Vol.2021-SE-207 No.20
2021/3/1

データ依存解析によるユースケース記述中の不吉な臭い検出

関 洋太朗[1,a)]　林 晋平[1,b)]　佐伯 元司[1,c)]

概要：ユースケース記述の品質はソフトウェア開発全体の品質やコストに関わり重要であるが，現実的には品質問題が発生している．その中でも，文章の長さや動詞の数といった構造上の特徴によらない問題については検出が難しく，自動化があまりなされてこなかった．本論文では，記述中の低品質と思われる箇所を指す不吉な臭いのうち，データの依存関係が原因なものを自動検出するために，動作の入出力となるデータをノードに保持した制御フローグラフを構築し検査を行うツールを開発した．本手法で作成したツールは，文中の動詞を辞書及び単語の類似度を用いて 7 種類に分類し，各分類に割り当てられた格フレームと記述の構造上の特徴に基づいてグラフを構築し，“更新するデータはそれ以前に作成されているべき” といったデータの依存関係を解析することで，不吉な臭いを自動検出する．また，作成したツールを評価するために，架空の鉄道会社による列車予約システムを表す 5 つのユースケース記述を用いて調査を実施した．

1. はじめに

ソフトウェア開発において，顧客の要求を正しく理解し，実装へと結びつけることは非常に重要である．顧客の要求を正しく理解できなかった場合，作業工程に手戻りが発生し，修正に多大なコストがかかる可能性がある [1].

ソフトウェア開発プロセス中の要求獲得及び要求理解ステップにおいて，ユースケースモデルが広く用いられている [2]．ソフトウェア開発者は，ユースケースモデルを使用することで，システム設計案の妥当性がチェック可能になり，その設計が全ての要求を満たしていることを保証できるようになる．他にも，ユースケースモデルはテスト計画や開発スケジュールの作成，ユーザーガイドの記述など，ソフトウェア開発の幅広い範囲で用いられる [2]．上記のように，ユースケースモデルはソフトウェア開発において幅広い役割を担っている．

それゆえに，ユースケースモデルの品質を高めることは，高品質のソフトウェアを低コストで開発するために重要である．しかし，ユースケースモデルの品質問題は頻発している．その原因の 1 つとして，ユースケースモデルの記述部であるユースケース記述に，公式のガイドラインやチェックリストが存在しないという点が挙げられる．ユースケースモデリング手法において，各ユースケースをどれくらい詳細で具体的に記述すべきかのガイドラインがなく [3]，実際に低品質なモデルが数多く作成されている [4].

この公式のガイドライン及びチェックリストが存在しないという課題に対し，多くの非公式なガイドラインとチェックリストが提案されている．Phalp らは，7Cs と名付けられたユースケース記述に対する自然言語で記述されたチェックリストを提供した [5]．また，Törner らは，欠陥の特徴ごとに分類された評価基準と，その欠陥の位置を特定するための質問リストを提供した [6].

また，チェックリストの適用をツールで自動化することで，適用の手間を省くとともに検出の属人性を解消した手法もあり，Liu ら [7]，El-Attar[8] らはそれぞれ記述の文の長さといった構文上の特徴や，構造上の特徴をもとに低品質箇所を指摘するツールを，Seki ら [9] は記述中の低品質と思われる箇所を “不吉な臭い” として指摘するツールをそれぞれ提案している．

しかし，これらの手法の問題点として，記述の意味を考慮しなければならないものについては自動検出がなされていないという点があげられている．ユースケース記述は，見た目が単純明快であるだけでなく，記述の内容も高品質であるべきであり [10]，記述の内容をより良くするためには，記述の意味を考慮した不吉な臭いの検出が重要である．その中でも，データの依存関係に基づく不吉な臭いが記述中に残されてしまった場合，システムの詳細設計段階まで開発が進んでから発覚する可能性があり，手戻りが大きく

[1] 東京工業大学情報理工学院
Tokyo Institute of Technology, Tokyo 152–8550, Japan
[a)] yotaro@se.cs.titech.ac.jp
[b)] hayashi@c.titech.ac.jp
[c)] saeki@se.cs.titech.ac.jp

Vol.2021-SE-207 No.20
2021/3/1

座席を選択する

アクタ

- 予約者
- 管理者

概要

必要に応じて座席を選択する

事前条件

- システムは予約したい列車を特定している。

基本系列

1. システムは今の予約状況を図でユーザーに表示する。
2. ユーザーは空いている席を選択する。
3. ユーザーは予約ボタンを押す。
4. システムはユーザーが予約したい席のアクセシビリティを検査する。

