変更における状態を含むテスト網羅尺度とテストケース抽出法の提案

湯本剛
筑波大学大学院
tsuyoshi.yumoto@gmail.com

松尾谷徹
デバッグ工学研究所
matsuodani@debugeng.com

津田和彦
筑波大学
tsuda@gssm.otsuka.tsukuba.ac.jp

要旨

ソフトウェアの一部に変更を加えた場合，その変更の波及を探る変更波及解析（*Change Impact Analysis*）は実務上の大きな課題である．産業界において，変更にかかる活動は新規開発よりも大きな割合を占めている．ソフトウェアの多様化，複雑化，再利用範囲の増大などから変更波及の範囲が拡大し，かつ安易な変更による弊害など課題が山積している．本論文では，変更波及のうち，状態遷移を持つソフトウェアにおいて，変更波及がデータベースや外部変数などの保持データを介して生ずる場合のテストについて取り上げ，テスト網羅基準とテスト設計手法を提案する．具体的な例を使って手法の適用可能性を確認し，テストケースの数を他の手法と比較し合理的であることを示す．

1. はじめに

ソフトウェアの一部に変更を加えた場合，その変更の波及を探る変更波及解析（Change Impact Analysis）は，実務上の大きな課題である．産業界において変更にかかる活動は，新規開発よりも大きな割合を占めている．ゼロから新規にソフトウェアを開発するケースは稀であり，何らかの流用を基に変更を加える開発が主流となっている．開発方法においてもアジャイルが主流となり，変更の積み重ねによって開発が行われている．

しかし，変更波及を合理的に制御する技術は，ソフトウェア工学とって未完成の分野である [1]．変更の背景は，時代と共に課題を難しくしている．ソフトウェアの多様化と複雑化，再利用範囲の増大などから変更波及の範囲が拡大し，かつ安易な変更による弊害など課題が山積している．これらの課題に対してソフトウェア工学は十分な解を提供できていない状況にある [2]．

本論文では，状態遷移を持つソフトウェアにおいて，変更波及がデータベースや外部変数などの保持データを介して生ずる場合のテストについて考える．課題の一つは，網羅基準である. データフローテストの全使用法（AU 法）[3] を基にして, 変更波及のテスト網羅基準を波及全使用法（IDAU:Impact Data All Used）として提案する．もう一つの課題は，IDAU 法を満たす具体的な変更波及のテストケースの抽出方法である. 変更波及のテストの設計手法として, 順序組合せテストを提案する. 最後に，ここで提案する IDAU 法のコストの評価、すなわちテストケースの数を従来技法である状態遷移テストの S1 網羅基準と比較をして考察を行い, 提案する方法が合理的であることを示す．

2. 変更波及とその解析

本節では，提案する網羅基準やテスト設計手法の前提となる，システムの構成と変更について定義し，変更波及テストの課題について詳細を示す．

2.1. 対象とするアプリケーションの構成

一般的にソフトウェアは，入力 In に対して，何らかの出力 Out を返す．ソフトウェアの機能は，何らかの入

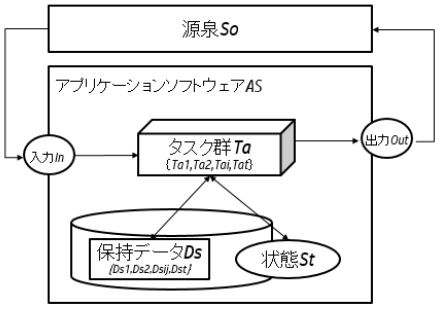

図 1: アプリケーションソフトウェアの構成

力を出力に変換する処理により実現されていると考えられる．この処理を本論文ではタスクと呼ぶ．タスクは，該当のテストレベルからみた入力を出力に変換している 1 つの処理である．そのため，タスクの粒度は，テストレベルによって決まる．ソフトウェアの構成要素であるタスク Ta の出力について考えると，Ta への入力 In だけでなく状態 St と保持データ（データベースや内部メモリに保存されているデータ）Ds の影響を受けると考えられる．例えば，Web アプリケーションにて予約を行うタスクについて考えると，予約が可能か否かを示す状態と，予約オブジェクトの予約状況を示す保持データによって，予約の成否が決まる．

