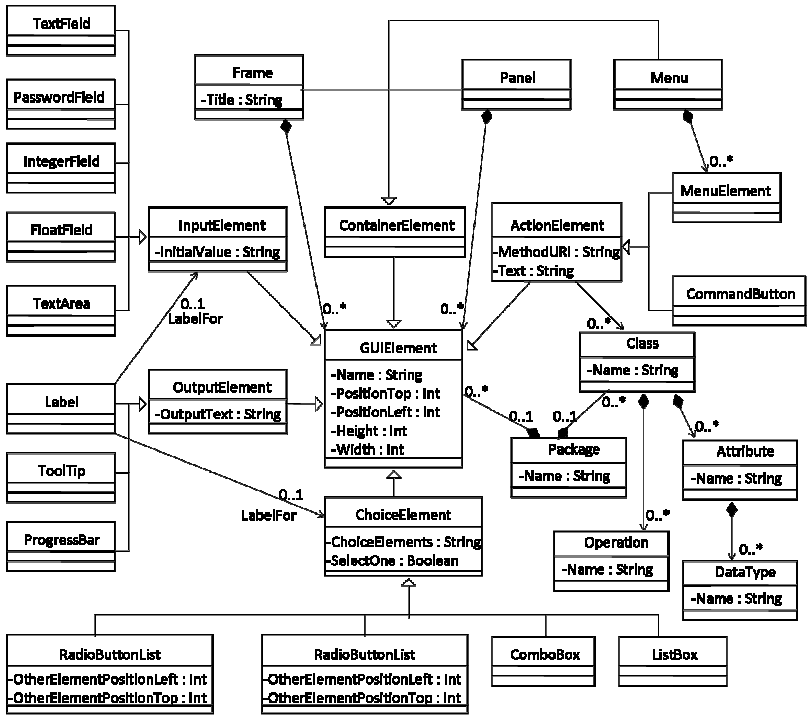

図 3-4 GPIM メタモデル

3.3.5 ⑤PSM へのモデル変換

GPIM の表現を，GUI エレメントについては，抽象的な GUI の情報をプログラミング言語に依存した形に，その他のクラスについては従来技術を用いて変換する．具体的には,各エレメントの名称をプログラミング言語に依存した形に変換する.例えば，フレームである Frame の場合，Java 依存では JFrame に，C#依存では Form になる．また，座標系の原点が左上でなく，一般的なピクセル座標系とは違う場合は，座標情報をこの時点で変換する.例えば左下が原点の Objective-C の場合は,Y 座標を反転する．PSM の依存しているプログラミング言語によってメタモデルが違うが，例として図 3-5 に Java プラットフォーム依存の PSM メタモデルを示す.

3.3.6 ⑥ソースコード生成

PSM からソースコードを生成する．プログラミング言語によって変換ルールは違うが，Java の場合の例を挙げると，GUI を形成するクラスと，内部処理やメインメソッドを持つクラス，計２つのクラスを生成する．PSM への変換とソースコード生成は従来技術を用いる．振る舞い部分の生成方式は本提案の本質部分ではないので，手法としては既存技術を使う．

