Vol.2019-SE-201 No.2
2019/3/7

要求分析と基本設計間のトレーサビリティ確保のためのユースケース記述変換ツール

吉野魁人[†1]　松浦佐江子[†2]

ソフトウェアの品質や保守性を高めるためには開発工程でのトレーサビリティの確保が重要である．要求分析段階におけるユースケース分析では，一般にテンプレート形式の自然言語によってユースケースを記述する．次の設計段階ではユースケース記述に基づき，シーケンス図を用いて記述内容を分析し，振舞いをクラスに割り当てる．しかし，人手による解釈の違いによってはこの振舞いの割り当てに漏れや誤りが発生する可能性がある．本研究ではアクティビティ図を用いてユースケースを記述することでその実現可能性を検証し，ユースケース記述の要素をシーケンス図の要素へ自動変換することにより，トレーサビリティを確保する手法を提案する．

UML Requirements Analysis Model Conversion Tool to Ensure Traceability of the Basic Design Model

KAITO YOSHINO[†1]　SAEKO MATSUURA[†2]

In order to improve the quality and maintainability of software, it is important to ensure traceability in the development process. In the use case analysis at the requirements analysis stage, the use case is generally described by the natural language of the template form. In the next design stage, based on the use case description, software designer analyzes the description content using the sequence diagram and assigns the behavior to the class. However, depending on difference in interpretation of software developer, leakage and errors may occur in this behavior assignment. In this study, we propose a method to ensure traceability by verifying its feasibility to describing the use case by using the activity diagram and automatically converting the elements of the use case description to the elements of the sequence diagram.

1. はじめに

ソフトウェア開発ではシステムの保守性や品質を高めるために，トレーサビリティ（追跡可能性）と呼ばれる概念が重要である．情報システム開発においては各開発工程における成果物間の関連と定義されている[1]. また，Gotel ら[2]は，要求のライフサイクルを記述して前工程と後工程をともに探索できる能力のことを要求トレーサビリティと定義している．トレーサビリティを確保することにより，要求定義から設計，ソースコードまで一貫した管理ができるようになり，上流の成果物から下流の成果物への対応を漏れなく確認することができる．また，開発中のシステムに変更が生じた場合に影響を受ける範囲が明確となる．しかし，実際のシステム開発の現場ではシステムが複雑化・大規模化するにつれてトレーサビリティの管理が煩雑になり，作業負荷が高くなる．

効率的に要求定義と設計間のトレーサビリティを管理する方法としてこれまでにさまざまな支援のツールが開発されている[3][4].これらは成果物である要求定義と設計書を項目ごとに分解し，それぞれの対応を後述するトレーサビリティマトリックスやツリー形式によって表示を行う．しかし，この対応付け自体はツールの利用者が行うため，成果物の文書の解釈次第によっては漏れや誤りが発生する可能性がある．

そこで本研究では，文書形式ではなく Unified Modeling Language（UML）のモデル図を用いて要求分析，設計を行うことにより，モデルの要素間の対応としてトレーサビリティを確保する手法を提案する．本方式では，要求分析の工程で機能要件を定義するユースケース記述をアクティビティ図とクラス図によって分析し，これらの各モデル要素を設計段階で用いられるシーケンス図の要素へと変換して，その対応関係に基づき要求定義と設計情報の関係を保存する．

本稿では，2 章で本研究と関係する従来の手法や研究を紹介し，3 章で本研究のアプローチを説明する．4 章では実装した変換ツールについて説明する．5 章では関連研究について述べ，6 章で本研究のまとめと今後の課題について述べる．

2. 従来研究：現状と問題点

要求トレーサビリティの確保のために重要な点は要求定義や設計書，テストケースなどを項目ごとに分解し，それぞれの対応付け情報であるトレーサビリティリンクを作成することである．これにより，成果物を相互に参照することができる.このときの課題として以下の点があげられる．

†1†2 芝浦工業大学
Shibaura Institute of Technology

●トレーサビリティの保守

トレーサビリティリンクを作成するために一般的によく用いられるトレーサビリティマトリックス（表 1）は分解された項目を縦軸，横軸に並べて関連の有無を表に書き込むものである．この例では要求項目と設計項目の関連の有無を「●」で表している．また，設計５の関連がどの要件とも無いため，設計段階で新たに追加されたものであることが分かる．分解された項目を表に書き込む作業はシステムが大規模なものであるほど労力がかかり，誤りも発生しやすくなる．また，設計に変更が生じた場合には合わせてトレーサビリティマトリックスも変更するため，保守を継続して行う必要がある．