本論文では，対象として図 1 に示すような状態と保持データを持つアプリケーションソフトウェア (AS) について考える．AS の構成要素は，タスク群 Ta と状態 St と保持データ Ds とし，各タスクは外部の源泉 So からの入力 In と So への出力 Out があるとする．タスク群 Ta は，その要素を $Ta = \{Ta_1, Ta_2, \cdots, Ta_i, \cdots, Ta_t\}$ とし，変更タスクを Ta_i とする. 対応する入出力は In_i と Out_i とする．

2.2. 変更と変更波及

AS に対して，何らかの変更を加える場合について考える．変更には，なんらかの意図がある．AS が持つ機能の変更であったり，不具合に対する変更や，性能や保守性の改善のためのリファクタリングであったりする．本論文では，変更の意図については取り扱わず，AS の構成要素（タスク，状態，保持データ）に対する具体的な変更について考える．ただし，テスト実行をするためには，タスクを動かすことが必須となる．そのため，以降の議論はタスクに焦点を絞る．

ひとつの変更 Q について考える．変更 Q は，タスク群 Ta のあるタスク Ta_i に対して行われたとする．変更 Q は，コードの削除や追加を含み，その結果 Ta_i の版 R が $R+1$ に変更される．この変更の結果を Ta_i^R から Ta_i^{R+1} とする．

変更 Q の波及には，3 つのケースが考えられる．

1. 変更波及が無い場合．
2. 変更波及が他のタスクへ波及しない場合．
3. 変更波及が他のタスクへ波及する場合．

3. の変更波及は，タスク間の参照が図 2 に示すように状態と保持データに限られるならば，該当する状態や保持データの参照を介した範囲が限られると考えられる．本論文では，この考え方から波及を受けるタスクを特定し，その合理的なテスト設計について論じる．

変更波及，あるいはその解析に関する研究は古くから行われている．プロダクトラインや UML 図面群を基に依存関係生成モデルを用いて波及解析を行う研究がある [4, 5, 6]．タスク内のデータフローを基に変更波及を詳細に解析した研究がある [7]．データベースなど保有データを基に変更波及解析を行う研究も行われている [8, 9]．一方，状態と状態遷移はマルコフ過程として実装されているので，過去の状態が未来の状態に影響しない．状態遷移に関する波及解析の研究は見当たらないのはそのためだと推測する．

2.3. 状態と変更波及のテスト

変更波及は，保持データを介して変更タスクから他のタスクへ伝達する．状態や状態遷移自身は変更波及に関

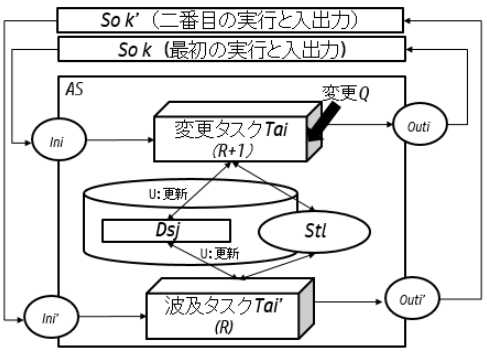

図 2: 変更タスクと変更波及

与しないが，変更波及のテストにおいて与えるデータの順序において状態を考慮する必要が生じる．状態 St において, 考慮する必要がある状態を St_l とする (図 2).

保持データの構成について考える．その要素を $Ds = \{Ds_1, Ds_2, \cdots, Ds_j, \cdots, Ds_t\}$ とし，変更波及を受ける保持データの要素を Ds_j とする．保持データに対する操作は，データのライフサイクルである「C : 生成」「R : 参照」「U : 更新」「D : 削除」を記した CRUD 図で定義する．Ds_j を介して変更波及が生ずるのは，変更タスク Ta_i において「C : 生成」あるいは「U:更新」が行われ，他のタスクでその保持データを利用した場合である．これらのタスクを波及タスクと呼ぶ.