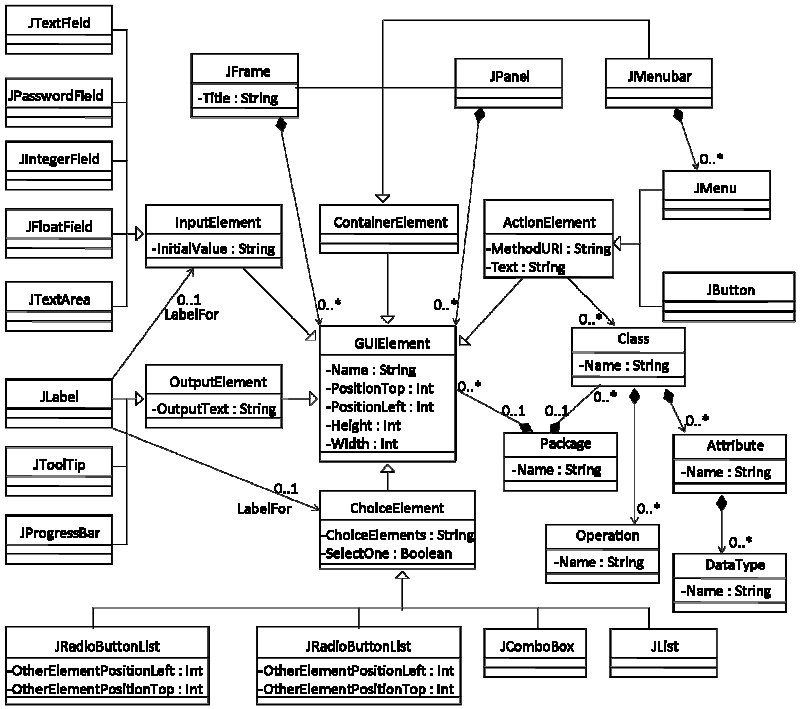

図 3-5 JavaPSM メタモデル

4. 評価実験

4.1 MDA 環境の実装

本研究の実現可能性を実験により実証する為に，MDA 環境を実装する．統合開発環境(IDE)として，Eclipse7)を用い，MDA の実装に EMF(Eclipse Modeling Framework) 8)を用いた．GPIM から PSM への変換には ATL(ATLAS Transformation Language)9)を，PSM からソースコードへの変換には Acceleo10)を利用した．プラットフォームは，GUI ソフトウェア開発において一般的な Java および C#プラットフォームを選定した．また今回は，抽象 GUI ビルダは実装せず，NetBeans IDE11)を利用し，その情報を抽象化する事によって代用し，GPIM モデルを手入力によって作成する．

Vol.2011-SE-172 No.7
Vol.2011-EMB-21 No.7
2011/5/18

4.2 評価実験の目的

本研究では，抽象 GUI ビルダを用いて抽象 GUI 情報を作成し，抽象 GUI 情報をプラットフォーム依存の PSM ではなく，プラットフォーム独立の GPIM に含ませることにより，GUI を考慮した MDA 開発手法を提案することを目的とした．このことを踏まえ実験の目的を以下のように定めた．

① 実現可能性の確認

抽象的な GUI 情報を含んだ GPIM から GUI アプリケーションのソースコードを生成できるかを確認する．更に，GPIM から複数の PSM を生成し，プラットフォームの変化に対応可能か確認する．また，GUI ビルダで指定した GUI と本研究によって生成された GUI を比較し，GUI ビルダで指定した通りの GUI が生成できるかどうか確認する．

② GUI 設計の PIM との独立性の確認

GUI 設計が PIM と独立しているか確認する．

4.3 評価実験の方法

評価実験には実装した MDA 環境を利用する．評価実験では，本手法によりサンプルプログラムを作成し，評価する．このサンプルプログラムは，個人情報を GUI の入力フォームによって入力させ OK ボタンを押すと，内容を出力したテキストファイルが作成される，というプログラムである．

① 実現可能性の確認方法

実装した MDA 環境により生成したアプリケーションを実行し，GUI ビルダで指定した GUI と生成された GUI の比較を行う．また，Java と C#のソースコードを同じ GPIM から生成し，実行結果の比較を行う．

② GUI 設計の PIM との独立性の確認方法

①の確認で作成したサンプルプログラムと同じ CIM，PIM で，抽象 GUI ビルダによる抽象 GUI 情報のみを変更したサンプルプログラムを作成し，①の実験で作成したサンプルプログラムと比較する．

4.4 実験結果と考察

① 実現可能性の確認実験の結果と考察

上述の内容のサンプルプログラムの CIM を，アクティビティ図で作成したものが図 4-1 である．上述のように，個人情報を入力すると，テキストファイルが作成されるというプログラムである．