事後条件

- 予約確認画面に進む。

図 1　不吉な臭いを含むユースケース記述の例

なってしまう．

例として，**図 1** に示すユースケース記述では基本系列の 1 番で “予約状況” というデータに対して操作をしており，この文を実行するには事前に “予約状況” が作成されていなければならない．しかし，記述中のどのステップでも “予約状況” というデータは作成されておらず，基本系列の 1 番を実行できなくなってしまう．その結果，この記述を実際に実装しようとしても正しく座席選択機能が作成できない．この例は，“動作の対象は事前に作成されているべきという条件” を満たしていないため，Seki ら [9] の定義した臭いのうち “条件を無視した動作” に当てはまる．

本論文の目的は，ユースケース記述の不吉な臭いのうち，“条件を無視した動作” のようなデータの依存関係に矛盾を生じている臭いの検出を自動化することである．また，評価については実際のユースケース記述を用い，不吉な臭いを高い精度で検出することが本論文の目標である．

本論文の構成を以下に示す．2 章では，実際にデータ依存解析によって不吉な臭いを検出するための手法について説明する．3 章では，データ依存を解析するために行う動詞の分類について説明し，4 章では，動詞の分類から得られた動作の対象や源泉となる単語を元にグラフを構築し，検査式を適用することで不吉な臭いを検出する処理について詳細に説明する．5 章では，提案手法を実装したツールについて説明する．6 章では，提案手法を実際のユースケース記述に対して適用し，その結果について考察する．7 章では，関連研究について述べる．8 章では，本論文の結論と今後の課題について述べる．

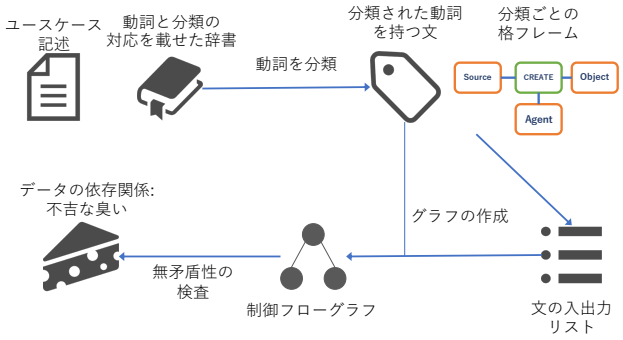

図 2　提案手法の概要

2. データ依存解析によるユースケース記述の分析手法

2.1 概要

本手法では，データの依存関係を検査するために，それぞれのデータが記述中のどこで登場していてどこで使用されているかの情報を保持したグラフをユースケース記述から構築する必要がある．そして，そのようなグラフを構築するためには記述中に現れる単語から動作の対象や情報の源泉を特定する必要があるため，本手法では動詞を 7 種類に分類し．その格フレームに基づいて文中に出現する単語を解析する処理を行う．

提案手法の流れを**図 2** に示す．まず，動詞と分類の対応を載せた辞書とユースケース記述を入力することで，記述中の各文章の動詞を分類し格フレームを得る．そして，得られた格フレームを元に文中に登場する入出力データを特定する．次に，記述中での文の実行順序を元に制御フローグラフを構築し，それぞれのノードに入出力となるデータの情報を付与する．最後にこのグラフに対して “動作の入力となるデータは事前に作成されているか” といった条件を調べることで不吉な臭いを検出する．

2.2 実際の検出例

提案手法で実際に不吉な臭いを検出する流れを，図 1 のユースケースを用いて説明する．

まず事前条件（$pre\_conditions$）について着目すると，この文では “特定する” という動作を “列車” というデータに対して実行している．ここで，“特定する” という動作は動作対象のデータをユースケース内に取り込むはたらきをしているため “$RECEIVE$” という分類に割り当てることができ，動作主（$agent$）が “システム” で動作対象（$object$）が “列車” であることから，この文を実行することで “列車” というデータがユースケース内に出力されていることがわかる．この文をグラフのノード $n_0 = \langle idx = 0, agent = \text{システム}, object = \text{列車}, verb = \text{特定}, type = RECEIVE \rangle$