保持データのライフサイクル上の「C : 生成」「R : 参照」「U : 更新」「D : 削除」などの操作は，無条件に行われるのではなく，操作するタスクの制御フローに沿って行われる．制御フローは，図 2 に示すとおり, 2 階層として捉えることができる．上位の制御フローはタスクの実行順序により決定する大きな制御の流れに相当する．個々のタスク内の制御フローが下位にあたる．個々のデータ参照実行文はその制御フロー上の条件文で実行が決定される．条件にはタスクへの入力，保持データ，状態が含まれる．

変更波及を確認するには，変更が波及した保持データを参照するデータフローに沿ってテストを設計することになる．このデータフローを決定するのは，関係するタスクの実行順序とタスク内の制御フローであり，その制御フローを決定する際に状態が影響する．よって，実際のテスト実行においては状態を考慮する必要が生ずる．

2.4. 変更波及テストの網羅基準

テストにおける網羅基準については，その強度を含め制御フローとデータフローの観点から研究が行われ体系が作られている [3]．最も弱い網羅基準は制御フローのみに着目した実行文網羅，次が分岐網羅であり，最も強い網羅基準はデータフローを含めた全パス網羅（All Paths）である．全パステストは，すべての分岐の積であり現実的には実現不可能のため，全使用法（All Uses）が推奨されている [3].

変更波及をテストする場合，波及に関与するデータに着目し，そのデータフローテストを行う．一般的な全使用法は「データを定義したすべての場所から始まり，データを使用するすべての場所に至るまでのパスセグメントを最低限 1 つを含むテストケース」と定義されている [3]．この定義を変更タスクと波及タスクとの関係に置き換え「変更タスクにおいてデータの生成および更新があるデータを使用するすべてのタスク（波及タスク）を 2 つのタスクを実行するまでに経由するルートにかかわらず最低限 1 つ含むテストケース」とし，波及全使用法（IDAU:Impact Data All Used）とする．

3. 順序組合せによるテストケース抽出法

本節では，図 2 で示した変更タスクと波及タスクの組合せを抽出する手法として順序組合せテストを提案する．この手法で抽出するテストケースには，必ず変更タスクを実行した後に波及タスクを実行する，といったように組み合わせるタスクに実行順序があるため，順序組合せと命名した．

3.1. 提案手法に必要となる入力情報

一般的にテストケース抽出のために必要な入力情報をテストベースと呼ぶ [10]．提案手法に必要なテストベースは DFD，ER 図，CRUD 図である．以下，DFD，ER 図，CRUD 図を簡潔に説明する．

- DFD（データフローダイアグラム）

  DFD はシステムにおけるデータの流れを表現した有向グラフであり，要求分析において用いられている．DFD はデータ指向設計の要として用いられ，オブジェクト指向設計においても抽象化する前段階として実践の場で用いられている．

  DFD は，最上位のコンテキストレベルから階層として詳細化され，各階層は 1 枚以上の DFD から成る [11]．テストベースとして用いる場合，テストの範囲は DFD で与えられるとする．DFD の階層はテストレベルに対応し，階層が下がると単体テストとなり，上がると統合テストとなる．

  DFD はノードとエッジからなる．ノードは 3 種類の要素である N 個のタスク（プロセス）Ta と，M 個の保持データ（データストア）Ds と，L 個の源泉（外部エンティティ）So から構成されている．3 種類の要素を一意に特定する際は Ta_i，Ds_j，So_k と表記する．

エッジは，ノードからノードへのつながりを有向線分で表記している．エッジはデータの流れを表しており，制御の流れは表していない．エッジの特定は，起点ノードと終点ノードを用いて行う．ある特定のタスクからデータストアへの入力がある場合のエッジの特定は，Ta_i/Ds_j となり，源泉から出力してタスクで処理をする場合は，So_k/Ta_i と表す．