次にこの CIM を基に，UML モデリングツールにより，PIM であるクラス図を作成した．PIM は図 4-2 のようになった．

また，CIM を基に，GUI ビルダで図 4-3 のように GUI を指定し，抽象 GUI 情報を出力した．

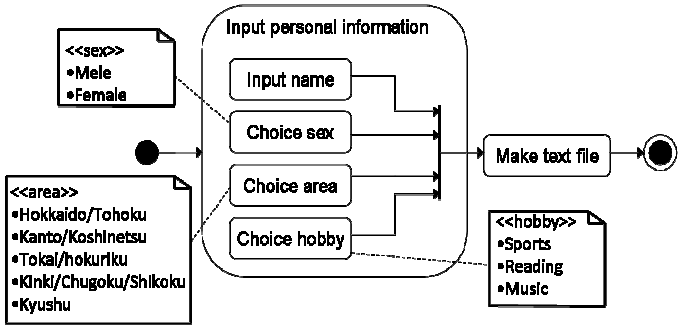

図 4-1 作成した CIM

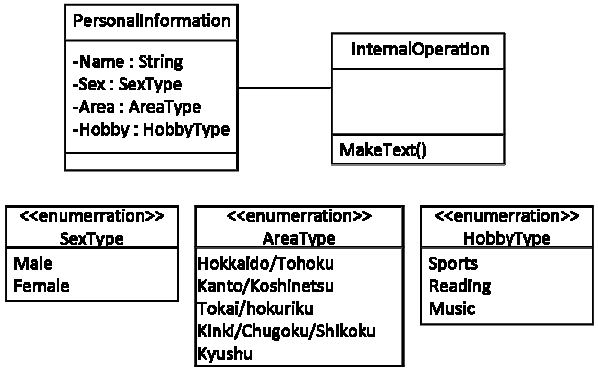

図 4-2 作成した PIM

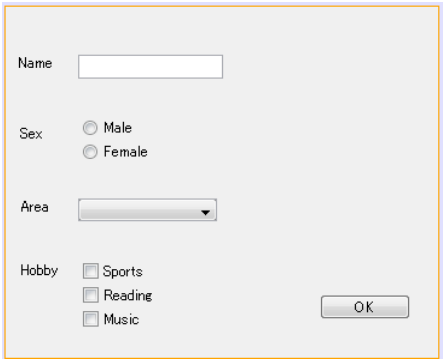

図 4-3 GUI ビルダで指定した GUI

Vol.2011-SE-172 No.7
Vol.2011-EMB-21 No.7
2011/5/18

次に，図 4-2 の PIM と抽象 GUI 情報を 3.3.4 の統合ルールに従って統合し，図 4-4 のような GPIM を作成した. それを Java 依存の PSM に変換したものが図 4-5 である.

