Vol.2011-SE-174 No.8
2011/11/2

統一的表現に向けた形式表現と UML との連携

阿部　睦[†]

仕様記述のために，UML およびその派生言語が利用されることが多いが，その利用においてはいろいろな課題もある．そこで，報告者は仕様としての表現が求められる内容を網羅でき，実装に対する設計情報等も表現可能な形式表現の開発を行った．しかしながら，新たな表現方法を提案するだけでは，導入コストが大きい等の理由のため，実際の開発への適用が困難である．そこで，従来技術での表現物を開発した形式表現で利用可能とすることで，表現物やツール等の従来技術での資産を活用したり，従来技術と併用することで利用を促すことが期待できる．

本報告では，開発した形式表現と従来技術である UML との連携方法や得られた考察について報告する．

Description form for Integrated Representation and its coordination with UML

Mutsumi Abe[†]

UML and its derivative language are used for describing specification. However, there are various plobrems using these languages. Reporter developed the description form that has caability to describe the specification and design information for system development.However, It is difficult to apply the new representation method to actual development because of the highly introduce cost,etc.So,This paper reports study of the description form and its coordination with UML.

1. はじめに

従来の仕様表現技術として UML[1]やその派生言語である SysML[2]，MARTE[3]等があるが，利用にあたっての課題があるため，報告者は統一的に仕様が記述可能な形式表現について検討を行い，例題を用いた評価を行った[4].

しかしながら，新たな表現方法を開発しただけでは，実際の開発への導入には以下にあげる項目が一般的な課題として存在すると考える．

・開発者の表現方法の学習コスト

開発に忙しい開発者に新たな表現方法を習得してもらい，さらにそれを利用して開発を行ってもらうのは開発者の負担が大きい．

・開発資産の再利用コスト

新たな表現方法を使った開発を行う際に，既に何らかの言語等によって表現された仕様を利用するためには，新たな表現方法へ置き換えるためのコストがかかる．

・不十分な開発環境による開発効率低下

新しく開発された表現方法は提案時はツール類等の開発環境が十分でないことが想定されるため，新たな表現方法への単なる置き換えでは開発効率の低下を招く恐れがある．

そこで，報告者は開発した形式表現と従来技術との連携を図ることで，上記課題へ対応できると考えた．

本報告では，従来技術として UML を取り上げ，UML と開発した形式表現との連携についての開発結果と得られた考察について述べる．

２章では開発した形式表現の概要について，以前の報告からの差分も含めて説明する．３章では UML との連携方法の検討結果について述べる．４章では検討した連携方法に対応したツールの概要について説明する．５章ではツールを用いて実際に変換した結果について述べる．６章で得られた考察と今後の予定も含め，まとめる．

2. 開発した形式表現の概要

ここでは開発した形式表現の説明として，基本となる構成要素と実際の表現方法と表現を組み合わせてまとめる仕組みである類型化について以前[4]からの差分も含めて述べる．

2.1 形式表現の構成要素

開発した形式表現を用いて表現作業を行う場合，データを中心として表現する．データを中心といってもデータ中心アプローチにおけるデータとは異なり，開発した形

[†] 株式会社トヨタ IT 開発センター
Toyota InfoTechnology Center, Co., Ltd.

ⓒ2011 Information Processing Society of Japan

式表現では「データ」は機能や振る舞い等も含む広い範囲の物事/事物を対象としている．これは対象を表現する場合，何らかの手段により観測された結果として表現される，つまりデータであるということを意図して構成要素の名称として用いている．

開発した形式表現の基本的な構成要素はデータと制約の二つである．それぞれの定義としては辞書的な意味では以下の通りである．

データ：状態・条件などを表す数値・文字・記号

制約：物事の成立に必要な条件や規定

データについては，上記で述べた通り，辞書的な意味よりは広い範囲の定義となっている．制約については，辞書的な定義における物事の変わりに本仕様記述方法におけるデータの成立に必要な条件や規定となる．

これらを用いた表現の考え方としては，表現したい対象をまずデータと捉え，それを説明する形でデータに制約を接続する．その際，説明のために必要なデータを制約に接続することができ，そのデータを参照データと呼ぶ．これにより，定義したデータが他のデータと関係付けられる．

2.2 表現方法

表現のための追加の要素として複製データがあるが，それを含めて開発した形式表現の図形的な表現は以下の通りである．

・データは矩形で表現
・制約は角括弧で表現
・データを説明する制約は説明するデータに対して矢印で接続
・参照データは制約に対して通常の線で接続
・複製データは破線の矩形で表現

