B-016

UML/OCL アサーションのサーバサイド実装用ツールの開発

Server-Side Implementation Tool for UML/OCL Assertion

田中 聖一† 小林 洋†
Sei Tanaka Hiromi Kobayashi

1. はじめに

最近のソフトウェア開発においては，ソフトウェアの振る舞いの安全性の観点から，表明（assertion）の記述が重視されるようになって来ている．表明を厳密に表すには論理仕様が適しており，UML では，OCL（Object Constraint Language）が制定されている．これを業務系アプリケーションに実装する場合，クライアント側プログラムでの実装が一般的である．しかし分散システムの安全性の観点からは，サーバ側での実装も考慮する必要があると考えられる．我々は前研究において OCL から SQL のトリガ（trigger）への変換方式を提案し，変換ツールの開発も行ったが，このツールで扱える表明のパターンは限られており，ツールの操作性も良いものではなかった．そこで，現在，扱える表明のパターンの種類を増やすことと，操作性の向上を目指し，ツールを再開発中である．本稿は，その途中結果の報告である．

2. 表明と Trigger について

ソフトウェア仕様記述で用いる表明は，事前条件（precondition），事後条件（postcondition），および不変条件（invariant）に分類される．図式記述では表明の記述が困難であるために，図式記述中心の UML でも一階述語論理に基づく OCL[1]が制定されている．

表明を通常のようにクライアント側で実装する場合の研究はいくつか行われており[2-5]，Java コードに OCL から変換した JML（Java Modeling language）で注釈を付け，それを検証に用いるという方法や，Java コードから表明を自動生成する研究などが知られている．表明をサーバ側で実装しようとした場合，SQL に ASSERTION 文が制定されているが未だ商用データベースでは用いることができず，単純なものはチェック（check）で可能であるが，一般的にはトリガを使用せざるを得ない．トリガの記述は製品によって多少異なるが，本研究では PostgreSQL を用いることにした．トリガは，ストアドプロシージャの一種で，データベースに対して何らかの操作が行われた場合に，自動的に実行される処理のことである．トリガでは処理のタイミングを指定する．PostgreSQL では，トリガとその処理をあらわす関数がセットになっている．なお，トリガでは，一般的なプログラミング言語同様，事前条件と事後条件はそのまま実装可能であるが，不変条件については，事前条件と事後条件の両者により擬似的に実装せざるを得ない．

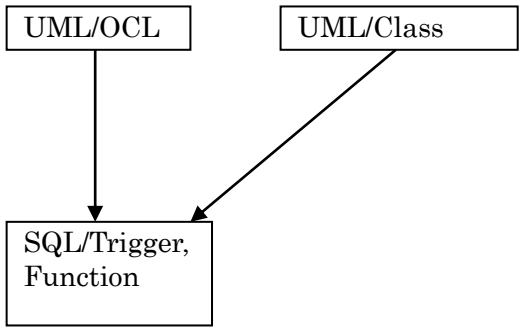

図 1 変換のマッピング

```
context class_name
assert_type：expression

where
assert_type = pre | post | inv

expression =
  【expr_type1】
class_name.attri_name  operator1  value1
|【expr_type2】
class_name -> size() operator1 value1
|【expr_type3 】
class_name-> select(class_name.attri_name
                operator2 value2) -> size() operator1 value1
```

図 2 OCL 記述の枠組み

3. 変換方式の概要

変換ツールの作成に先立ち，まず，変換方式の形式化を行った．変換は，図 1 のようにクラス図に OCL でコメントとして記述された表明からトリガへのマッピングにより行われる．そのため，入力側の OCL とクラス図および出力側のトリガと関数についてテンプレートを作成し，それに基づきマッピングを行った．図 2 に OCL のテンプレートを，図 3 に PostgreSQL のトリガと関数のテンプレートを示す．

本研究では，OCL の制約式にコレクション演算を用いている．コレクション演算とは，クラス図のオブジェクトから生成されたインスタンスの集合を対象としている演算であり，その中から size 演算と select 演算を用いている．size 演算は，対象とするクラスまたは属性のインスタンスの合計数を表す．select 演算は，対象とするクラスまたは属性の検索条件式が真となるインスタンスの集合を表す．