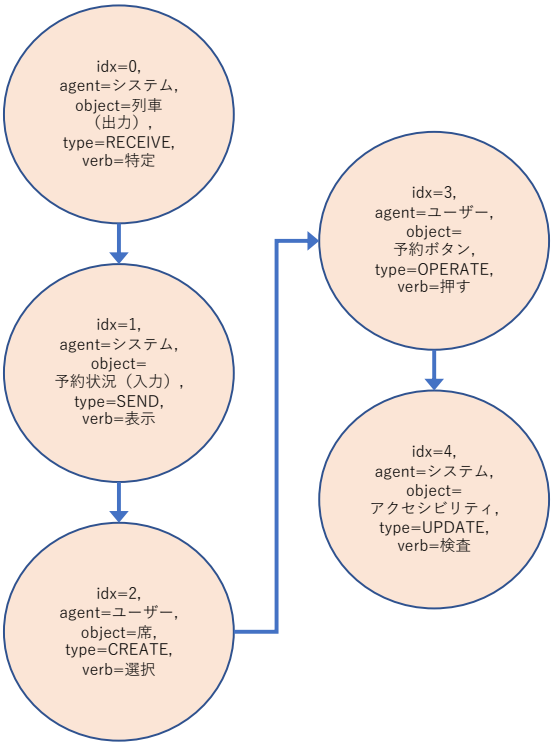

図 3 図 1 の記述から作成したグラフ

と表現する．ここで idx が 0 なのは，事前条件が記述中で一番最初に判定されるべきであることを表すためである．

次に基本系列（$main$）の 1 番について着目すると，この文では “予約状況” というデータを “表示” していることから，ユースケース内に既に存在している “予約状況” というデータを入力として外部に送り出す “$SEND$” という動作に分類することができる．同様に本手法でこの文をグラフのノード $n_1 = \langle idx = 1, agent = \text{システム}, object = \text{予約状況}, verb = \text{表示}, type = SEND \rangle$ と表現する．基本系列については各文に番号が振られているので，その値を idx に当てはめていく．

次にユースケース記述中の文章の実行順序に着目すると，まず最初に事前条件が判定され，その後に基本系列内の動作を順々に実行するという流れになっているため，ノード n_0 はノード n_1 を実行した直後に実行されることがわかる．そこでこれらのノード間にエッジを張ることで実行順を表現する．現時点で得られるグラフは**図 3** の通りである．

上記の動作を基本系列，代替系列及び例外系列に対して実行することで記述全体の制御フローグラフを作成することができる．

得られたグラフの 2 番目のノードに着目すると，“予約状況” を入力として受け取る動作であることがわかる．そこで，グラフを遡ってそれ以前のノードを確認すると 1 番目のノードのみが存在するが，このノードで得られるデータは “列車” となっているため，2 番目のノードを実行する際に必要なはずの “予約状況” が存在しないことがわかる．

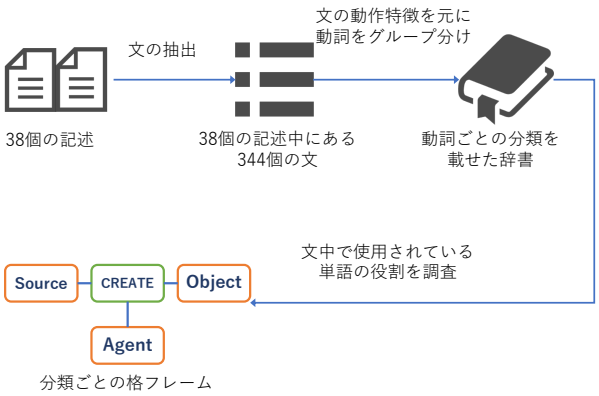

図 4 動詞の分類に関する事前調査

従ってこの記述の基本系列の 1 番では “条件を無視した動作” という臭いが発生していることがわかる．

本節で説明した流れで臭いを検出するためには，入出力データの特定及び制御フローグラフの構築が必要不可欠である．以降の 3 章及び 4 章では，これらの操作について詳細に説明する．

3. 動詞の分類に基づく入出力データの特定

本章では動詞の分類方法及び分類を作成するにあたって行った事前調査について説明する．

3.1 事前調査の概要

動詞の分類に関する事前調査の概要を**図 4** に示す．

本論文では以下の手順で調査をした．

(1) Seki ら [9] の収集した 38 個の記述から，基本系列（Main），代替系列（Alternate）及び例外系列（Error）の各ステップについて動作動詞を特定する．

(2) 得られた動作動詞を，“新たにデータを作っているか”，“入力を持っているか” といった動作の特徴からグループに分けていく．

(3) 2 で得られたグループ同士を比較することで分類を精査し，動詞から分類への対応関係を載せた辞書を作成する．

(4) 各分類に割り当てられた動詞とその動詞が登場する文章内で使用されている単語の役割を調べることで，分類ごとに必要な深層格のリストを作成し格フレームを特定する．