複製データは同一のデータを表すため，図形表現として追加した．基本的にはデータと制約のみで表現可能であるが，複製データを導入することで，同じデータへの参照の線を減らすことができるため，記述のための領域を節約し，可読性を上げることができる．

図 2-1 に表現例を示す．「変数」という名前のデータは，抽象的なデータである．「名称」制約を持ち，「x」という名前になる．ここでは，x は複製データとして，どこか別の場所に定義されているものとする（もちろん，近傍にあれば，元となるデータ「x」から参照線を引くこともできる）．従って，x は参照データでもあり複製データでもある．x には「値」制約として，データ「0」が設定されている．

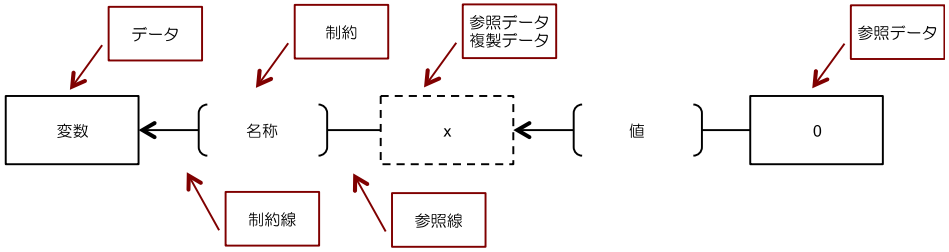

図 2-1　データと制約による表現例

2.3 類型

開発した形式表現では，データと制約の接続パターンを定義できる仕組みを用意した．これを類型と呼ぶ．接続パターンの登録即ち類型化により，記述の容易化や作成者間の表記の統一を図ることが可能となる．

図 2-2 に類型の適用例を示す．図２で示す通り，類型とは，一つないしは複数の制約と無名データ（図２の左側の名称が記載されていないデータのこと）からなるパターンとして定義されるものである．

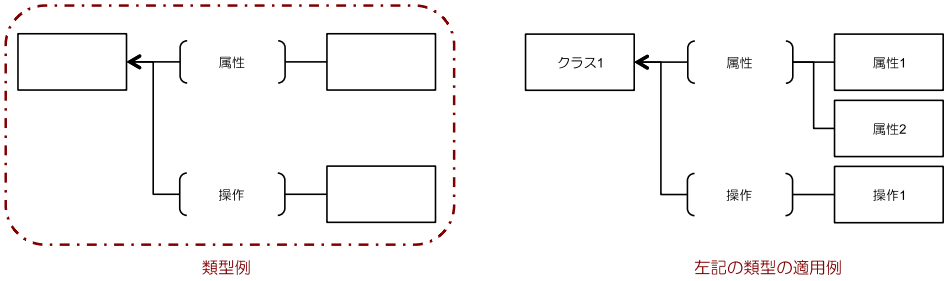

図 2-2　類型の適用例

以前[4]の類型は，まとめたものを新たな構成要素として定義していたが，今回はパターンの登録に留めた．

ⓒ2011 Information Processing Society of Japan

3. UML との連携

3.1 連携方法

UML での表現にあたっては，表現用のツールが複数種存在し，実際の作業では手書きで書くことより，UML 編集ツールを使うことが多い．また，開発した形式表現でも編集ツールを準備していることから，ツールで作成されるデータを変換することで連携を実現する．

このような連携方法により，１章であげた各課題に対して，それぞれ次の効果が期待できる．

・開発者の表現方法の学習コスト

UML 編集ツールで作成されたものが変換されるため，UML 利用者であれば，UML で書いたものがどのように変換されるかがわかり，各自の興味のある題材を UML で具体的に書いてから変換することにより，学習コストの低減が期待できる．

・開発資産の再利用コスト

変換を行うため，開発した形式表現へ資産自体の置き換えについては人手で置き換えることに比べて無視できる程度である．

・不十分な開発環境による開発効率低下

UML 側に十分なツールがあれば，UML で必要な作業を行った後，変換することで効率低下を防止することが可能となる．

3.2 UML ツールとのデータ交換

変換にあたって，まず UML ツールとのデータ交換方法について述べる．UML ツールでは，XMI(XML Metadata Interchange)での保存が可能となっていることが多い．そこで，UML ツールとのデータ交換は XMI 形式を介して行うこととした．

3.2.1 XMI の概要

XMI は OMG が策定した規格であり, Meta-Object Facility (MOF)で表現されるデータを Extensible Markup Language (XML)を使って表現し交換することを可能とする[5].