- ER 図

  ER 図はシステムにおけるエンティティ間の関係を示す図であり，UML のクラス図に対応している．DFD では表現できないエンティティの詳細化やエンティティ間の関係について示しており，DFD と共に用いられている．

  ここでは，DFD のデータストア Ds_j が持つエンティティと，CRUD 図の対応から，後述する拡張 CRUD 図を作成するために用いる．よって，テストベースとしては，システムすべての ER 図を必要とするものではない．

- CRUD 図

  CRUD 図とは，タスク Ta_i からデータストア Ds_j への C：生成，U：更新，R：参照，Ds：削除の操作を表した図である [12]．CRUD 図から，DFD と ER 図では表現されていないタスクのエンティティへの操作を知ることができる．

  本論文では，タスクがデータストアに対して行う操作を特定するために CRUD 図を用いる．タスク Ta_i のデータストア Ds_j に対する操作が U：更新であればタスクによる操作はエッジを介した操作として Ta_i/Ds_jU と表記する．ただし，タスクが操作するデータストアが 1 つだけの場合は，Ta_iU といった省略した表記を使う．

3.2 順序組合せテストの概要

提案する手法は，2 タスク間の順序組合せを対象とする．2 タスク間の順序組合せの抽出は以下のルールを適用する．

- ルール 1：変更タスクの特定

  対象とする DFD 内の変更タスクのうち，データストア Ds へ出力エッジを持つタスクを選択し，順序組合せの変更タスク群 $P\{Ta\}$ とする．変更タスク群からの出力するデータストア群を $P\{Ds\}$ とする．

- ルール 2：波及タスクの特定

  ルール 1 で求めた $P\{Ds\}$ からの入力エッジを持つタスクを波及タスク群 $S\{Ta\}$ として特定する．

- ルール 3：順序組合せテストケースの抽出

  拡張 CRUD 図を基に変更タスク群 $P\{Ta\}$ とそのデータストア群 $P\{Ds\}$ を介する波及タスク群 $S\{Ta\}$ を組合せ，順序組合せのテストケースとする．

以降からは，順序組合せを抽出してテストケースとするまでの実施手順を詳細に説明する．

表 1: 拡張 CRUD 図

タスク	データストア			源泉		
	Ds_1	...	Ds_j	So_1	...	So_k
Ta_1						
...						
Ta_i						

3.3 ルール 1：変更タスクの特定

ルール 1 を用いて変更タスクとそのデータストアを特定し，拡張 CRUD 図の変更タスク部分を作成する．

拡張 CRUD 図とは，テストベースとして与えられた DFD，ER 図，CRUD 図から $P\{Ta\}$ の各 Ta_i と関連する So_k，そして $P\{Ds\}$ となる Ds_j の関係を追加して作成したものである．表 1 に拡張 CRUD 図の表記を示す．拡張 CRUD 図のデータストアに対する情報は C，U，R，D のいづれか，または組合せか空白である．源泉に対する情報は In か Out，または組合せか空白である．空白は関係が無いことを示す．

1. 源泉からの入力エッジを持つ変更タスクの特定

   テストケースは，外部からのテスト対象への入力から，外部への出力結果を確認するものであるため，テスト入力とテスト結果のペアで構成されている．そこで，テスト対象範囲の外からの入力，即ち So_k からの入力エッジを持つ Ta_i を見つける必要がある．この特性を持ったタスクのうち，さらに変更の

表 2: 中間の拡張 CRUD 図の例

タスク	データストア			源泉		
	Ds_1	Ds_2	Ds_3	So_1	So_2	So_3
$Ta_1[St_1]$	CU			In		
$Ta_3[St_1]$		C		In		

