OCL から JML への DSL を用いた変換ツールの試作型の実装

花田　健太郎[†1]　岡　野　浩　三[†1]　楠　本　真　二[†1]

研究グループでは，OCL から JML への変換手法に関する研究を行っている．本稿では，DSL 開発のフレームワークである Xtext を用いてツールの試作型を実装した．具体的には，まず OCL の DSL を構築し，次に DSL から JML へのマッピングを定義する方法で行っている．

Implementation of a prototype translation tool from OCL into JML by using DSL

Kentaro Hanada,[†1] Kozo Okano[†1] and Shinji Kusumoto[†1]

Our research group has been studying a translation method from OCL to JML. In this report, we present a prototype translation tool from OCL to JML by using the following manner. First, we define a DSL of OCL using Xtext. Next, we describe translation rules from the DSL to JML. The syntax and rules are used to translation in a framework provided by Xtext which is a plug-in for Eclipse.

1. はじめに

研究グループでは，OCL から JML への変換に関する多くの既存研究[1)2)] において非対応だった iterate 演算に対応した変換手法の提案・実装を行っている．[3)] しかし，既存ツールは MDA の主流であるモデル変換的な実装ではなく，抽象構文木を介したマッピングで実装されている．今後 OCL から JML 以外の複数の言語への変換の実現を目標としているが，既存手法ではそれぞれの変換を独立に実装する形になるため実装の再利用性が低く，複数言語への変換の実装には向かない．また，既存の実装は変換の達成を主な目的としており，ツールそのもののユーザビリティは低い．

本稿では，Xtext を用いて DSL を構築し，その DSL からテキストへの変換を定義する方法での実装を試みた．Xtext では DSL と変換部分が独立しており，構築した DSL の再利用性が高いことが利点として挙げられる．また，構築した DSL に対応した高機能なエディタを生成することができるため，ツールそのもののユーザビリティの向上も見込めると考えられる．

2. 準　　備

本章では背景知識として OCL，JML，DSL，Xtext について順に説明する．

OCL は，UML モデルに対してさらに詳細な性質記述を行うために設計された言語であり，OMG によって標準化されている．

JML は，Java のメソッドやオブジェクトに対して制約を記述する言語である．記述においては Java の文法を踏襲し，初心者でも記述しやすい特徴を持つ．また，記述は Java コメント中に記述できるため，プログラムの実装，コンパイルや実行に影響がない．

DSL は，ドメインに特化した言語を設計利用する方式である．DSL の例としては Web に特化した HTML，RDB に特化した SQL などが挙げられる．

Xtext は，DSL の構築や，DSL に従ったモデルからテキストへの変換の定義などをサポートする DSL 開発のフレームワークである．DSL を定義することで、コード補完機能やエラー検出機能などを備えた DSL 用のエディタを生成することができる．

3. 実 装 方 針

Xtext を用いて OCL の DSL 構築し，構築した DSL から JML へのマッピングを定義することで変換を実現する．この実現方法には従来の実装に比べて以下の利点が挙げられる．

- 構築した DSL は，テキストへの変換部分の実装から独立しているため再利用性が高い．
- DSL を構築することで，コード補完や文法エラーの検出を行えるユーザビリティの高いエディタが得られる．

†1 大阪大学
Osaka University

4. 実　　装

本章ではツールの実装についての説明を行う．

4.1 DSL 構文の構築

OCL を付加できる UML クラス図の DSL を構築した．UML の部分に関しては，Xtext で既存のものが存在するので流用し，OCL 部分を追加する形で DSL の構築を行った．OCL 部分に関しては，メソッド名や引数の数，戻り値の型などの場合分けを追加した EBNF を元に DSL を構築した．変換ルールの定義は DSL に依存するので，DSL の段階で詳細な場合分けを行うことで意味的な解析にかかるコストが削減され，DSL を再利用することの有用性が高まると考えられる．また，生成されるエディタのエラー検出やコンテンツアシストは定義した DSL に依存するので，DSL の場合分けを厳密に定義するほどコンテンツアシスト機能などが充実したエディタが得られる．これらのことから，メソッド名なども考慮した DSL 構文の構築は，ユーザビリティの観点や実装の再利用性の観点から有用であると考えられる．

4.2 OCL から JML への変換ルールの定義

OCL-JML 間の変換は既存研究とほぼ同様に定義している[3]．exists 演算や forall 演算のようなコレクションループは表 1 のように iterate 演算に置き換えることが可能なため，既存研究では iterate 演算以外のコレクションループも iterate 演算に置き換え，その OCL 式と意味的に等価なメソッドを生成して，その値を JML 側から参照する方法で変換していた．しかし，この変換方法ではコードの可読性が低下するといった問題点が挙げられていたため，本稿の実装では JML にメソッドが存在している exists 演算や forall 演算などは直接的にマッピングを行っている．

表 1　Collection-Iterate 対応表の一部

c_1->exists($a_1 \mid a_2$)	=	c_1->iterate(a_1; res : Boolean = false | res or a_2)
c_1->forAll($a_1 \mid a_2$)	=	c_1->iterate(a_1; res : Boolean = true | res and a_2)

また，DSL に従ったモデルの例を図 1 に，モデルからの変換結果を図 2 に例として示す．

```
entity Sample {
    inv : sampleVariable >= 0
    sampleVariable : Integer
}
```

図 1　DSL に従ったモデル

```
package ;
public class Sample {
/*@
invariant ((sampleVariable)>=0);
@*/
    private Integer sampleVariable;

    public Integer getSampleVariable() {
        return sampleVariable;
    }

    public void setSampleVariable(Integer sampleVariable) {
        this.sampleVariable = sampleVariable;
    }
}
```

図 2　モデルから変換された出力結果

4.3 既存研究との実装規模の比較

既存研究では，ANTLR に OCL の EBNF を与えることで構文解析器を得ていた．しかし，ANTLR から得られた構文解析器の規模は数万行で可読性，拡張性の低いものだった．また，構文解析器に手動で変更を加えた後に OCL 構文に変更を加える必要が出た場合，手動による変更が反映できないといった問題が存在していた．Xtext を用いた場合は，構築した DSL は他のコード等に依存しないので，DSL の変更，拡張も既存研究に比べ柔軟に行える．

OCL から JML への変換部分の実装は既存のものが約 5000 行に対して，Xtext を用いた手法は約 1700 行とコードの規模は小さい．また，コードは DSL に沿って記述されており，可読性も既存研究に比べて高いと考えられる．

5. あ と が き

本稿では，Xtext を用いた OCL から JML への変換ツールの試作型の実装方法を示した．今後の予定としては，ツールを完成させてツールの評価実験を行うこと，JML 以外の他の言語への変換を今回定義した DSL を再利用して行うこと，OCL-JML 間での相互変換の実現などが挙げられる．

参　考　文　献

1) Hamie, A.: Translating the Object Constraint Language into the Modeling Language, *In Proc. of the 2004 ACM symposium on Applied computing*, pp.1531–1535 (2004).
2) Rodion, M. and Alessandra, R.: Implementing an OCL to JML translation tool, Vol.106, No.426, pp.13–17 (2006).
3) 宮澤清介，花田健太郎，岡野浩三，楠本真二：OCL から JML への変換ツールにおける対応クラスの拡張と教務システムに対する適用実験，信学技報，Vol.110, No.458, pp.115–120 (2011).