表 1　トレーサビリティマトリックスの例

	設計1	設計2	設計3	設計4	設計5
要件A	●				
要件B		●	●		
要件C			●		
要件D				●	

●トレーサビリティ確保に関する理解不足

トレーサビリティ確保の現状として，宇田川[1]はその管理コストに対する効果が十分に認識されておらず，一般のシステム開発プロジェクトではトレーサビリティの管理が定着しているとは言い難いと述べている．また，トレーサビリティを開発効率の向上のために活用した例は少なく，単純に対応に漏れが無いかの確認にのみ使われる場合が多い．これには上記の保守の問題も含まれるが，システムに変更が生じた場合に影響を受ける範囲を特定するためにはトレーサビリティリンクが常に最新である必要があるため，このトレーサビリティリンクの保守作業をより行いやすい環境が求められる．

山中ら[4]はトレーサビリティの保守を効率的に行うためのツールとして，要求仕様書や設計書から機能要求などの項目とそれらの設計項目を自動的に取り込み，ツール画面上でのドラッグアンドドロップや文書中の対応付け情報からトレーサビリティを確保する方法を開発している．また，このツールで得たトレーサビリティリンクの活用法として，設計時には対応する要求を参照して作業し，その要求をすべて反映したことを確認するまで設計を続けることによって設計漏れを防止するといった点があげられている．これにより，要求項目と設計項目の対応に誤りがある場合でもそれに気付きやすくなり，トレーサビリティリンクを直ちに修正することができるといった利点がある．ただし，文書の解釈違いによって最初のトレーサビリティリンクに誤りが多く生じた場合はこの修正作業も多く行う必要が生じる．

トレーサビリティを効率よく確保するためには，工程ごとの成果物の定義とその関連の効率的な作成が重要である．開発者は工程の成果物の作成に注力したいので，次工程の成果物への関連を自動的に生成できることは望ましい．要求には機能要求と非機能要求があり，これらをどのように整理して定義するかが問題である．本研究では，UML を用いて，要求分析において，機能要求を核にユースケースを分析し，実装までのトレーサビリティの効率的な確保の方法を提案する．

3. 要求分析モデルから実装までの UML モデリング

3.1 ユースケース記述

要求分析段階におけるユースケース記述は，一般的にテンプレートに従った自然言語によってシステムとアクター間のやり取りを記述し，システムの機能要求を明らかにするためのものである．このアクターとはユーザやハードウェア，外部システムなどシステムにたいして何らかの振舞いを行う人やものを表す．記述要素としてはユースケース名，事前条件，事後条件，アクター，基本フロー，代替・例外フローがあげられる．しかし，これらの記述方法は自然言語による非形式なものであるため，振舞いの主体やフローの分岐点と合流点，および対象となるデータやその構造が必ずしも明確にされない場合がある．