あるタスク群を変更タスクの集合となる $P\{Ta\}$ 候補とする．変更が特定の状態でのみ起こり得る場合は，その状態を［St_l］と記載する．

2. データストアへの出力エッジを持つタスク特定

   Ta_i から Ds_j への出力エッジは，C か U か D の操作を行うことを意味する．CRUD 図から該当する出力エッジを持つ Ta_i を選択する．$P\{Ts\}$ 候補の中から，該当する Ta_i を選び，変更タスク群 $P\{Ta\}$ を確定する．

3. 中間の拡張 CRUD 図作成

   拡張 CRUD 図には，変更タスク群 $P\{Ta\}$ に該当する So_k から Ta_i への入力 (In)，もしくは Ta_i から So_k への出力 (Out) の情報を付加する．特定した Ta_i に対して，入力となる So_k に In を記入し，Ds_j については CRUD 図を参照して C か U か D かその組合せかを記入する．中間の拡張 CRUD 図として例示した表 2 では，3 つの源泉 $\{So_1, So_2, So_3\}$ と 3 つのデータストア $\{Ds_1, Ds_2, Ds_3\}$ があり，2 つのタスク $\{Ta_1[St_1], Ta_3[St_1]\}$ が変更タスクである．この段階で作成する拡張 CRUD 図は，作業途中のものである．

表 3: タスク間のデータ共有の組合せパターン

		$P\{Ta\}$			
		C	R	U	D
$S\{Ta\}$	C	×	×	×	○
	R	○	-	○	-
	U	○	-	○	×
	D	○	-	○	×

表 4: 完成した拡張 CRUD 図の例

タスク	データストア			源泉		
	Ds_1	Ds_2	Ds_3	So_1	So_2	So_3
$Ta_1[St_1]$	CU			In		
$Ta_3[St_1]$		C		In		
$Ta_2[St_1]$	R				Out	
$Ta_5[St_1]$		RU				Out

3.4 ルール 2：波及タスクの特定

ルール 2 を用いて波及タスク群を特定し，拡張 CRUD 図へ波及タスク部分を追加し図を完成させる．

1. データストアを介した波及タスク特定

   先に作成した中間の拡張図から変更タスクの操作が C か U か D であるデータストアに着目する．着目したデータストアに対してエッジを持つタスクが波及タスクの候補となる．波及タスクとして選択するタスクは表 3 に示す表の”○”の組合せに該当するタスクである．波及タスクは， C : 生成，U : 更新，D : 削除を選択する．”-”をつけた組合せは，データストアを介した影響が生じないため，組合せテストの対象としない．”×”をつけた組合せは仕様上有り得ない組合せであり，ありえないことの確認は，順序組合せを網羅しなくともテストできるため，組合せテストの対象としない．

2. 拡張 CRUD 図の完成

   波及タスク候補のうち，源泉に出力エッジを持つタスクを波及タスクとして特定する．波及タスクの特性を DFD より読み取り，特定する．特定した波及タスクを拡張 CRUD 図に追記し完成させる．

   完成させた拡張 CRUD 図の例を表 4 に示す．この例では，データストア Ds_1 から源泉 So_2 への流れをタスク $Ta_2[St_1]$ が行い，データストア Ds_2 から源泉 So_3 への流れをタスク $Ta_5[St_1]$ が行っていることを示している．

3.5 ルール 3：順序組合せテストケースの抽出

拡張 CRUD 図を基に順序組合せのテストケースを抽出し，テストケース表を作成する．

表 5: 順序組合せテストによる論理的テストケースの例

No	論理的テストケース	順序組合せ
1	概要	$Ta_1C \rightarrow Ta_2R$
2	概要	$Ta_1U \rightarrow Ta_2R$
3	概要	$Ta_3C \rightarrow Ta_2R$
4	概要	$Ta_3C \rightarrow Ta_2U$

