5J-06

要求分析と設計間のトレーサビリティ確保のための UML 変換ツール

吉野　魁人†　　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

要求分析では，システムの境界におけるユーザや他システムとのインタラクションに着目して，システムのユースケースを分析する．設計では，各ユースケースをクラスの振舞い系列として実現するためにシーケンス図を用いてクラスのメソッドを定義する．要求分析で定義した振舞いフローを踏襲して，クラス定義に基づき，各クラスのメソッドを定義することにより，要求分析と設計間のトレーサビリティが確保できる．しかし，人手による解釈ではモデルの観点が異なるため設計の段階に漏れや誤りが発生する可能性がある．本稿では，UML（Unified Modeling Language）を用いて，インタラクションを実現する振舞いフローを定義したアクティビティ図とデータを定義したクラス図から，設計の起点となるシーケンス図を生成することにより，トレーサビリティ確保を支援する手法を提案する．

2. 要求分析・設計段階モデルのトレーサビリティ

2.1. アクティビティ図によるユースケース記述

ユースケースは一般にテンプレートに従った自然言語記述で定義されることが多い．われわれはユースケースの妥当性と実現可能性を検証するために，要求分析において，システムのユースケースをアクティビティ図とクラス図を用いて分析を行っている[1.2]．ユースケースにおける振舞いを「アクター」「インタラクション」「システム」のパーティションに分け，アクターとシステムのやり取りを主体と役割を明確にしたアクション系列として定義する．一方，システムにおけるデータはクラスとその属性として整理できる．これらが見えてくると，アクションとデータの関係も見えてくるので，オブジェクトノードとしてアクティビティ図に記述することができる．

2.2. シーケンス図

シーケンス図ではユースケースを，ライフラインのやりとりをメッセージ（操作の呼び出し）として時系列に記述する．ライフラインの種類には要求分析で出てきたクラスを表す「エンティティ」，入出力などアクター（ユーザ，ハードウェア，外部システム）とシステムの境界での役割を担うクラスとなる「バウンダリ」，エンティティとバウンダリ間の処理の流れを制御する「コントロール」がある．シーケンス図の目的は，要求分析で定義したユースケースの振舞いを各オブジェクトの連携で実行できるように，各クラスの操作として割り振り，そのシグネチャを決定することである．ユースケースに記述したアクティビティ図の振舞いをクラスのメッセージシーケンスとして表すことを，アクティビティ図のシーケンス図へのマッピングと呼ぶ．マッピングを行うことによりユースケースを要求分析の段階から設計の段階へとトレーサビリティを確保して移行することが出来る．

3. マッピングの問題点

マッピングを行う際にはユースケース記述として適切な内容が定義されている必要があるが，アクティビティ図のアクションは本来自由形式の自然言語記述であり，処理内容が一意に解釈出来ない曖昧なものを書くことがある．また，アクションの対象となるデータをオブジェクトノードとして書かずにアクション内にのみまとめて書くこともある．このため，マッピングを行う際に振舞い元や役割が不明となる場合がある．松井ら[2]は，ユースケース記述としての形式の定義と以下の意味的な評価を行っている．

- 語句が記述されたパーティションの行為として妥当でなければならない．
- 基本・代替・例外は，システムが何らかの条件を判断し，判断結果に応じてつぎの行為を定めることで，そのフローが定義できる．この際，判断結果は，判断するアクションにつづくディシジョンノードと分岐フローのガードとして記述すべきである．

などが挙げられている．以上の点に加えて，

- アクションがあるデータに対して CRUD（Create, Read, Update, Delete）の振舞いや入出力を行っているということを明確にするために，アクションから続けて繋いでデータ（オブジェクトノード）を書く．
- ディシジョンノードを用いて条件分岐をした場合はマージノードで分岐を対応付けて終了を表す．

これらを記法として定義する．

4. UML 変換ツール

設計者は本来どのように対応付いているかを考えシーケンス図へのマッピングを行う必要があるが，既に要求分析の段階で必要と判明したクラスに対するアクション系列が分析されている．これをそのままシーケンス図にマッピングすることで，要求分析で記述された振舞いは漏れ・誤りがなく設計の段階へ持ち込める．そこで本研究ではこのマッピングを自動で行うツールを作成する．

4.1. 変換規則の概要

ツール上で行うマッピングを本研究では変換と呼ぶ．アクティビティ図の変換の規則を表１のように定義した．ディシジョンノードのガード条件は条件分岐の際に具体的な条件を書くためのものである．

表１　アクティビティ図の変換規則

アクティビティ図の要素	シーケンス図の要素
アクション	メッセージ
パーティション	ライフライン
オブジェクトノード	
ディシジョンノード（ガード条件付き）	複合フラグメント

A Conversion Tool for Ensuring Traceability between UML Requirements Analysis Model and the Design
†Kaito YOSHINO ‡Saeko MATSUURA
†‡Department of Electronic Information System, Collage of System Engineering and Science, Shibaura Institute of Technology

4.2. ライフラインの生成規則

アクションが属するパーティションはそのアクションの振舞い元となる．これらパーティションから対応するライフラインを生成する．さらに，後述するメッセージの生成ではアクションの先にあるオブジェクトノードからライフラインを識別する必要があるため，このオブジェクトノードにあるクラスを用いてもライフラインを生成する．変換の規則は表 2 のように定義した．

表 2　ライフラインの変換規則

アクティビティ図の要素	ライフラインの種類
システムパーティション	コントロール
アクターパーティション	バウンダリ
インタラクションパーティション	
アクション先のオブジェクトノードクラス	エンティティ（クラスと同名）