この問題に対して，我々はユースケースのフローをより

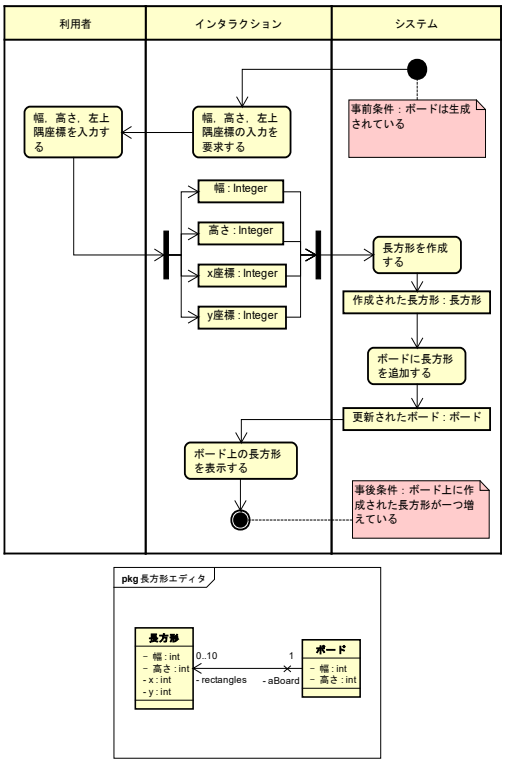

図 1　ユースケース記述とクラス図

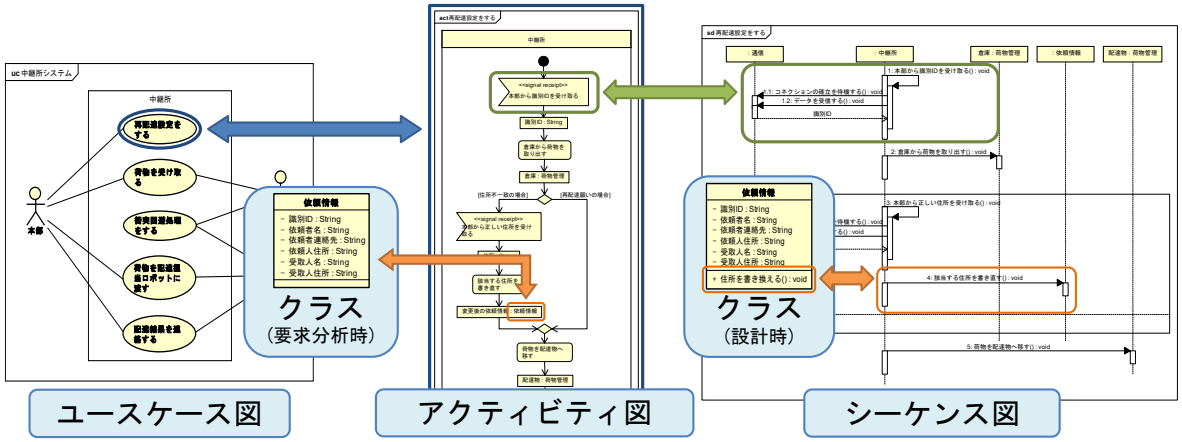

図 2　各モデル要素間のトレーサビリティリンク

可視化・形式化してデータに不整合が起こらないか検証するために，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なっている[5][6]. 例としてアクティビティ図によるユースケース記述と，分析時に出てくるデータを表したクラス図を図 1 に示す．ここでは入力された数値によってキャンバス上に長方形を描画する簡単なプログラムを例としている．

各モデル要素の使われ方を説明する．ユースケース内の振舞いの主体はシステム内部とアクター，インタラクションに分離し，アクティビティ図のパーティションを用いて表している．さらに開始ノードから終了ノードまでを，振舞いを表したアクションノードとフローでつなぎ，システムとアクター間のやり取りとして可視化する．ここでインタラクションパーティションはアクターの入力に対する要求や出力など，アクターとシステムの相互作用に関わるアクションを割り当てるものであり，これによりシステム内部の振舞いとアクターとシステムのやり取りを分離して整理することができる．また，基本・代替・例外フローはディシジョン・マージノードとガード条件を用いることによって定義する．振舞いの対象となるデータはオブジェクトノードあるいはアクション内の記述の目的語にすることによって表す．このように可視化を行うことによって自然言語による記述よりも直感的になり，曖昧な表現に気づきやすい．また，データをオブジェクトノードで表すことによってユースケースにおける必要な入出力を見極め，その実現可能性を検証しやすくなるといった利点がある．要求分析の段階では，データの型を定義するものとして Object Constraint Language（OCL）[7]を用いる．これに用意されている文字列や数値，真偽値といった基本型やコレクション型，列挙型によって対象データの特徴を表現できる．実装段階ではこれらの型を参考にして適切な API を選択すればよい．図 1 の例では，幅や高さといった複数の入力値を Integer 型で表している．また，この段階では入力の順序を特に定めないためフォーク・ジョインノードでフローを挟んで動作を並列にしている．

一方，クラス図を使ってシステムにおけるデータをクラスとその属性として整理すると，アクティビティ図内のアクションと緩い結びつきを持たせることができる．ここで緩い結びつきとは，アクション内の記述における目的語と述語や，オブジェクトノードとアクションの位置関係といったことを指す．例えば，「**を生成する」といったアクションの直後にその対象のデータを表したオブジェクトノードを記述することによって，データに対するアクションの明確な役割を表すことができる．これにより，設計段階では，各クラスの責務に合わせたメソッドの割り振りを考えやすくすることができる．

3.2　要求分析モデルと設計モデルの関係

設計の初期段階である基本設計ではユースケース記述に基づき，シーケンス図を用いてシステムの振舞いをクラスのメッセージシーケンスとして表す．ここでアクティビティ図によるユースケース記述を用いた場合, 3.1 節で述べた緩い結びつきが存在するため，この関係に基づいてシーケンス図を作成することによって要求分析で記述された振舞いをクラス固有の振舞いとして分析・整理していくことができる．そこで本研究ではアクティビティ図によって書かれたユースケース記述を対象とし，緩い結びつきを読み取ることによってアクション系列をシーケンス図へ自動的にマッピングするツールを開発する．これにより，要求分析で得た振舞いを基本設計へと漏れ・誤り無く移行し，次の段階の詳細設計・コーディングへと続けることができる．本研究では要求分析･設計段階の成果物の対応関係として，それぞれで用いられるモデルの要素間の対応付け情報をトレーサビリティリンクとして定義する．対象となる各モデル要素間のトレーサビリティリンクを図 2 に示す．