XMI ファイルは，タグ「 xmi:XMI 」を最上位要素とし，出力ツール情報，モデル情報，独自拡張情報から構成される．

なお，各種 UML ツールで生成される XMI の形式は互換性が乏しいため，異なる UML ツール間で XMI ファイルでのやりとりはできないことが多い．

3.3 変換方法

変換方法の検討としてまず, XMI 形式で表現されたものを XMI 形式を開発した形式表現で表現し，その上で，UML の図としての意味を取れる形で変換するという２段階での検討を行った．

これによる利点は，まず XMI 形式レベルでの変換を行うことにより，UML 以外の仕様記述言語のツールで XMI 形式を利用しているものとも XMI 形式レベルでの連携がそのまま可能となり，連携対象を広げやすくすることを狙っている．

また，XMI レベルでの形式を把握しておくことで，UML の図のレベルでの変換方法の検討が容易になるとも考えた．

3.4 XMI 変換

まず，XMI レベルでの変換について述べる．基本的には，XMI のタグ構造をそのままデータや制約で置き換える形で表している．

XMI レベルでの変換は以下の 4 つのルールに基づいて行う．

(a) タグ名を名称とするデータ（以下,「タグ名データ」と呼ぶ）を作成し，そのデータに対する制約の形で属性，および子要素を出力する．
(b) タグの属性情報は，タグ名データに対する制約「属性」として変換する．
(c) 子要素は，タグ名データに対する制約「構成」として変換する．
(d) 子要素についてもタグごとに，再帰的にこのルールを適用して変換を行う．

このルールに基づいて XMI を形式仕様記述に変換する例を例 1 と図 3-1 を用いて示す．例 1 が変換対象の XMI であり，それを開発した形式表現で変換した結果を図 3-1 に示す．なお，ここではタグ名および各タグの属性名の部分のみを変換しており，各属性の値の部分の変換は省略している．

例 1) 変換対象の XMI

```
<packagedElement name=“クラス”  xmi:type=“uml:class”>
    <ownedAttribute   name=“myAttr… >
        <lowerValue value=“1”/>
    </ownedAttribute>
    <ownedOperation … />
</packagedElement>
```

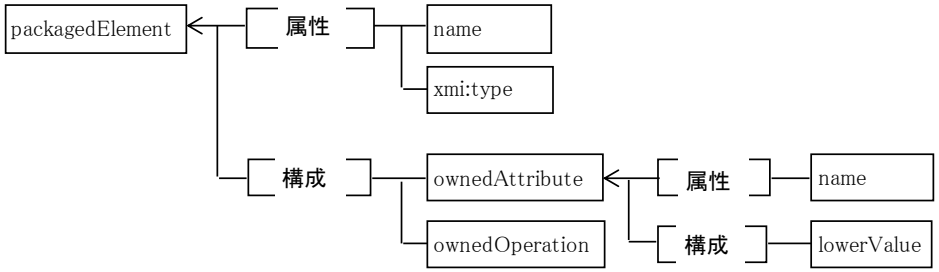

図 3-1　XMI の変換例

ⓒ2011 Information Processing Society of Japan

3.5 モデル図変換

機械的に変換した形式表現から，各データの意味を踏まえた変換を行い，UML としての仕様，設計情報として必要なもののみを抜き出した形式表現に変換することを，モデル図変換と呼ぶ．本節では，この変換を行うための変換ルールの記述方法についてまとめる．

3.5.1 機械的類型パターンの定義

モデル図変換ではまず，UML 図を示す XMI ファイルの機械的変換を行った結果の形式表現に含まれる各種制約または参照データに着目し，その中から仕様，設計情報として意味を持つ要素の抜き出し，類型を用いて定義する（以下，機械的類型パターンと呼ぶ）．UML 図を 1 つの類型で定義するのではなく，クラスシンボルのようなデータ構造を示す機械的類型パターン，依存関係を示す機械的類型パターンのように，複数の機械的類型パターンに分けて定義する．

3.5.2 変換ルールの定義

変換ルールは，クラス定義などデータ構造を変換するためのルールと，関連線のように 2 つのデータの関係情報を変換するためのルールの，大きく分けて 2 通り定義する．

データ構造の変換では，機械的類型パターンに，変換先に該当する類型への変換ルールを追記することでルールを定義する．図 3-2 に変換ルールの記述例を示し，図 3-3 に図 3-2 のルールの適用による変換後の構造を示す類型の定義を示す．

図 3-2 において，制約の［変換］とその参照データである属性および可視属性が追加されたルールの部分であり，その他の部分は機械的類型パターンで定義された類型である．この変換ルールで示される内容は図 3-3 のデータである属性と可視属性である．