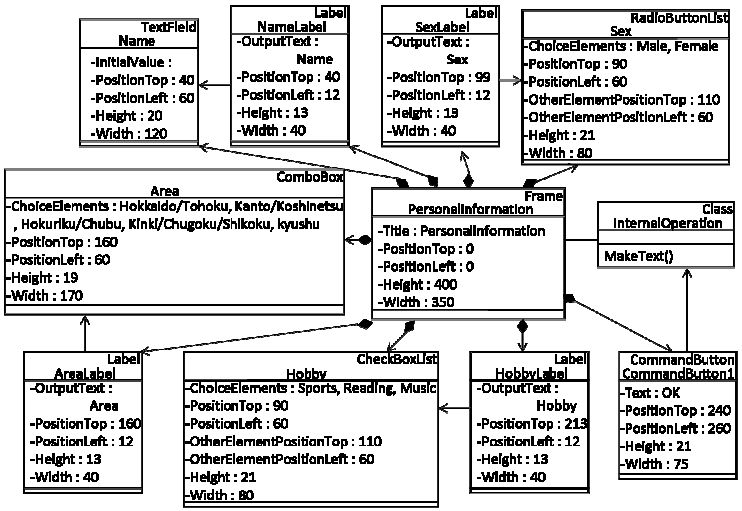

図 4-4 作成した GPIM

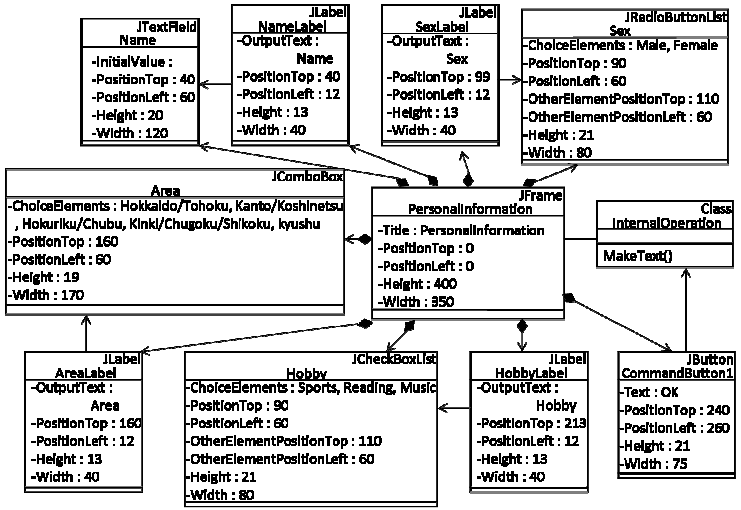

図 4-5 Java プラットフォーム依存の PSM

最後に，実装した Acceleo を用いて PSM からソースコードを生成した．その実行結果が図 4-6 であり，GPIM から GUI アプリケーションのソースコードを生成できること確認した．更に，C#によるサンプルプログラムの実行結果は図 4-7 であり，Java での実験結果とほぼ同じ結果となり，本研究の実現可能性を確認できた．また，GUI ビルダで指定した通り（図 4-3）の GUI となっていることも確認できた.

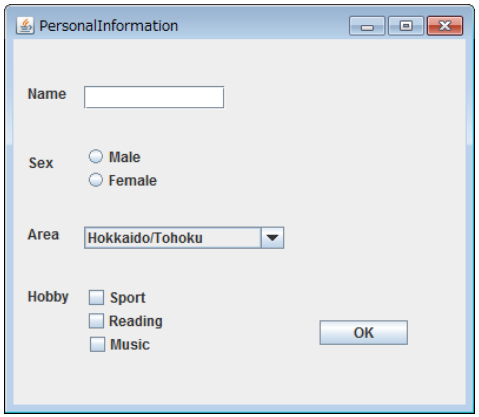

図 4-6 サンプルプログラム(Java)実行結果

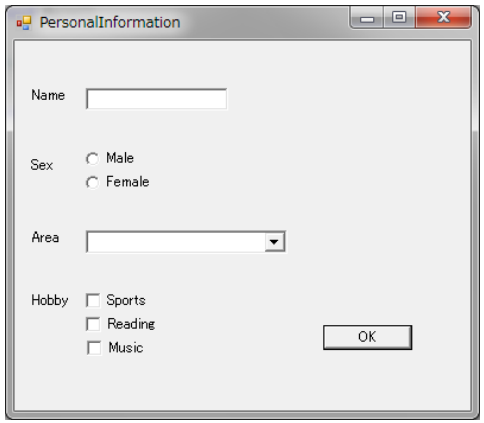

図 4-7 サンプルプログラム(C#)実行結果

② GUI 設計の PIM との独立性の確認実験の結果と考察

①と同様の CIM と PIM で，抽象 GUI 情報のみを変更したサンプルプログラムを，実装した MDA 環境により作成し，確認した．実行結果は図 4-8 となった.