3.3　モデルの形式化

自動的にマッピングを行うためには，アクティビティ図

の要素の使われ方をより形式的にする必要がある．松井ら[5]はモデルに定義した要件を読み手に正しく伝えられるという観点から，アクティビティ図を用いたユースケース記述の方針を定めている．以下に 3.1 節で既に説明したもの以外から主な点を述べる．

- アクティビティ図名をユースケース名と同じとし，対応を明らかにする．
- 開始から終了までのパーティションを跨るアクションのフローで，振る舞いの順序を表す．基本及び例外フローが存在するので，開始は一つであるが終了は各フローに対応して一つ以上になる．
- アクションの記述文で，行為の対象と動作を表す．記述文では目的語でアクションの対象となるデータを，動詞でその主体の振る舞いを表す．
- ディシジョン・マージノードにより，振る舞いの代替及び例外の分岐と合流を表す．代替及び例外による分岐後のアクションは同じ主体の行為でなければならないので，同じ主体のパーティションに定義する．また，分岐と合流を明確にするために，別のノードを用いる．
- フォーク・ジョインノードは順序を定めないアクション系列をこれらの対によって定義する場合に用いるが，振る舞いの主体は同じであるので，同じパーティションに定義する．
- 事前条件・事後条件はそれぞれ開始・終了ノードにノートとして記述する．
- アクションの対象となるデータをオブジェクトノードで適切な位置に適切なインスタンス名と分類子となるクラスで定義する．

最後の点はアクションとデータの関係を読み取る上で重要である．しかし，ここでは具体的に「適切な位置」は設定されていない．この適切さを考えるためにはデータに対してどのような振舞いを行うのかを考慮する必要がある．そこで，データに対する基本的な操作である CRUD（Create, Read, Update, Delete）の観点から，本学で行われている授業である「情報実験 II」で生徒たちが作成したモデルなどを参考にし，慣用的にアクションで用いられる動詞の種類及びアクションとオブジェクトノードの位置関係を定めた．表 2 に各 CRUD の動詞，図 3 に具体的な位置関係の例を示す．

表 2　各 CRUD アクションの動詞

Create	生成する，作成する，作る
Read	取得する，読み取る，検索する
Update	更新する，変更する
Delete	削除する，消去する，消す

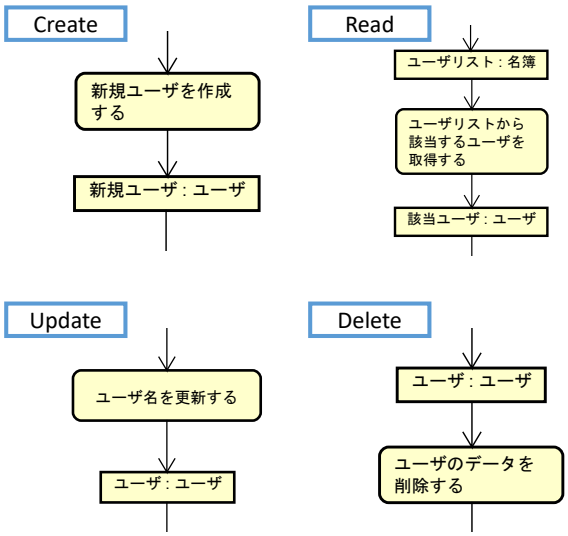

図 3　CRUD のアクションとオブジェクトノードの位置関係

対象の動詞を用いた場合，その種類によってアクションに対応するオブジェクトノードを図 3 のように見つけ，そのオブジェクトノードのクラスの振舞いと判断してシーケンス図にマッピングを行う．Read はこの例の場合，該当ユーザの取得元であるユーザリスト（名簿クラス）の振舞いと判断し，該当ユーザを戻り値としてマッピングする．このようにすることによってより形式的かつ自然な図の見え方になるため，レビュー時にも不自然なモデルに気づきやすくなる．