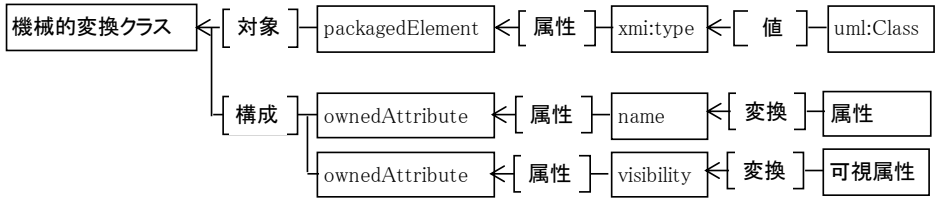

図 3-2 変換ルールの記述例

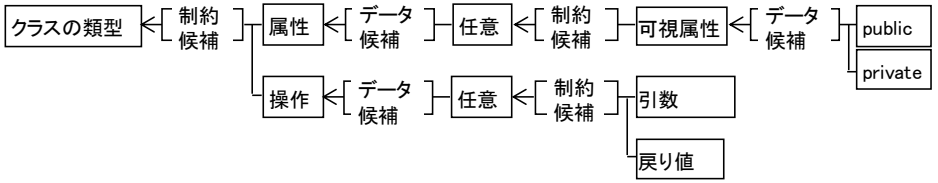

図 3-3 変換後の構造を示す類型の定義

データの関係情報の変換ルールは，UML での関連線をもとに定義する．関連線は UML 図においては 2 つのシンボル要素を接続するために用いられるため，変換ルールにおいても，作成済みの 2 つのデータの関係を，関係種別に応じた制約で結び付ける形への変換を行うルールとして記述する．つまり，関係線の変換は，データ，制約，参照データという 2 つのデータを制約で結ぶという形式仕様記述言語の基本パターンへの変換となる．

このため，変換後の記述パターンは類型で示すような複雑な構造とはならない．代わりに，変換対象がデータ，参照データのどちらになるかを示す形で変換ルールを記述する．図 3-4 に関係を定義する機械的類型パターンと変換例を示す．

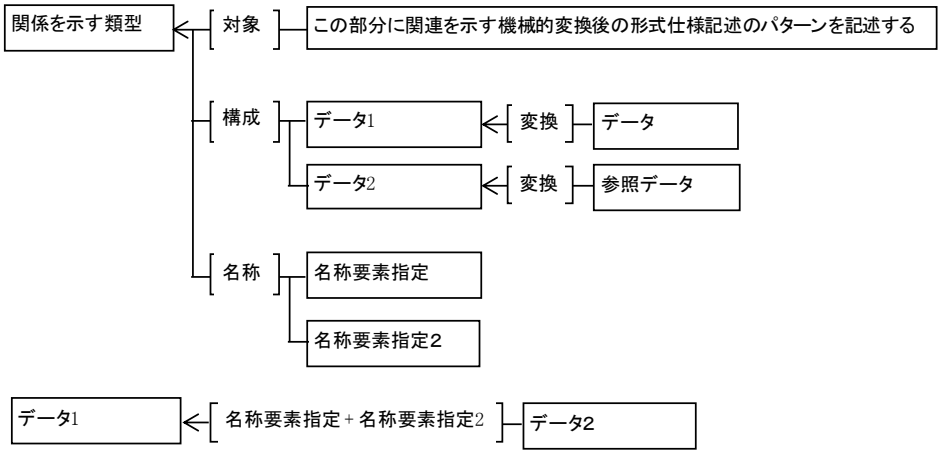

図 3-4 関係を定義する機械的類型パターンと変換例

ⓒ2011 Information Processing Society of Japan

4. 連携機能付き編集ツール

4.1 概要

今回試作した編集ツールにおける以前の報告での編集ツールからの変更点や追加点の主なものを以下に挙げる．

・以前の編集機能が構成要素を自由に配置できるグラフィカルエディタであったのをキーボード主体の編集を可能とするグリッド配置のエディタに修正
・複数ユーザでのプロジェクト等を想定したデータベース機能
・UML 変換機能

なお，１点目の編集方法の変更については，仕様規模が大きくなると，ダイアグラム中心の仕様記述では記述した仕様の理解・維持管理が容易でなくなるとの指摘もあり[6]，表示と操作をコンパクトにするために実施した．