本調査では，ユースケース記述に登場する動詞にはどのような分類が存在するのかを明らかにするとともに，動詞から分類への辞書を作成することを目的とする．

3.2 調査結果

事前調査で得られた分類は以下の通りである．

- CREATE: 既存のデータから新たにデータを作成している動作を表す．

表 1 各分類の格フレーム

分類	動作主格	目的格	対象格	源泉格	道具格
CREATE	○	-	出力	入力	-
DELETE	○	-	入力	-	-
RECEIVE	○	-	出力	-	-
SEND	○	-	入力	-	-
UPDATE	○	-	入力，出力	-	-
DIRECT	○	○	○	-	-
OPERATE	○	-	○	-	○

表 2 各分類に割り当てられた動詞の数

分類	動詞の数
CREATE	11
DELETE	4
RECEIVE	12
SEND	20
UPDATE	11
DIRECT	4
OPERATE	9
合計	71

- DELETE: 既存のデータを削除している動作を表す．
- RECEIVE: ユースケース外のデータを内部に取り込む動作を表す．
- SEND: ユースケース内のデータをユースケース外に送り出す動作を表す．
- UPDATE: データを更新している動作を表す．
- DIRECT: 他のアクタやシステムに動作を指示する動作を表す．
- OPERATE: UI 操作をする動作を表す．

また，事前調査で得られた分類の格フレームを**表 1** に示す．表中で “入力”，“出力” と書かれている箇所がそれぞれの動作における入出力となる深層格，○がつけられている箇所が入出力ではないが分類ごとに必須とされている深層格である．

なお，事前調査で各分類に割り当てられた動詞の数は**表 2** の通りである．本調査の結果得られた 71 種類の動詞から 7 つの分類への対応を，辞書として保存した．

3.3 動詞の分類方法

まず，分類したい動詞を持つ文を形態素解析器に入力し，形態素に分割することで動作動詞を特定する．次に，分類したい動詞が事前調査で得られた辞書に登場しているかどうかを調べる．登場していた場合は，対応する分類に割り当てる．

辞書に対応する動詞がなかった場合，単語の類似度を求めることで一番類似していると判定された分類に割り当てる．それぞれの分類を**表 3** で表す単語とし，その単語と動詞の類似度を求め，一番値が大きかった分類へと割り当てる．

表 3 各分類を表す単語

分類	単語
CREATE	作成
DELETE	削除
RECEIVE	入力
SEND	出力
UPDATE	更新
DIRECT	指示
OPERATE	UI

表 4 表層格から深層格への対応ルール

表層格	ハ句，ガ格	ニ格	ヲ格	カラ格	デ格
深層格	動作主格	目的格	対象格	源泉格	道具格

表 5 例題の文から得られる深層格

深層格	動作主格	目的格	対象格	源泉格	道具格
-	システム	-	アクセシビリティ	-	-

3.4 格フレームに基づく入出力データの抽出

まず，記述中のステップ，事前条件及び事後条件に対し，3.3 節の操作で動詞の分類をする．次に，**表 4** に示すルールを元に，ステップ内の各単語の深層格を表層格から判別する．ここで，各表層格の単語が “、” 及び並列助詞の “と” で接続されていた場合には，それらの語を全て深層格に割り当てる．その後，分類ごとの格フレームと深層格とを照らし合わせることで，動作の入出力となるデータを抽出する．

実際に図 1 の基本系列の 4 番目の文について本手法を用いて深層格を判別した例を**表 5** に示す．この文は “検査する” という動詞が *UPDATE* に分類されるため，対象格となる単語が “アクセシビリティ” な点と表 1 から入力として “アクセシビリティ” を必要とし，出力として “アクセシビリティ” を送り出すはたらきをしていることがわかる．

4. データの依存関係を保持した制御フローグラフの構築及び検査

本章では，データの依存関係を保持した制御フローグラフの性質と，実際に構築したグラフに対する検査手法について説明する．

4.1 本手法で用いるグラフ

本論文で作成するグラフは，ノードにユースケース内でのステップ番号を表す *idx*，*object*（対象格），*source*（源泉格），*agent*（動作主格），*instrument*（道具格），*goal*（目的格）及び文の動作動詞本体である *verb* とその分類である *type* を持つ．また，ステップ番号が連番になっているノード間や分岐が発生しているノードから分岐先のノード，分岐先のノードから基本系列へと合流するノードにエッジを張ることで記述の制御フローを表す．