4.3. メッセージの生成規則

アクティビティ図のアクション内に書かれる文を読み取り，シーケンス図内のシグネチャが無い状態のメッセージとして変換する．メッセージの送信先対象とするライフラインはアクションに続くオブジェクトノードを読み取ることで判断する．このとき，オブジェクトノードがアクションの先に無い場合は自分自身（アクションの属するパーティション）に対応したライフラインにメッセージを送信する．また，アクション内で特定の動詞を使用した場合はさらに詳細なメッセージとして記述する．今回は頻繁に用いられる create メッセージに着目した．具体的な例をあげると，「**を生成する」という動詞を読み取り，オブジェクトの生成と判断することでシーケンス図に create メッセージを発生させる．

アクターやインタラクション中に出てくるアクションは標準入出力や GUI の入出力など汎用的なものに置き換えることが出来る．ここでシステム間の通信処理を汎用的に考えると図 1 のようなクラスが想定出来る．これを使用し，特殊なアクションノードである「シグナル送受信アクション」を用いた場合はシステム間が通信処理をしているものとし，該当するメソッドをメッセージとして記述する（図 2）．

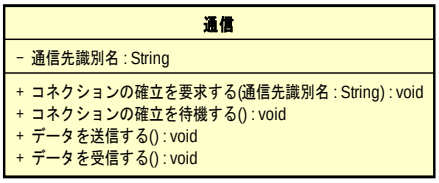

図 1　通信用クラス

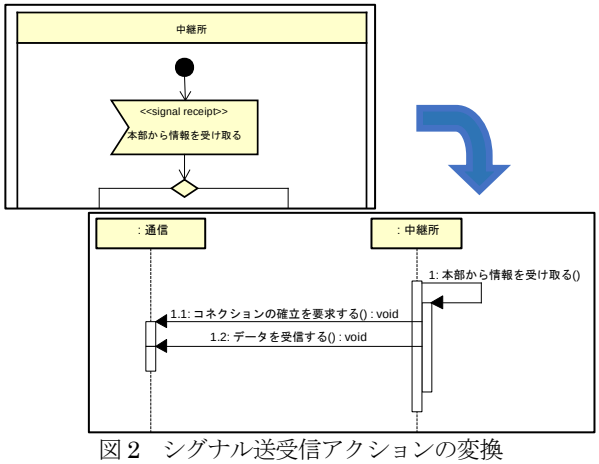

図 2　シグナル送受信アクションの変換

4.4. 複合フラグメントの生成規則

ディシジョンノードとガードを用いた条件分岐や反復する構造をアクティビティ図に記述した場合はシーケンス図内に制御構造を表現するための機能である複合フラグメントを追加する．

4.5. ツールの実装

本ツールは UML モデリングツールの一つである astah[3] 内のプラグインとして実装した．ツールには以下の機能がある．

- アクティビティ図の変換
- 変換の際に解釈不能な箇所の表示
- 変換の詳細設定

4.5.1. 解釈不能な箇所の表示

変換する際にオブジェクトノード内のクラス指定が無い場合はメッセージの送信先が解釈出来ないため，その場合は「修正」という名前のライフラインにメッセージを送信させることでツールの利用者にモデルの修正を促すことにした．

4.5.2. 変換の詳細設定

オプションの機能として好みに合わせて変換規則の一部や変換後の見た目を変更できる機能を用意した．ここでは 4.2. で述べた create メッセージを生成するための動詞の追加や，生成するライフラインの横の間隔を設定することなどが出来る．

4.5.3. ツールによる生成物

本ツールを用いることにより，シグネチャは定められていないがアクティビティ図のアクションに対応したメッセージが作られたシーケンス図が出力される．これにより要求分析の結果を踏襲した設計モデルを用いて，最終的な設計モデルの作成を支援する．利用者はクラス毎にシグネチャの未定義部分（戻り値．パラメータ）を決定し，要求分析の段階で記述されていなかったクラスの下部構造のクラスへの操作の委譲を追加する作業のみを行えばよいことになる．

5. 適用実験と考察

ツールを実際に適用するモデルとして「ソフトウェア設計論」で用いる教科書[1]内のモデルと，同じく本学科で行われる「情報実験 II」で作成したモデルを用意した．これらのモデルに対してツールによる変換を行った結果，アクティビティ図のアクションの順序通りにシーケンス図のメッセージが生成されていることが確認出来た．しかし，条件分岐などをツールで変換する場合は 3.のとおり条件の終わりを表すためのマージノードを使う必要があるため，対応付いたマージノードがないモデルはうまく変換が出来なかった．これは本来設計者が解釈してシーケンス図を書くとしたときに，アクティビティ図内の記述を省略していたためであると考えられる．同様に課題点として，条件分岐の際にアクションを書かずガードに条件判定を書いた場合や複数の入力をまとめたアクションなどは意図したメッセージを生成することが出来ない．また，同じ内容のアクションを異なる書き方にした場合はメッセージも異なるものが生成されるため，処理の流れは正しいが冗長性に問題があるシーケンス図を出力する場合もある．そのため，ツールを使用する際は記法に 3.で述べた問題点が無いかと，同じアクションの表記が統一されているかを確認する必要がある．今後は定義した記法を省略したモデルでも変換が行えるようにしたい．

参考文献

[1]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
[2]松井，奥田，野呂，岡田，加藤，渡辺，松浦，モデリング能力育成を目的としたユースケース記述の自動評価手法，電子情報通信学会論文誌，VOL.J97-D，NO.3，p.465，2014
[3]astah，http://astah.change-vision.com/ja（2018/01/09 参照）