２点目のデータベース機能の追加については，記述量の増大に対処するための手段の一つとして実施した．

次節からは，本報告の主たる目的である UML 変換機能について説明する．

4.2 類型管理機能

UML 変換機能を実現する上で，開発した形式表現が持つ類型の仕組みを活用した．そこで，まず類型の仕組みを説明する．

類型の定義は，データと制約の関係と，データや制約に対する条件の指定を，開発した形式表現を用いて定義を記述する方式を用いる．

類型定義では，データと制約の関係性，制約と参照データの関係性，更に制約や参照データに対する条件の指定について制約を用いて記述する．この制約の総称を「メタ制約」と呼ぶ．開発した形式表現と類型定義の関係を図 4-1 に示す．これを用いて類型定義の書式の基本を定義する．

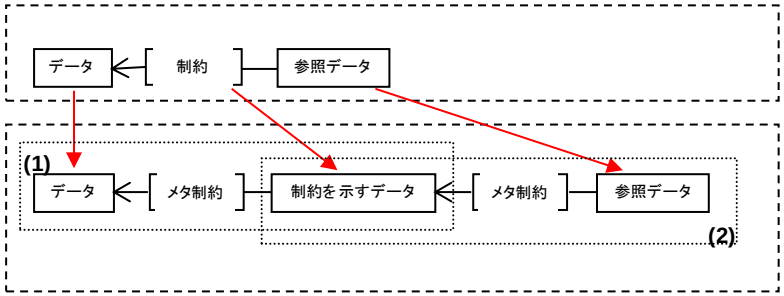

図 4-1 開発した形式表現を用いた類型の定義

データと制約の関係は，類型定義の「メタ制約」を用いて，図 4-1 の(1)の形で定義する．もともとの「データ」が類型定義の「データ」に，もともとの「制約」が類型定義の「制約を示すデータ」に対応する．同様に，制約と参照データの関係も「メタ制約」を用いて，図 4-1 の(2)の形で定義する．もともとの「制約」が類型定義の「制約を示すデータ」に，もともとの「参照データ」が類型定義の「参照データ」に対応する．

そして，この基本構造を繰り返す形で類型を定義する．繰り返す際は「参照データ」が次の繰り返しの「データ」となる．

さらに，「データ」，「制約を示すデータ」，「参照データ」に対して，メタ制約を用いて条件を示すデータを定義できる．これらの定義を図 4-2 に示す．

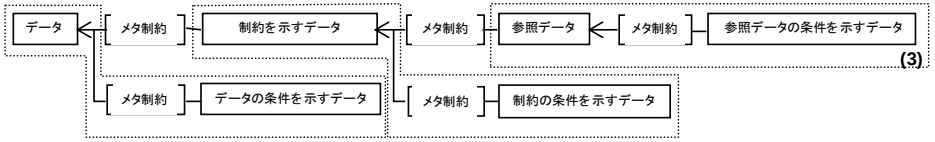

図 4-2 メタ制約による条件定義

図 4-2 の(1)のように．「データ」に対する条件を「メタ制約」と「データの条件を示すデータ」で定義する．同様に，図 4-2 の(2)のように．「制約を示すデータ」に対する条件を「メタ制約」と「制約の条件を示すデータ」で定義する．また，図 4-2 の(3)のように．「参照データ」に対する条件を「メタ制約」と「参照データの条件を示すデータ」で定義する．

このように定義することで，今後類型を拡張する場合にも，新たな位置づけのメタ制約を追加する形で容易に拡張できる．

4.3 UML 変換機能

今回，UML との連携ということで，UML データの変換機能を試作した．3.2 節で述べたように，UML ツールから出力される XMI データはツール依存の部分があるため，今回の変換機能が対象とする UML ツールは Enterprise Architect[7]とした．

機能として試作したのは，３章で述べた XMI 変換とモデル図変換の２つの機能であり，具体的には，次の通りである．

・XMI 変換
　XMI タグ構造をそのまま開発した形式表現に変換
・モデル図変換

ⓒ2011 Information Processing Society of Japan

XMI 変換により UML ツールで作成された XMI ファイルから変換された形式表現データを変換用の類型パターン定義を用いて，元々の UML の図の意味を保つようにした変換

変換操作においては，ユーザは次の変換方法を指定できる．

・XMI 変換のみ
・モデル図変換で一度 XMI 変換を実施した後，UML の図の意味を保つようにした表現に変換
・モデル図変換で XMI 変換を経由せず直接 UML の図の意味を保つようししした表現に変換