表 6 グラフに対する検査で検出する不吉な臭い

臭い
条件を無視した動作
事前条件を満たさないフロー
満たされない事後条件
同一役割の複数フロー
事後条件と関係のないフロー

4.2 データ依存の検査

本論文では，Seki ら [9] の定義した不吉な臭いのうち，4.1 節で定義したグラフに対してデータの依存関係を検査することで，**表 6** に示す 5 種類を検出可能にした．

ここでは “条件を無視した動作” と “事後条件と関係のないフロー” の 2 種類について具体的な検出方法を説明する．

条件を無視した動作：

検査式 $\exists p \in$ target 以前のノード.(p から target に入出力パスがある $\wedge$ p と target 間に target の入力データに対する DELETE 動作がない)

指摘内容 検査式が *false* になった場合，*target* が “条件を無視した動作” であると判定される．

補足 入出力パスは表 1 を元に，データを出力しているノードから同じデータを入力としているノードに向けて張られる．図 3 では 1 番目のノードから 2 番目のノードへのエッジが存在するが，これらのノード間では入出力が異なるため入出力パスは存在しない．

事後条件と関係のないフロー：

検査式 $achievedAt \neq \emptyset \wedge (\exists a \in achievedAt.($ target から a に入出力パスがある)

指摘内容 検査式が *false* になった場合，*target* が “事後条件と関係のないフロー” であると判定される．

補足 *achievedAt* は事後条件の達成に関わるノードの集合を表し，本手法では事後条件と同じ object かつ同じ type のノードが要素として割り当てられる．

実際に図 3 を例として本手法で “条件を無視した動作” を検出する流れを説明する．2 番目のノードでは “予約状況” というデータを入力していることから，それ以前のどこかで “予約状況” が作成されていることが期待される．しかし，注目している 2 番目のノード以前のノードは 1 番目のノードのみであり，このノードの出力が “列車” であることからこれら 2 つのノード間には入出力パスが存在しないことがわかる．従って検査式の結果が *false* となるため，2 番目のノードで “条件を無視した動作” が発生していると指摘される．

5. ツール実装

本研究では，2 章から 4 章で説明した手法の適用を自動化したツールを作成した．ツール全体は Ruby スクリプトとして実装されており，Seki ら [9] の定義したメタモデルに従った YAML 形式のユースケース記述を入力することで，表 6 に示す 5 種類の不吉な臭いを検出する．

本ツールでは記述中の文を単語ごとに分割するために MeCab[*1] 及び mecab-ipadic-NEologd[*2] を使用した．また，3 章で作成した辞書に存在しない動詞が登場した場合，Word2Vec 手法 [11] を元に Wikipedia[*3] の全日本語記事から学習したモデルを用いて単語の類似度を計算することで，7 種類のいずれかに割り当てる．

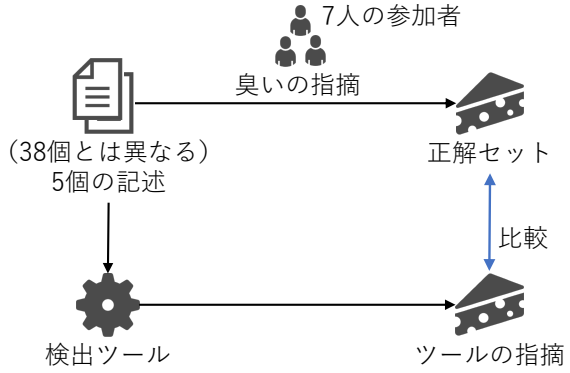

図 5 評価に際して行った調査の概要

6. 評価

本論文では，提案手法を評価するために，以下の RQ を設定した．

RQ 本論文で作成したツールは，実際に表 6 の不吉な臭いを検出する際に有用か．

この RQ は，作成したツールが実際に記述中の不吉な臭いをどの程度の正確さでどの程度漏れなく検出できるかを調べることで，自動検出という目的を達成できたかどうかを評価するために立てられた．

6.1 調査

図 5 に RQ を評価するために行った調査の概要を示す．

例題として，架空の鉄道会社による列車予約システムを取り扱った．本システムに含まれるユースケース記述は 5 つで，それぞれの記述のステップ数は**表 7** の通りである．

まず，正解セットを作成するために 7 人の参加者に 5 つの記述を渡し，表 6 で示した不吉な臭いをその発生箇所とともに記述中から指摘させた．調査に参加した 7 人はいずれも要求工学に対して深い知識があり，ユースケース記述を作成した経験がある．このようにして得られた不吉な臭いとその発生箇所の和集合をとることで正解セットとした．ここで，一人でも指摘していたら正解としたのは，より多