4. ユースケース記述変換ツール

本ツールは UML モデリングツールの一つである astah* professional[8]内のプラグインとして実装する．本ツールでは，アクティビティ図で書かれたユースケース記述からシーケンス図を自動的に作成し，アクションとメッセージといった振舞い単位のモデル要素のトレーサビリティリンクを作成，保存することによって要求分析と基本設計間のトレーサビリティを確保する．

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．アクティビティ図全体の変換規則を表 3 のように定義した．アクションはクラスのメソッド呼び出しを表すメッセージに，アクションの主体であるパーティション名とアクショ

ンの対象となるオブジェクトノードはメッセージを送受信する主体であるライフラインに設定した．また，ガード条件が付いたディシジョンノードは alt 複合フラグメントを用いて，条件による分岐を表すことにした．さらに，アクティビティ図の開始・終了ノードに付けられるノートに書かれた事前条件・事後条件は，後の詳細設計で行うメソッドのシグネチャ制定の参考とするためにそのままシーケンス図の上部と下部に表示することにした．その他，アクションやオブジェクトノードに付けられたノートは変換後の要素であるアクションやライフラインに同様の内容を書いたノートを付けることにし，設計の参考とする．

表 3　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（alt）

4.2 ライフラインの変換規則

アクティビティ図で振舞いの主体を表すパーティションはシステム内部とアクター，インタラクションに分離されている．これらとアクションの対象となるオブジェクトノードは設計の初期段階で用いられるエンティティ，バウンダリ，コントロールといったクラスに分類してライフラインに変換する．各要素の持つ意味的な観点から表 4 のように変換規則を定めた．

エンティティクラスはシステムがサービスを提供するために管理するデータであるためオブジェクトノードで表されたクラス，バウンダリクラスはアクターとシステムとの境界で相互作用するデータであるため，同様に境界部分で振舞いを行うアクターとインタラクションパーティションを割り当てた．また，コントロールクラスはエンティティとバウンダリ間の処理の流れを制御し，ユースケース全体の処理を行うクラスであるため，システム内部を表したシステムパーティションを変換元とした．

表 4　ライフラインの変換規則

アクティビティ図の要素	ライフラインのクラス
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	
アクションの対象となるオブジェクトノード	エンティティ（オブジェクトノードのクラスと同名）

4.3 メッセージの変換規則

アクションに記述された文はシグネチャが無い状態のメッセージとして変換する．このとき，メッセージの送信元はユースケースの処理を制御するコントロールクラスのライフラインとする．メッセージの送信先とするライフラインのクラスは以下の手順で決める．

(1) アクション内の文を解析し，述語の動詞と目的語を抽出する．
(2) アクターパーティションやインタラクションパーティションに書かれたアクションは，そのままバウンダリクラスのライフラインを送信先とする．
(3) 3.3 節で述べた CRUD の特性を持つ動詞がアクションで使われている場合は，その対応するオブジェクトノードのクラスを送信先とする．
(4) CRUD の動詞以外の場合は目的語に書かれたデータをクラス図から探索し，クラス名またはロール名と照合する場合はそのクラスを送信先とする．
CRUD の動詞であってもオブジェクトノードが書かれていなかった場合は省略されていたと解釈し，同様に目的語から送信先を探す．
(5) 目的語に対象のクラスが無かった場合，送信先が判断できなかったとして「修正」という名前のライフラインを作成し，それを送信先とする．
(6) 目的語がそもそも無かった場合は，自分自身（コントロール）の振舞いと解釈し，メッセージの送信先をコントロールのライフラインにする．

(2)は入出力といった境界部分をすべてバウンダリクラスにまとめ，後の詳細設計で具体的にどのような外部インタフェースの実装を行うか検討するために行う．また，(5)はツール上で判断できないモデルに対して利用者に修正を促すために設けたものである．(6)では目的語が無いアクションを粒度の高い（抽象度の高い）振舞いであると考え，ユースケース単位の処理としてコントロールのライフラインを送信先に設定した．

また，astah のシーケンス図にはオブジェクトの生成を表すために<<create>>のステレオタイプが付けられた Create メッセージの作成や，戻り値を表す Reply メッセージを作成する機能がある．今回定めた CRUD の動詞の中で，Create にあたる動詞はこの Create メッセージを利用して生成を表すことにする．また，Read の動詞を使用した場合は 3.3 節で述べたように戻り値を表現する必要があるため，Reply メッセージを利用する．

4.4 変換の詳細設定

本ツールを使用するにあたり，使いやすさの向上や開発現場の規則などに合わせるために，変換規則の一部を変更できるようにした．これらの例として，表 2 の CRUD の動詞を自由に追加・消去する機能や，具体的なパーティシ