†東海大学大学院工学研究科 Tokai University

```
CREATE TRIGGER trigger_name
manipulation_type
trigger_type
ON table_name
FOR EACH ROW
EXECUTE PROCEDURE
function_name();

where
  manipulation_type =
                DELETE | UPDATE | INSERT

  trigger_type = BEFORE | AFTER
```

```
CREATE FUNCTION  function_name()
 RETURNS TRIGGER
 LANGUAGE PLPGSQL AS'

DECLARE
  count  int;
  var_funcution var_type;
   cursor_name CURSOR
   FOR SELECT key FROM table_name
   WHERE variable1 operator_cur numeric1

BEGIN
   OPEN cursor_name
   count = 0;
  LOOP
     FETCH cursor_name
     INTO var_function
     EXIT WHEN var_function IS NULL;
     count = count + 1;
  end_loop1
  IF variable2 operator_if  numeric2
  ELSE THEN NEW = NULL;
 end_loop2
  CLOSE cursor_name
  RETURN NEW;
END ' ;
```

図 3　トリガ記述と関数記述の枠組み

4. ツールの概要

前研究[6]で開発したツールでは，入力がファイルとコマンドラインに別れており扱いづらかったため，今回は操作性の向上を図るために作成し直した．今回開発したツールでは，入力は全て GUI 画面から行えるようにした．図 4 にツールの入力画面を示す．この画面からテキスト形式のファイルを選択後，読込ボタンを押すことにより出力タブに結果がトリガと関数に分かれて表示され，保存も可能である．

5. 表明のパターンの検討

前研究では，プロトタイプとしてのツールの開発を優先したということもあり，対象とするアプリケーションのモデルは履修登録システムを基にした単純なもので，表明パターンについても 3 種類を実装したのみであった．今回は，モデルについては，もう少し実際的な規模に拡張を行った．

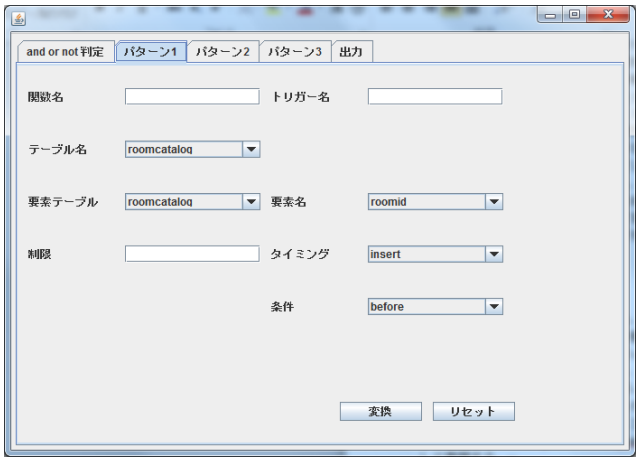

図 4 ツールの画面

表明パターンについては，モデルを検討した結果，今回新たにユニークと排他という二つパターンを見出すことが出来た．OCL では，各々isUnique と exclude と記述される．前者は主キー等の重複チェック用，後者は二重登録の防止や条件に合わないものの登録防止用である．

6. おわりに

OCL で書かれた表明を，サーバ側で PostgreSQL のトリガに変換するツールの開発を行っている．これには変換ツールの開発と表明のパターン化という二つの課題がある．表明のパターン化については，履修登録システムを対象に実施している．表明のパターン化については，対象領域によるところもあると考えられるが，業務系システムの一つの典型である履修登録システムでの研究は，販売管理等の他の分野でも適用可能な部分があるものと考えられる．

参考文献

[1] J. ヴァルメル，A. クレッペ，竹村司訳：UML/MDA のためのオブジェクト制約言語 OCL，エスアイビ ー・アクセス (2004).
[2] A. Hamie: Translating the Object Constraint Language into the Java Modeling Language, Proc. 2004 ACM symposium on applied computing (SAC'2004), 99.1531-1535 (2004).
[3] R. Moiseev and A. Russo: Implementing OCL to JML translation tool,信学技報,SS2006-58,pp.13-17(2006).
[4] M. D. Ernst, J. Cockrell, W. G. Griswold, and D. Notkin:Dynamically discovering likely program invariants to support program evolution, IEEE Trans. Software Engineering, Vol.27, No.2, pp.99-123 (2001).
[5] 宮本敬三，岡野浩三，楠本真二，Java に対するループインバリアントを含む Daikon 生成アサーションの妥当性評価，信学論 D，Vol. J91-D，No.11，pp. 2721-2723 (2008).
[6]黒澤慎太郎，小林洋：表明の UML/OCL から SQL/Trigger への変換，FIT2009，第 2 分冊，pp.219-223(2009.9).