[*1] https://taku910.github.io/mecab/
[*2] https://github.com/neologd/mecab-ipadic-neologd
[*3] https://ja.wikipedia.org/wiki/

表 7 各ユースケース記述のステップ数

記述名	基本フロー	代替フロー	事前条件数	事後条件数	検出した動詞数
アカウントを作る	6	2	1	1	10
ログインする	3	2	2	1	8
列車を検索する	6	2	1	2	11
座席を選択する	4	0	1	1	6
予約する	3	0	1	1	5

表 8 臭いごとの結果

不吉な臭い	*TP*	*FP*	*FN*	*Precision*	*Recall*
条件を無視した動作	1	11	4	0.08	0.2
事前条件を満たさないフロー	0	1	2	0	0
満たされない事後条件	6	0	0	1	1
同一役割の複数フロー	0	0	0	-	-
事後条件と関係のないフロー	9	22	0	0.29	1
合計	16	34	6	0.32	0.72

数の臭いを検出したいという本論文の目的からである．

その後，本手法で作成したツールに対し，正解セット作成に用いたものと同じ 5 つの記述を入力し，ツールによる指摘と正解セットとを比較した．評価基準としては，ツールが指摘したものがどれだけ正解セットで真に臭いであったかを調べるために *Precision* を，正解セットの臭いをどれだけ多く指摘できたかを調べるために *Recall* をそれぞれ算出した．

自動検出ツールが不吉な臭いと判断し，正解セットと同じ臭いが発生していた箇所を *TP*，自動検出ツールが不吉な臭いと判断したが実際には正解セットに含まれていなかった箇所を *FP*，正解セットに入っているがツールは同じ臭いとして検出できていない箇所を *FN* として，式 1，2 により、ツールが指摘した臭いが正しかったかという観点で *Precision* と *Recall* を算出する．

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

6.2 結果

表 6 で示した 5 種類の臭いについて算出した TP，FP，FN の数，*Precision* 及び *Recall* の値を**表 8** に示す．

6.3 考察

表 8 の結果より，全体の *Precision* は 0.32，*Recall* は 0.72 となっていることから，ツール全体の傾向として実際に発生している臭いを検出することについて比較的良い結果が出た一方，誤って不吉な臭いではない箇所を臭いとして指摘しまうことが多いとわかる．

不吉な臭いごとの *Recall* については，“満たされない事後条件” 及び “事後条件と関係のないフロー” の 2 種類については値が 1 となり，全ての臭いを正しく指摘できたことがわかる．一方で “条件を無視した動作”，“事前条件を満たさないフロー” の 2 種類は *Recall* が低いことが明らかとなった．

また，臭いごとの *Precision* については，“満たされない事後条件” は誤検出がなく 1 となった一方で，“条件を無視した動作” 及び “事後条件と関係のないフロー” の 2 種類に関しては表 8 に示すようにそれぞれ *FP* となった個数が 11 個，22 個と誤検出が特に多いという結果になった．

また，同一役割の複数フロー” については正解セットとツールの回答とのどちらにも含まれておらず，*Precision* と *Recall* を算出することができなかった．

6.3.1 *Precision* が十分でない原因の考察

本調査で *Precision* が十分高くならなかった原因として特に大きな原因が，“事後条件と関係のないフロー” の誤検出の多さである．

本来この臭いでわかることは，“事後条件がおかしいか基本系列や事前条件などがおかしい” ということである．しかし，本手法では事後条件を満たす際に関係してこないステップが存在した場合そのステップが間違っているとして指摘をしており，事後条件は正しいという前提の元で指摘をしてしまっていた．その結果，図 1 に示す記述では事後条件である “予約確認画面に進む” という条件が明らかに誤りであるのに，ツールは事後条件が誤っている可能性を指摘できず基本系列が誤っているとして誤検出をしてしまった．

また，正解セット作成時に調査参加者に指摘してもらう臭いのリストを渡したが，その際も事後条件がおかしい場合はこの臭いではないという誤解を与えてしまった結果，正解セットでも事後条件が “事後条件と関係のないフロー” として指摘されないという結果になってしまった．実際に調査参加者は図 1 に対して事後条件がおかしいことを “満たされない事後条件” として指摘していることから，誤解を与えていなかった場合は “事後条件と関係のないフロー”