ョン名を付けた場合にどれがシステム･インタラクション･アクターパーティションなのかを設定できるようにした．

4.5 トレーサビリティリンクの保存

これまでに述べた変換規則によってユースケース記述の要素は一通りシーケンス図へとそのまま移行し，アクションとメッセージの対応が付いた状態になる．続く詳細設計では，このシーケンス図の仕様を実現できるように，内部の詳細な設計を行う．具体的にはメッセージをクラスの構造に従って分割し，そのクラスのメソッドとして適切な名前やシグネチャを与える必要がある．しかし，これらの変更が生じた場合，アクションとメッセージの対応関係が図を見るだけでは分かりにくくなる．これの解決として変換時にアクションとメッセージの対応を保存し，横軸と縦軸をアクション名とメッセージ名にしたトレーサビリティマトリックスを自動的に作成するとともに，どのようにモデル要素の内容が遷移していったかの階層構造を作成して表示することにした．トレーサビリティマトリックスは変換後のシーケンス図を消去しない限りは保存，管理を行うため，メッセージの追加を行った場合は設計段階で新たに追加したことが分かるようになる．また，変換して出来たメッセージの活性区間の中で新たにメッセージを追加する場合は，そのメッセージも変換元のアクションと関係があるものとし，トレーサビリティマトリックスに関係を追加する．

以下に具体例を述べる．図 1 のモデルを変換するとまず図 4 のようなシーケンス図と表 5 のトレーサビリティマトリックスが生成される．この段階ではアクションとメッセージが変換規則に従って 1 対 1 の関係になっており，トレーサビリティマトリックスもその関係性が表示されている．次に，クラスのメソッドとしてふさわしい名前付けやシグネチャの制定と，要求分析時には無かった長方形への入力値に問題が無いか（値が負であるか，ボードの外に出る座標でないかなど）を判定するメッセージやボードに長方形を追加できるかを判定するメッセージを加えてみる（図 5）．このとき，トレーサビリティマトリックスは表 6 のように更新される．メッセージ名の変更と，追加されたメッセージとアクションとの関係が追加されている．

5. 関連研究との比較

Kang ら[9]は，断片的なシナリオを表したシーケンス図

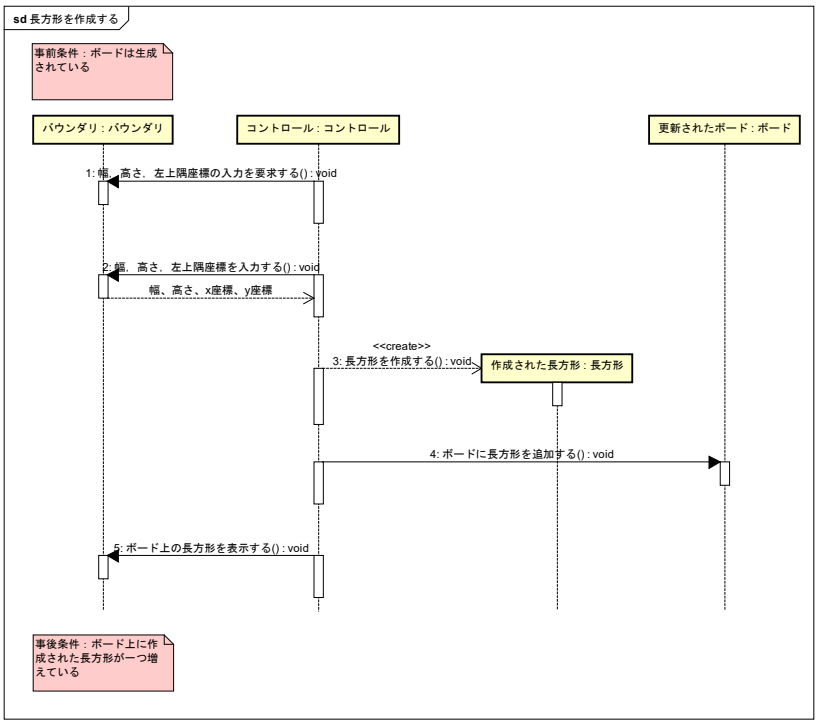

図 4　図 1 から変換されたシーケンス図

表 5　生成されるトレーサビリティマトリックス（イメージ）

		メッセージ				
		幅、高さ、左上隅座標の入力を要求する	幅、高さ、左上隅座標を入力する	長方形を作成する	ボードに長方形を追加する	ボード上の長方形を表示する
アクション	幅、高さ、左上隅座標の入力を要求する	●				
	幅、高さ、左上隅座標を入力する		●			
	長方形を作成する			●		
	ボードに長方形を追加する				●	
	ボード上の長方形を表示する					●

