5N-09

トレーサビリティ確保のための Xtend を用いた UML モデルコンパイラ

松隈　暖†　　松浦　佐江子‡

芝浦工業大学　システム理工学部　電子情報システム学科†‡

1. はじめに

トレーサビリティとは開発工程における複数の成果物の関連を定義するものであり，Gotel 等は，「要求トレーサビリティとは，要求のライフサイクルを記述し，前方にも後方にも探索できる能力である」と述べている[1]．トレーサビリティを確保することで要求が設計段階を経て実装されていることが確認でき，漏れなく要求を実装することが可能となり，ソフトウェアの品質を保障することができる．

モデル駆動開発はドメイン依存しないモデルから，ドメインに依存したモデルへ変換するモデルコンパイラの定義により，トレーサビリティ確保に貢献すると考えられている．しかし，モデル駆動開発には，モデルの定義・変換方法とその品質に関する問題がある．本研究では，これらの問題を解決して，トレーサビリティ確保のためのモデルコンパイラを提案する．

2. モデル駆動開発の問題点

モデル駆動開発では各段階のモデルを変換して最終的なコードを生成するため，コードの品質がモデルの品質に依存することと，トレーサビリティを確保するには各成果物の間で要求分析・設計・実装の関係が明確になるように変換方法を定義する必要があるという問題がある．前者に対しては，要求分析段階のモデル品質を向上することが重要である．本研究では，UML とモデルを補完するための OCL（Object Constraint Language）[2]を用いて，要求分析，設計段階のモデルを定義し，実装への変換方法を定義することで，後者の問題を解決する．

各段階の定義の依存箇所を限定することで，定義間の関係が複雑にならず，実装に必要な要素がどこで定義されたか明確になり，トレーサビリティ確保が可能となる．

3. UML モデルコンパイラ

UML を用いたソフトウェア開発では要求分析モデルとしてソフトウェアがもつ複数の機能をユースケース図，ユースケースごとの振舞いをアクティビティ図，必要なデータをクラス図で定義する[3]．設計モデルでは，シーケンス図を用いて要求分析モデルで定義した振舞いをクラスに割り当て，振舞い仕様を定義し，要求分析モデルの入出力の振舞いをまとめたバウンダリクラスを定義することでクラス図を完成する．このように，要求分析モデルと設計モデルを関連付ける．設計モデルからコードを自動生成することで要求から実装までのトレーサビリティを確保する．

3.1. 要求分析モデルと設計モデル

要求分析・設計モデルでは，特定のプログラミング言語に依存しないようにクラス定義に OCL の基本型，コレクション型を用いる．基本型には真偽値を表す Boolean，自然数を表す Integer，実数を表す Real，文字列を表す String 型がある．コレクション型には性質として重複の有無，順序関係があり，この組み合わせで 4 種類ある．重複あり，順序関係ありの Sequence，重複なし，順序関係ありの OrderedSet，重複なし，順序関係なしの Set，重複あり，順序関係なしの Bag である．

要求分析モデルではユースケースには事前条件，事後条件，クラスには不変条件として要求が関連付いており，これらの条件を満たすようなユースケースの振舞いをアクティビティ図で記述する．

設計モデルではユースケースの振舞いを満たすようにクラスの操作と関連付いたメッセージを時系列に並べることで，要求を満たすユースケースとクラスの操作のアルゴリズムを作成することができる．

3.2. 実装への変換の条件

設計モデルから実装への変換をするためには，設計モデルの記述方法にいくつかの条件がある．

3.2.1. クラス図の条件

クラス図を実装へ変換するには，関連の誘導可能性により，属性の設定が決定されるので，クラスの関連には少なくとも一方に誘導可能性を持たせる必要がある．また，バウンダリクラスへの関係は，その意図を明確するために使用依存を使う．

3.2.2. シーケンス図の条件

シーケンス図を実装へ変換するにはシーケンス図の要素がソフトウェアの構造であるクラスと関連付いており，かつ，演算を行うメッセージにはその仕様が記述されている必要がある．そのため，シーケンス図の記述に条件が 5 つある．1 つ目の条件はメッセージ列とクラスの要素を関連付けるために，メッセージには対応するクラスの操作を割り振る．2 つ目の条件は記述されたアルゴルズムがどのインスタンス間で行うかを明確にするためにライフラインにはクラスと名前を割り振る．3 つ目の条件はシーケンス図に記述されたアルゴリズム全体を操作としてまとめ，クラスの構造へ入れるために，シーケンス図の最初のメッセージにまとめる操作を割り振る．4 つ目の条件はメソッドの呼び出し系列以外の算術演算，論理演算，集合演算のアルゴリズムを記述するために，これらの演算を行うメッセージ対して，その仕様を OCL 式で記述したノートを関連付ける必要がある．仕様をクラスやシーケンス図上で定義されている用語を使い OCL 式で記述することで，演算を実装に依存することなく設計することができる．最後に 5 つ目の条件は複合フラグメントのガードを明確にするために，ガードは OCL 式で記述する必要がある．

3.2.3. ユースケース図の条件

ユースケース図の条件は複数のユースケースの中からソフトウェアの起点となるユースケースを決定するために，起点となるユースケースのステレオタイプに main と

Traceability Assurance of Software Product by UML Model compiler using Xtend

†Dan MATSUGUMA ‡Saeko MATSUURA

†‡Department of Electronic Information System, Collage of System Engineering and Science, Shibaura Institute of Technology