Vol.2011-SE-172 No.7
Vol.2011-EMB-21 No.7
2011/5/18

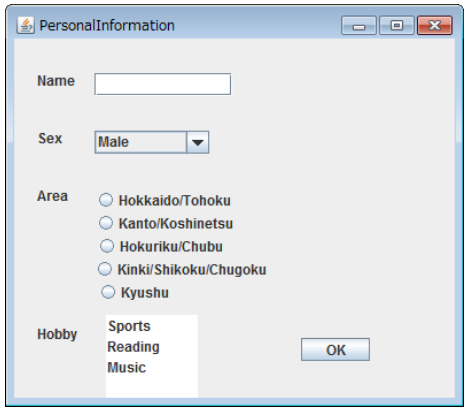

図 4-8 抽象 GUI 情報のみ変更したサンプルプログラム実行結果

抽象 GUI 情報を変更する事により，ラジオボタンやコンボボックスといった選択エレメントを変更できることを確認した．①と同じ CIM・PIM でも，違う GUI を生成することができ，GUI 設計の独立性を確認できた．

5. おわりに

5.1 結論

MDA において GUI を考慮することは重要であるが，既存の手法では，レイアウト指定に手間がかかる，プラットフォームの変化に対応できない，という課題があった．そこで本研究は，GUI ビルダを用いてレイアウト指定の負担を減らし，プラットフォームの変化に対応できるよう PIM に GUI 情報を組み込む MDA 手法を提案した．本手法は，抽象 GUI 情報と PIM を別々につくり，それぞれを GPIM として統合した上でソースコードを生成することにより，プラットフォーム独立性を高めるものである．

そして，MDA 環境の実装を行い，本研究の実現可能性を示すための評価実験を行った．評価実験の結果から，GUI ビルダで指定した通りの GUI が生成できることが確認できた．また，Java と C#，二つのプラットフォームで同じ GUI を生成できることを確認した．以上より，本研究が提案する手法の実現可能性を確認できた．また，抽象 GUI 情報と PIM の独立性も確認できた．

今回は，抽象 GUI ビルダは実装せず，NetBeans IDE11)を利用し，その情報を抽象化する事によって代用し，GPIM モデルを手入力によって作成したが，抽象 GUI 情報と PIM から GPIM の作成は自動化可能であり，将来的には GPIM における GUI レイアウト指定を自動化できると考えられる．

5.2 今後の課題

今後の課題として，画面遷移のあるプログラムへの適応，解像度の違いへの適応，が挙げられる．

- 画面遷移のあるプログラムへの適応

本研究では対象を，画面遷移のない，比較的小さなソフトウェア作成に限定したが，GUI ソフトウェアの多くは，画面遷移を行っている．これには画面遷移をモデルに表す難しさや，PIM との対応関係などの問題があるが，これを解決できればより汎用的で実用的な手法になると考えられる．

- 解像度の違いへの適応

本研究では，GUI を指定する PC の解像度と，アプリケーションを実行する PC の解像度は同じという条件で行った．しかし，現実には作成と実行で別の解像度であることが普通である．解像度もプラットフォームの一つと考えることができるので，実用的な手法を目指す上でこの問題は必ず解決する必要があると考えられる．

参考文献

1) OMG「OMG Model Driven Architecture」
http://www.omg.org/mda/
2) 吉田裕之：「基礎からわかる MDA」，日経 BP 社(2006)初版.
3) 日本人間工学会・アーゴデザイン部会　スクリーンデザイン研究会：「GUI デザイン・ガイドブック」，海文堂出版株式会社(1997)第 2 版.
4) 古賀直樹：「UI デザインの基礎知識」，株式会社技術評論(2010)初版.
5) Stefan Sauer, Gregor Engels：「Easy model-driven development of multimedia user interfaces with GUIBuilder」, Universal Access in Human Computer Interaction - Coping with Diversity, pp.537-546(2007).
6) Stefan Link, Philip Hoyer, Sebastian Abeck, Thomas Schuster：「Focusing Graphical User Interfaces in Model-Driven Software Development」, Advances in Computer-Human Interaction, pp.3-8(2008).
7) 宮本信二：「Eclipse 3.2　完全攻略」，ソフトバンククリエイティブ株式会社(2006)初版.
8) Eclipse.org「EMF」
http://www.eclipse.org/modeling/emf/
9) Eclipse.org「ATL」
http://www.eclipse.org/atl/
10) Eclipse.org「Model to Text(M2T)」
http://www.eclipse.org/modeling/m2t/
11) NetBeans IDE, http://ja.netbeans.org/