なお，モデル図変換においては，今回は UML の図において，クラス図，状態遷移機械図，アクティビティ図の３つの図を対象とした．

これは，ソフトウェアはデータ構造，ふるまい，処理の流れで表現されることが多く，データ構造はクラス図，ふるまいは状態機械図（ステートチャート），処理の流れはアクティビティ図で表すことができ，また，UML の多くの図は上記で集約できるため，今回はこの３つの図を対象とした．

なお，この３つの図は開発した形式表現での表現でも UML の図の意味を持たせた類型を使って表現されたものは UML ツールで読み込める XMI ファイルの出力が可能である．以下に３つの図の変換例を示す．

図 4-3 にクラス図例とそのモデル図変換結果を，図 4-4 に XMI 変換結果を示す．なお，開発した形式表現での表現のデータ部分の右肩の T の字は類型定義であることを示す．

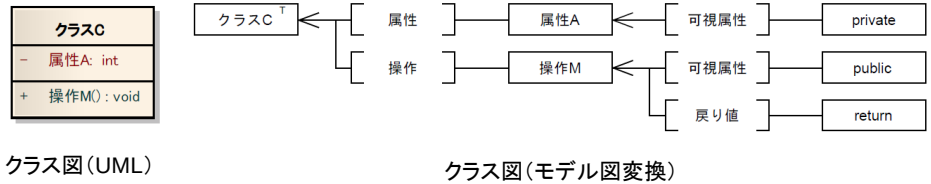

図 4-3 クラス図の変換例（モデル図変換）

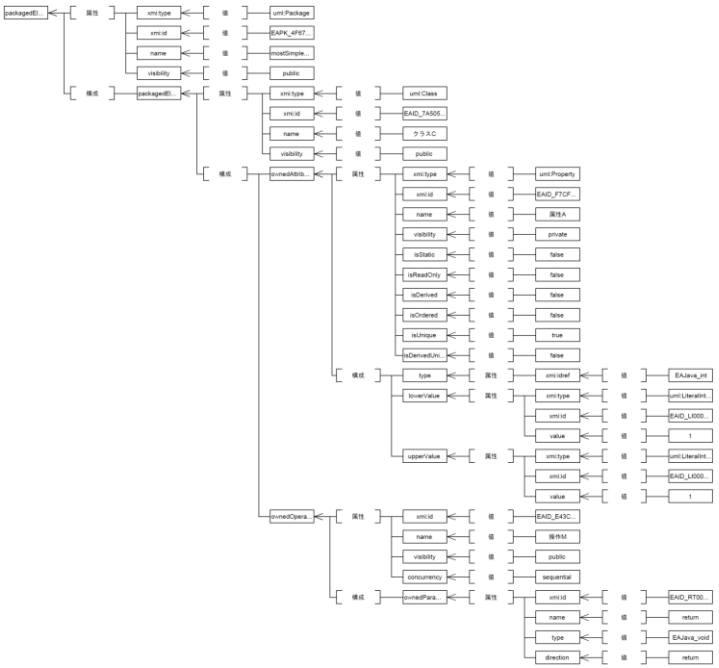

図 4-4 クラス図の変換例（XMI 変換）

図 4-5 に状態機械図例とそのモデル図変換結果を，図 4-6 に XMI 変換結果を示す．

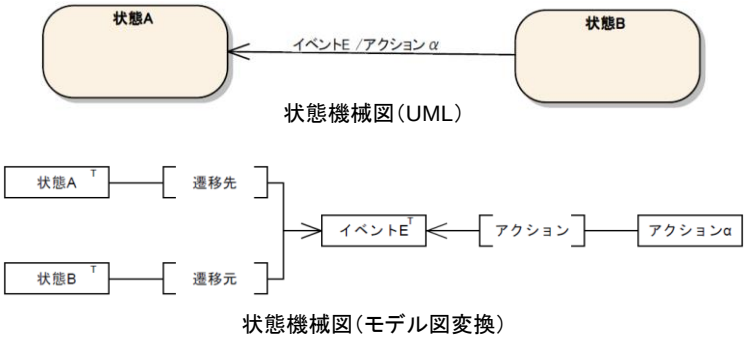

図 4-5 状態機械図の変換例（モデル図変換）