記述する必要がある．

3.3. 変換方法

設計モデルは言語に依存する形式で書かれていないため，実装するJavaの演算との変換規則が複雑になり，定義することが難しい問題がある．

この問題に対して，本研究では中間言語としてプログラミング言語を対象ドメインとするDSL（Domain Specific Language）であるXtend[4]を用いることで解決する．Xtendの特徴は，拡張メソッドが使え，ラムダ式を簡素に記述できるため，Javaより簡素な構文を持っていることと，独自のコンパイラによりJavaへの変換が可能であることが挙げられる．中間言語としての役割は，OCL式に対応する演算を行うメソッドをライブラリーとして作成し，拡張メソッドとして使うことで，OCL型が持つメソッドのように記述することができる．例えば，設計モデルにrectanglesというRectangleが要素のOrderedSet型の属性を持ったクラスに対して，rectanglesの要素番号を指定して要素を取り出す下記のようなシグネチャーの操作を作成した場合を考える．

+ 長方形を取得する(number : Integer) : Rectangle

シーケンス図上のメッセージに対して，この操作の仕様はOrderedSet型の演算を用いて下記ように記述できる．

self. rectangles->at(長方形番号)

at演算は順序があるコレクションに対して行うことができ，指定された場所の要素を意味する．長方形番号は取得したい要素の場所であり，シーケンス図上のInteger型の変数名である．この仕様を作成したライブラリーを使用して変換すると下記の式がXtendのメソッドのボディに記述される．

this. rectangles. at(number)

この式はOCLのat演算をライブラリーとして実装したatメソッドを拡張メソッドとして記述している．

これにより，OCL式で書かれた演算をJavaに依存することなく実装することができる．そのため，設計モデルを実装と切り離して作成することができるようになる．

また，Javaに依存する変換規則はOCL型の変換規則のみである．Bagの性質を満たすJavaでの実装はないが，要求分析段階で集合に対する要求を満たすOCLコレクション型を選び，設計段階で性質を満たすように設計することで実装することができる．

3.4. ツールの実装と機能

ツールはモデリングツールである astah[5]のプラグインとして実装した。ツールの機能は以下のとおりである。

- 変換できないモデルの修正箇所の表示
- 設計モデルからのXtendのコードの生成

3.4.1. 修正箇所表示

クラス図，シーケンス図が実装へ変換する条件を満たしていない場合や，OCL式に誤りがある場合はその箇所を修正箇所として問題点の内容とともに，表形式で別画面に表示する．また，シーケンス図を変換する過程で生成されていない可能性があるインスタンスを使用した場合は，警告としてそのインスタンス名を表示する．

3.4.2. 設計モデルからXtendのコード生成

クラス図をクラス構造，シーケンス図をメソッドのアルゴリズムに変換する．

クラス図の変換規則はastahのクラス図からJavaのスケルトンコードを生成する機能を基にクラス図からXtend のクラス構造へ変換する変換規則を作成した．astah のスケルトンコード生成機能と異なる点として，関連と使用依存は誘導可能性がある場合のみフィールドとして出力する点である．

シーケンス図は表1の変換規則で変換する．

表1：シーケンス図の要素の変換規則

シーケンス図の要素	Xtend
メッセージ	メソッドの呼び出し
クリエイトメッセージ	インスタンスの生成
ライフライン	インスタンス
複合フラグメント	分岐やループ
相互作用の利用	メソッドの呼び出し

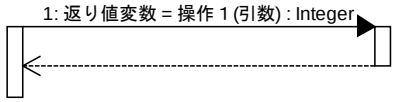

図1：メッセージの例

例として図 1 のようなメッセージは以下の形に変換する．

返り値変数名=ライフライン名. 操作名(引数);

4. 適用実験と考察

ツールを実際に適用するモデルとして本学科で行われるプログラミング課題「長方形エディタ」のモデルを用意した．長方形エディタは利用者が長方形を作成，移動，削除などができることが，目標である．作成したモデルはユースケースが6個，クラスの数は5個である．なお，バウンダリクラスは API が提供されており，定型的に記述することができるため，クラス図に記述された構造と同じJavaのクラスを用意することで実装した．

ユースケース「長方形を作成する」のシーケンス図はメッセージ数 14 個，複合フラグメントが 3 個，OCL での仕様の記述が 5 個である．このシーケンス図を変換した結果，生成された Xtend のコード行数は 33 行，Java のコード行数は 37 行になった．

生成されたコードに対してテストした結果，基本フローのテストは成功したが，例外フローのテストは失敗した．原因はクラスの不変条件を満たすようにするアルゴリズムがシーケンス図に記述されていなかったためである．しかし，シーケンス図に記述されていたアルゴリズムは実装されていたことが確認でき，設計モデルと実装のトレーサビリティが確保されていることを確認できた．

5. 今後の課題

現状すべてのモデル要素には対応しておらず，ループや分岐以外の複合フラグメントなどにも対応できるようすることや，OCL型のInteger型とReal型の掛け算が変換出来ていないなどのOCL式を実装へ変換する箇所に課題が残っている．

参考文献

[1]Gotel,O.and Finkelstein,A.:An Analysis of the Requirements Traceability Problem,Proceedings of the 1st International Conference on Requirements Engineering, pp.94-101(1994)

[2]ヨシュ・ヴァルメル，アーネク・クレッペ，UML/MDAのためのオブジェクト制約言語OCL第2版，竹村司訳，エスアイビー・アクセス,2004

[3]松浦，ソフトウェア設計論 ―役に立つ UML モデリングへ向けて―，コロナ社，2016

[4]Xtend,http://www.eclipse.org/xtend/(2019/1/7 参照)

[5]astah, http://astah.net/ (2019/1/7 参照)