1. 変更タスクと波及タスクの組合せを抽出

   拡張 CRUD 図から変更タスクを選ぶ．先に作成した拡張 CRUD 図の例（表 4 を参照）であれば，$Ta_1[St_1], Ta_3[St_1]$ である．次に変更タスクが操作しているデータストアと，それを操作している波及タスクを対応付ける．例では，$Ta_1[St_1] \xrightarrow[Ds_1]{} Ta_2[St_1]$ と $Ta_3[St_1] \xrightarrow[Ds_2]{} Ta_5[St_1]$ である．

2. データストアに対する操作の組合せ

   操作の組合せとは変更タスクと波及タスクの操作の組合せである．表 4 の例であれば，変更タスクのデータストアに対する操作である Ta_1 は，Ds_1 に対して C と U の操作を行っている．波及タスク Ta_2 の操作は R である．組合せは $C \rightarrow R$ と $U \rightarrow R$ となる．変更タスクと波及タスク間に介在するデータストアが 1 つであれば $\xrightarrow[Ds_1]{}$ を省略して $\rightarrow$ で表してもよい．また変更の発生条件となる状態が 1 つであれは，$[St_1]$ を省略してもよい．表 4 の例における全組合せは，$Ta_1C \rightarrow Ta_2R$，$Ta_1U \rightarrow Ta_2R$，$Ta_3C \rightarrow Ta_2R$，$Ta_3C \rightarrow Ta_2U$ の 4 個である．

3. テストケース表の完成

   変更タスクと波及タスクの操作の組合せをテストケースとしてまとめる．表 5 にその例を示す．概要の部分は，当該組合せが持つ入力の条件や出力の特性を仕様から抜き出して記載する．

以上の手順で，順序組合せテストに必要なテストケースを抽出する．ここで用いたテストケースとは，ISTQB の定義による論理的テストケースに相当する [10]．具体的な値や期待結果，該当の処理までの状態を遷移させていく手順まで定義した記述を具体的テストケースと呼ぶが，本論文では扱わない．

4 順序組合せテストの適用評価

本節では，旅行代理店向けフライト予約システムの仕様を用いて，3 節で述べた実施手順を適用し，順序組合せが抽出できることを確認する．

4.1 題材の概要

フライト予約システムの概要を以下に示す．

＜フライト予約システム概要＞
・旅行代理店用に開発したフライト予約サーバにインターネット経由でアクセスできる専用のクライアントアプリケーション．
・旅行代理店の窓口での利用を想定しており，ユーザ認証されたユーザのみ利用可能である．
・旅行代理店の窓口数（クライアント数）は 50 としており，同時に予約処理を行うことができる．
・旅行代理店にて取り扱う全ての航空会社の飛行機の予約が可能である．
・本システムは，フライト予約サーバを仲介して複数の航空会社のシステムと同期をする．
・チケット情報や残チケット数は同期することで最新に更新される．
・フライトの新規予約、予約内容の更新，削除が可能である．更新と削除は新規予約したユーザのみ可能である．
・以下はシステム範囲外
 –チケット代金の決済（別システムと連携して行うため）．
 –マスタ情報設定（他システムとの共用マスタ設定アプリケーションがあるため）．

図 3: フライト予約システムの概要

題材となるフライト予約システムの仕様は，本研究の一環として評価実験の際に題材として使っているものである [13]．テスト対象の分析と，テストケース設計に関する用語は，国際標準である ISO/IEC/IEEE29119 の定義に従い，テスト対象の論理的なサブセットを機能セット（Feature set）と呼ぶ [14]．本論文では，表 6 の新規フライト予約を変更が入った機能セットとする．(表 6 の最後の列にある○が変更を表す.）新規フライト予約からテストケースを抽出するための前提として用意した仕様は，新規フライト予約に関連する DFD（図 4），ER 図（図 5），CRUD 図（表 7）とする．DFD に含まれるタスク数 N は 6，データストア数 M は 2，源泉数 L は 4 である．

4.2 ルール 1：変更タスクの特定

テストベースである DFD に含まれるタスク数 N は 6 であるが，変更が入った新規フライト予約の変更タスクは，表 7 の CRUD 図を確認するとフライト検索 Ta_1 とフライト予約 Ta_2 であることがわかる．図 4 から，Ta_1 と Ta_2 の外部入力を確認する．Ta_1 は，顧客 So_1 から