ⓒ2011 Information Processing Society of Japan

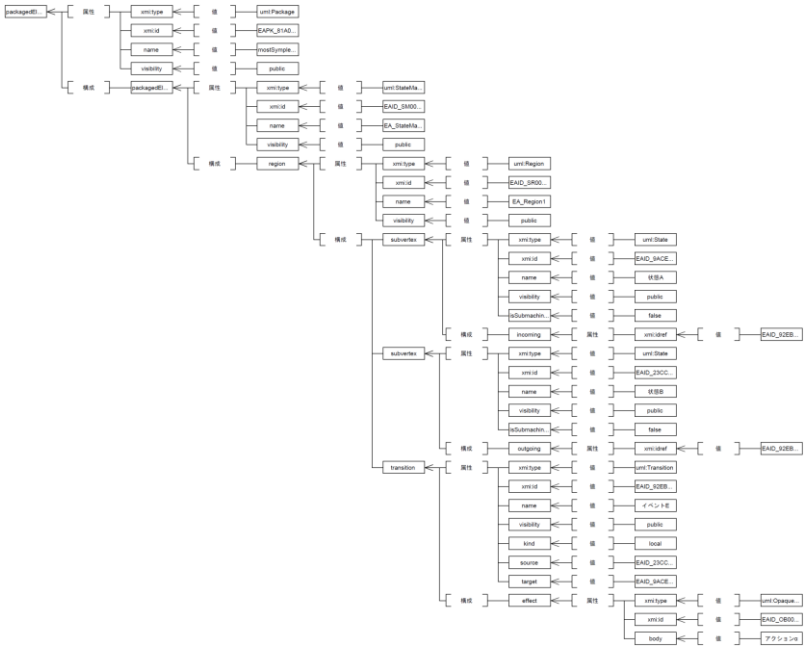

図 4-6 状態機械図の変換例（XMI 変換）

図 4-7 にアクティビティ図例とそのモデル図変換結果を，図 4-8 に XMI 変換結果を示す．

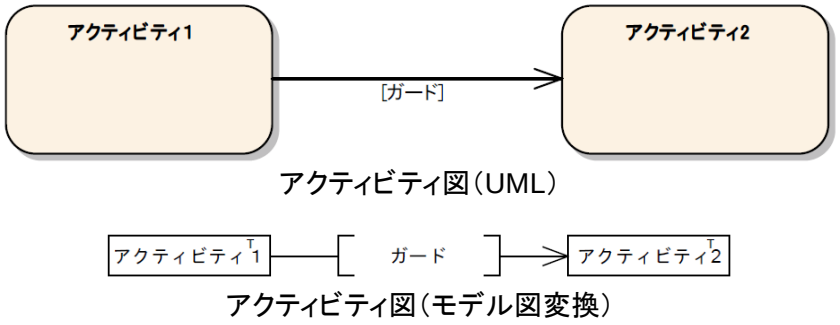

図 4-7 アクティビティ図の変換例（モデル図変換）

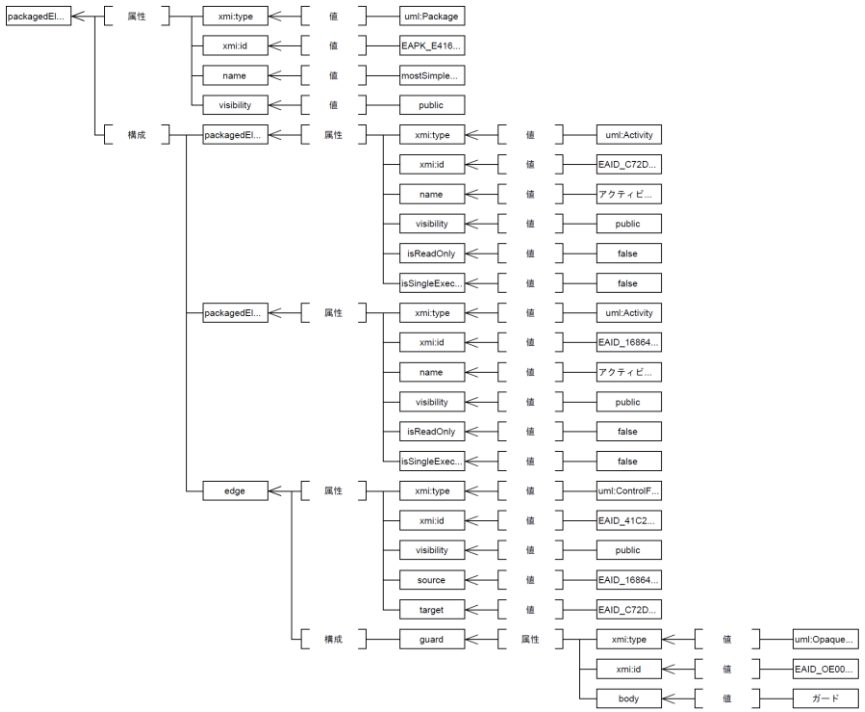

図 4-8 アクティビティ図の変換例（XMI 変換）

5. 連携結果と考察

ここでは，具体的な仕様を UML で表現し，試作した編集ツールに取り込んだ場合と UML への変換を意識し，UML の図の類型を使用して開発した形式表現で表現した結果をもとに考察を行う．

5.1 題材

UML および開発した形式表現で表現する題材としては，画面表示を伴うアプリケーションの概要仕様を用いた．これは，提供する機能に対して，処理する内容と表示される画面およびその遷移が定義されている．

ⓒ2011 Information Processing Society of Japan

5.2 UML による仕様表現

UML での表現では, 今回の変換機能で試作したクラス図と状態機械図とアクティビティ図のみで表現を行った.