表 9 想定される *Precision*

	TP	*FP*	*Precision*
事後条件と関係のないフロー	31	0	1
合計	38	12	0.76

について正しく指摘できていたと考えられる．

事後条件が誤っている可能性をツールが指摘しており，正解セット作成時に参加者が正しく事後条件でこの臭いが発生していることを指摘できたという仮定の元で *Precision* を算出すると，**表 9** に示す結果が得られる．ここで，正解セット中で “満たされない事後条件” として検出されている事後条件を全て事後条件がおかしいという指摘とみなし，“事後条件と関係のないフロー” でも同様に指摘されるであろうという仮定の元で算出した．

6.4 妥当性の脅威

内的妥当性については，6.1 節における正解セット作成に携わった調査参加者の判断基準の相違と，時間経過や他参加者との比較による判断基準の変化が脅威となる．

しかし，正解セット作成において時間制限を設けず，不吉な臭いの指摘順序も指定をせず何度でも他の記述を参考に指摘をし直してよいとしたので，時間経過による判断基準の変化による脅威は小さいと考えられる．また，正解セット作成作業は他参加者とやりとりを行わず，各個人がそれぞれ一人で判断するようにしたため，他参加者との比較による脅威も取り除かれている．参加者ごとの判断基準の相違についても，参加者全員が要求工学に関する一定の知識を有しているため脅威は軽減されていると考えられる．

一方で，参加者には臭いの説明をして対象の臭いを検出させたが，“事後条件と関係のないフロー” の例にあるように臭いの説明が誤っていたために正解セットの信頼性が低くなってしまった箇所がある．

外的妥当性については，正解セット作成に用いた記述数やステップ数が制限されているという点があげられる．本調査では 5 つの記述のみに対して調査しため，他の記述に対しても同様な結果になるとは限らない．そのため，より多くの記述に対して調査する必要がある．

また，本調査で対象としたのは，実システムに対するユースケース記述ではなく架空のシステムに対する記述であったため，実システムにおいても同様の結果が得られるとは限らない．そのため，実システムに対しての調査も今後の課題となっている．

7. 関連研究

7.1 ツールを用いたユースケース記述の低品質箇所検出手法

ユースケース記述における低品質箇所の自動検出についてはいくつかの試みがある．

Liu ら [7] は，Torner ら [6] によって提供されたリストのうち影響度合いが特に大きいとされる 4 種類の欠陥を形式的なものに再定義し，ユースケース記述に対して構文解析することでイベントなどの構成要素を抽出し，それに対して述語を適用することで自動検出を可能にした．しかし，この研究で検出できるようになった欠陥は 12 種類の欠陥のうち 4 種類に留まり，十分な範囲を対応しているとは言えない．

また，El-Attar ら [4] はユースケースモデルの Anti Pattern を 8 種類定義し，ユースケースモデルのメタモデルに対して OCL（Object Constraint Language）で記述された規則を適用することでそれらの自動検出を試みた．しかし，記述の内容に基づく Anti Pattern は人の手によるレビューが必要不可欠とされており，検出の自動化はなされていない．

Seki ら [9] はユースケース記述の低品質と思われる箇所を “不吉な臭い” として定義し，得られた 60 種類の不吉な臭いのうち 24 種類に対して検出を自動化した．しかし，自動化した 24 種類の不吉な臭いは，いずれも文の長さやアクタの一覧の有無といった記述の見た目から判断できるもので，データの依存関係などは考慮されていない．

7.2 ユースケース記述からのモデル抽出手法

自然言語で記述されたユースケース記述から形式的なモデルを抽出する研究として，Sarmiento らの研究 [12] や Erazo らの研究 [8] などがあげられる．

Sarmiento らは，一定の規則に従って自然言語で記述されたユースケース記述から Petri-Net を作成し，それに対して到達性分析を行うことで記述中で発生しているデッドロックなどの検査を試みた．この研究では実際に 2 種類の欠陥を Petri-Net に対する検査で発見できたとしているが，Petri-Net を用いて検出すべき欠陥については正解セットの作成が困難であるとしており，ツールの出力との比較は行われていない．

Erazo らは，純粋な自然言語ではなくテンプレートに従って記述されたユースケース記述から状態遷移モデルを抽出するツールを作成し，手動で作成した状態遷移モデルとツールの出力とを比較することでアプローチの有用性を示した．この研究では実際にモデルを用いた欠陥の検出までは行っておらず，状態遷移モデルを抽出するにとどまっているが，モデルを利用することで既存のモデルチェッカを用いた検査などができるようになるとしている．