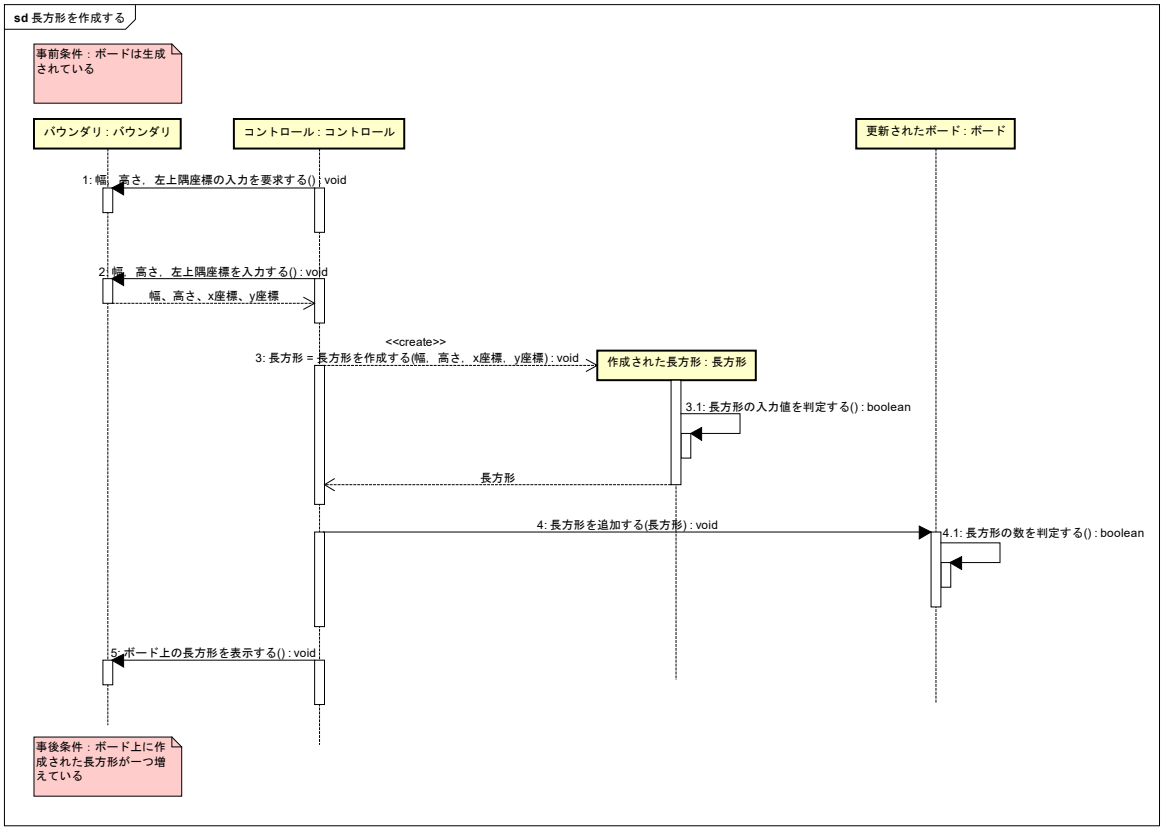

図 5　変更を加えたシーケンス図

表 6　更新したトレーサビリティマトリックス（イメージ）

		メッセージ						
		幅、高さ、左上隅座標の入力を要求する	幅、高さ、左上隅座標を入力する	長方形を作成する	長方形の入力値を判定する	長方形を追加する	長方形の数を判定する	ボード上の長方形を表示する
アクション	幅、高さ、左上隅座標の入力を要求する	●						
	幅、高さ、左上隅座標を入力する		●					
	長方形を作成する			●	●			
	ボードに長方形を追加する					●	●	
	ボード上の長方形を表示する							●

の仕様書から全体的なビジネスプロセスモデルとしてのアクティビティ図を自動的に作成する手法を提案している．本稿の手法とはアプローチは異なるものの，各モデルの変換される要素は同様であり，本研究における変換の定義が妥当であることが分かった．また，この研究では各モデル間のトレーサビリティは考慮していないため，モデルの変更が生じた場合は図の対応が不明となる．本研究では要求モデルと設計モデル間のトレーサビリティを保存するため，モデル要素の対応関係が不明になることは無い．