結果としては，処理等の表現については問題なかったが，処理完了後の画面状態等の表現は UML が持つ記法では適切に表現することが難しかった．例えば，ある処理の遷移で表示する内容が題材の資料には記載されているが，このような情報をモデルの情報として表現する方法が用意されていないためである.

UML により表現されたものを XMI ファイル経由で編集ツールに取り込んだが，取り込んだ後に, 画面状態等の情報をモデル情報と同じレベルで付加することができた.

これは，開発した形式表現の表現の柔軟性を示している.

5.3 開発した形式表現による仕様表現

同じ題材を開発した形式表現で表現し，XMI ファイルに出力し，UML ツールで取り込むことを行った.

今回の表現作業にあたっては, UML への変換を想定し，クラス図等の類型を用いて表現を行った.

結果としては，変換のための類型で表現された部分を変換したものは UML ツールでの取り込みが可能であることが確認できた．しかしながら，変換のために UML 変換向けの類型を使用することが必要であり，そのために余分な情報の入力が必要だった．これは，今回は UML への変換が目的だったため，題材の仕様とは別に UML 変換のための情報が必要であることを示している.

6. まとめ

6.1 得られた考察

今回，開発した形式表現の実際の開発への適用能力向上のため，UML との連携機能について検討を行い，機能の試作を行った．また，例題を用いて，UML での表現との連携についても確かめた.

また，今回の活動を通じ，次のことが得られた.

・XMI 変換によるツール間の連携

UML ツールでも XMI で出力された内容は互換性があることが少ない．そのため，XMI レベルで各ツールに沿った類型を定義することで，UML ツール間での変換機能が実現できる可能性が見出せた．また, XMI を扱える DSL との連携の可能性もある.

・開発した形式表現の表現能力の高さ

題材の表現の結果，仕様の表現領域が UML よりも広いことが確認でき，開発した形式表現の表現能力の高さが確かめられた.

・変換における情報の付加

今回，開発した形式表現を UML に変換することを行ったが，その作業から表現したい対象の仕様表現とは別に，表現結果を別の表現に変換するにはそのための情報も表現に組み入れる必要があることがわかった.

6.2 今後の予定

今回，UML への変換を行ったが，これをプログラミング言語に置き換えることでコード生成を想定することができる．これにより，仕様からコードへの生成の自動化が見込め，生産性の向上に寄与することが可能となる.

さらにこれを他のツールまで広げることで仕様表現からコード生成，またはそれ以外の工程で使われるツールとの連携が可能になると思われる.

モデリングを用いた開発においては今後はマシン支援が重要との指摘もあるが[8], 各種ツールに対して開発した形式表現を介して連携させることでマシン支援の向上を図ることができると考える.

よって，今後の予定としては，開発した形式表現を一種の中間言語として考え，仕様表現以外のツールとの連携を実現し，更なる開発効率向上に向けた検討を行う予定である.

謝辞 ツールの試作や具体例表現作業等で協力いただいた株式会社ニルソフトウェアのみなさまに感謝します.

参考文献

[1] OMG UML, http://www.uml.org/
[2] OMG SysML, http://www.omgsysml.org/
[3] OMG MARTE, http://www.omg.org/omgmarte/
[4]阿部睦 : 統一的表現に向けた仕様記述方法, 情報処理学会研究報告, Vol.2011-SE-171 No.24
[5] OMG XML Metadata Interchange, http://www.omg.org/spec/XMI/
[6] 田中明，高橋修：ビューポイント DSL を用いたシステム仕様記述に関する考察, 情報処理学会研究報告，Vol.2010-SE-168 No.13 Vol.2010-EMB-17 No.13(2010)
[7] Enterprise Architect, http://www.sparxsystems.jp/ea.htm
[8] 岸知二：スケーラブルなモデリング技法に関する考察，情報処理学会研究報告, Vol.2011-SE-173 No.10

ⓒ2011 Information Processing Society of Japan