8. おわりに

8.1 結論

本論文では，ユースケース記述の不吉な臭いのうちデータの依存関係に矛盾を生じているものについて，データの依存関係を保持したグラフを構築してそれに対して検査す

るツールを開発することで検出を自動化した．また，開発したツールをユースケース記述を用いて評価することで，全体の *Recall* が 0.72，一部の臭いについては *Recall* が 1 となっていることを確認し，実際に不吉な臭いである箇所を検出する際に特に有用であることを示した．

一方で，一部の臭いについては *Precision* が特に低くなってしまう，正解セットに含まれていない臭いがある，実システムに対する調査をしていないなど今後の課題もいくつか残されている．

8.2 今後の課題

8.2.1 辞書を用いた深層格の判別

本論文では，格フレームに基づいて文中の単語の深層格を判別する際，表層格を元に決定している．しかし，表層格と深層格は必ずしも対応せず，本手法では全ての記述に対して正確に深層格を判別できるわけではない．

深層格の判別については，個々の動詞がとりうる格フレームのパターンをまとめた格フレーム辞書を組み合わせることで，より正確な判別が可能になりうる．

8.2.2 不吉な臭いの再調査

本論文における調査では “事後条件と関係のないフロー” と “満たされない事後条件” をそれぞれ異なる不吉な臭いとしたが，実際には “事後条件と関係のないフロー” は事後条件が誤っている場合にも指摘されるべきという結果から，“事後条件と関係のないフロー” は “満たされない事後条件” を完全に含んでいる可能性が考えられる．そこで，実際にこれらの臭いは完全な包含関係にあるのか，また他の臭いについてもこのような例はないのか，カタログの調整も含めて再度調査する必要性がある．

8.2.3 実システムへの適用

6.4 節でも触れたとおり，本論文の評価実験では架空の鉄道会社による列車予約システムを例題として扱っており，実際のシステムに対しての適用はなされていない．実際に本論文で開発したツールが幅広い記述に対して有用であるかどうかを調べるためには，複数の実システムに対してツールを適用することが必要であると考えられる．

謝辞 本研究の一部は，日本学術振興会科学研究費補助金 (JP18K11237) の助成を受けた．

参考文献

[1] 大西 淳，郷健太郎：要求工学，共立出版 (2002).
[2] Geri, S. and Jason, W.: *Applying Use Cases: A Practical Guide*, Addison Wesley Longman (1998).
[3] A. Issa, A.: Utilising Refactoring To Restructure Use-Case Models, *Proc. World Congress on Engineering*, Lecture Notes in Engineering and Computer Science, Vol. 2165, pp. 523–527 (2007).
[4] El-Attar, M. and Miller, J.: Improving the quality of use case models using antipatterns, *Software & Systems Modeling*, Vol. 9, No. 2, pp. 141–160 (2010).
[5] Phalp, K. T., Vincent, J. and Cox, K.: Assessing the quality of use case descriptions, *Software Quality Journal*, Vol. 15, No. 1, pp. 69–97 (2007).
[6] Tørner, F., Ivarsson, M., Pettersson, F. and Öhman, P.: Defects in Automotive Use Cases, *Proc. ACM/IEEE International Symposium on Empirical Software Engineering*, pp. 115–123 (2006).
[7] Liu, S., Sun, J., Liu, Y., Zhang, Y., Wadhwa, B., Dong, J. S. and Wang, X.: Automatic Early Defects Detection in Use Case Documents, *Proc. 29th ACM/IEEE International Conference on Automated Software Engineering*, pp. 785–790 (2014).
[8] Erazo, L., Martins, E. and Greghi, J. G.: MARITACA: From textual use case descriptions to behavior models, *Proc. 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops*, IEEE, pp. 83–90 (2017).
[9] Seki, Y., Hayashi, S. and Saeki, M.: Detecting Bad Smells in Use Case Descriptions, *Proc. 27th IEEE International Requirements Engineering Conference*, pp. 98–108 (2019).
[10] Sinha, A., Sutton Jr, S. M. and Paradkar, A.: Text2Test: Automated inspection of natural language use cases, *Proc. 3rd International Conference on Software Testing, Verification and Validation*, pp. 155–164 (2010).
[11] Mikolov, T., Chen, K., Corrado, G. and Dean, J.: Efficient estimation of word representations in vector space, *arXiv preprint arXiv:1301.3781* (2013).
[12] Sarmiento-Calisaya, E., Cárdenas, E. H., Cornejo-Aparicio, V. and Alzamora, G. S.: Towards the improvement of natural language requirements descriptions: the C&L tool, *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, pp. 1405–1413 (2020).