徳本ら[10]は，要求トレースを目的として開発要件や非機能要件といった要件の種類と，要件定義や詳細設計などの各工程を表したタグ ID を成果物に付け，ID とタグの位置によってグループ化することにより自動的にトレーサビリティリンクの構築を行っている．これにより，各要件から設計書などのドキュメント類と相互参照を可能としている．しかし，この方式でもタグ付けの漏れや誤り，あるいは開発者によってタグ付け箇所が変わるといった点が生じるため，タグ付け作業自体の負担が大きい．本研究では現状ユースケース単位の機能要求のみを対象としているが，要求分析のモデルから基本設計段階のモデルに変換することで両工程間のトレーサビリティを確保している．また，変換時に各モデル要素の対応をツール内で保存することによって成果物を相互に参照できるため，正しいモデルを書けば人が特別に他の作業を行う必要が無い．

6. おわりに

本稿ではトレーサビリティ確保のために，要求分析段階で得られたユースケース記述の要素からシーケンス図を自動生成することによる基本設計の支援方法を提案した．ツールの利用者はその後シグネチャの制定や要求分析の段階では分析されていなかったメソッド内部の構造を追加する詳細設計のみを行えばよいことになる．これにより，従来では人が行っていたリンクの作成･保守作業が不要となり，リンクの漏れや誤りを無くすことができると考えられる．

また，シーケンス図のメッセージに関連付くノートに書かれた OCL 式からソースコードを生成することでトレーサビリティを確保する研究[11]もされており，本研究のツールと組みあわせることで要求から一貫した成果物の管理とソースコードの生成が期待できる．

今後の課題として，ツールの未完成部分の実装が挙げられる．今回は変換の定義とメッセージの変換までを実装したが，トレーサビリティリンクを実際にどのように表現するかまでは決定していない．現状のアイデアでは，アクションとメッセージの対応をトレーサビリティマトリックスで表示し，各モデル要素の変更履歴を階層状に表示することを想定しているが，より詳細な仕様は今後実装を進めていくとともに検討していく．さらに，トレーサビリティリンクを astah 上以外からでも確認できるように別のファイル形式に出力することも考える．

また，ツールの評価方法として本学の電子情報システム学科内の学生が作成した様々なモデルに対してツールを適用し，今回定めた変換規則によって問題なく変換が行われるかを確認する．また，ツールの利用者に対して本ツールが有用であるかのアンケートを実施し，トレーサビリティの確保手段としての有効性を考察する．

参考文献

[1]宇田川佳久：情報システム開発標準におけるトレーサビリティの事例と今後，情報処理 Vol.51 No.2，p.151（2010）．
[2]Gotel, O.C.Z. and Finkelstein, A.C.W.: An analysis of the requirements traceability problem, Proceedings of IEEE International Conference on Requirements Engineering (1994).
[3]microTRACER, 〈https://www.dts-insight.co.jp/product/eps_tool/microtracer/index.html〉 [アクセス日: 2019/1/7].
[4]山中美穂，大音真由美，山元和子：トレーサビリティーを確保してソフトウェア開発を効率化する要求管理ツール，東芝レビュー Vol.73 No.5（2018）．
[5]松井駿介，奥田博隆，野呂惇，岡田康治，加藤真，渡辺大貴，松浦佐江子：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（2014）．
[6]松浦佐江子：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）．
[7]Warmer, J. and Kleppe, A.: The Object Constraint Language: Getting Your Models Ready for MDA, Second edition, Addison-Wesley (2003). 竹村司（訳）: UML/MDA のためのオブジェクト制約言語 OCL 第 2 版，エスアイビー・アクセス（2004）．
[8]astah* professional, 〈http://astah.change-vision.com/ja/product/astah-professional.html〉 [アクセス日: 2019/1/7].
[9]Kang, S., Kim, H., Baik, J., Choi, H. and Keum, C.: Transformation Rules for Synthesis of UML Activity Diagram from Scenario-Based Specification, 2010 IEEE 34th Annual Computer Software and Applications Conference (2010).
[10]徳本修一，谷垣宏一，高橋洋一，中島毅：タグ情報による要求トレーサビリティリンクの構築と可視化，研究報告ソフトウェア工学（SE） Vol.2015-SE-187，No.46（2015）．
[11]松隈暖，松浦佐江子：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，情報処理学会第 81 回全国大会（2019）．（掲載予定）