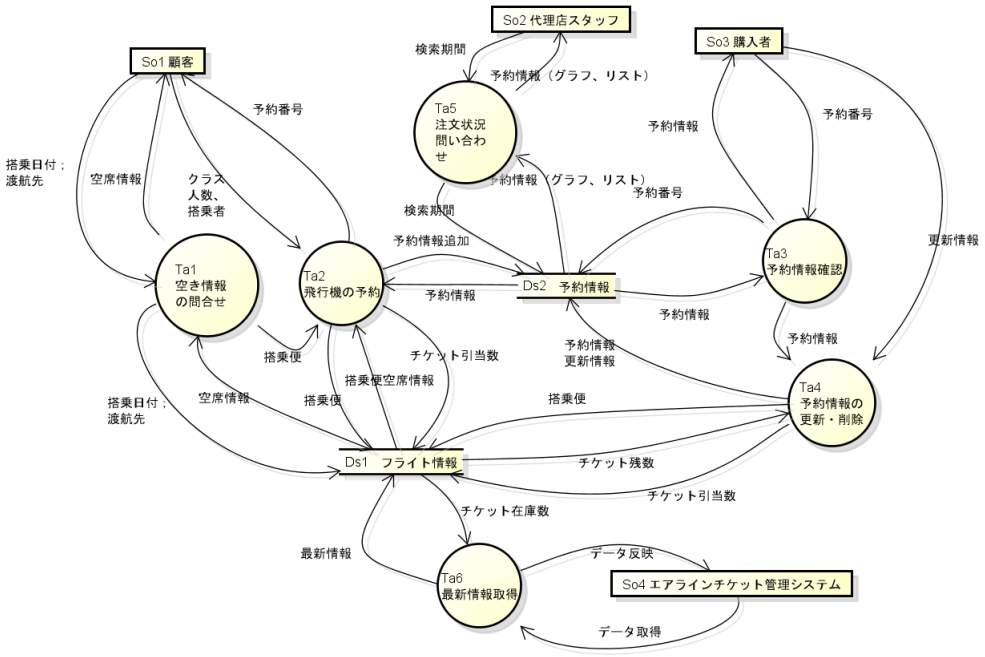

図 4: 新規フライト予約の DFD（一部分）

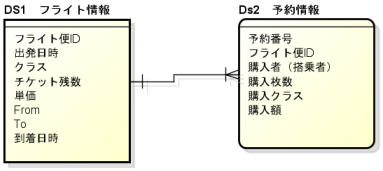

図 5: 新規フライト予約の ER 図（一部分）

搭乗日付と渡航先を外部入力し，Ta_2 は，顧客 So_1 から搭乗便，クラス，人数，搭乗者を外部入力している．

続いて，Ta_1 と Ta_2 の内部出力を確認する．Ta_1 はフライト情報 Ds_1 に対して検索条件を与えているのみで内部入力はしていないため，変更タスク群 $P\{Ta\}$ からは除外する．Ta_2 がフライト情報 Ds_1 で U，予約情報 Ds_2 で C を行っていることが表 7 から読み取れる．これらから，拡張 CRUD 図（表 8）を作る．表 8 から，ルール 1 に適合する Ta_2/Ds_1U，Ta_2/Ds_2C を特定できる．

表 6: フライト予約システムの機能セット一覧

テストアイテム	機能セット	変更
フライト予約システム	メニュー	
	ログイン	
	新規フライト予約	○
	予約変更 & キャンセル	
	予約一覧	
	予約グラフ	
	同期処理&	

4.3 ルール 2：波及タスクの特定

ルール 2 にて波及タスク群 $S\{Ta\}$ を抽出するために，タスクの外部出力を図 4 の DFD から調べる．$P\{Ds\}$ に含まれる Ds_1 と Ds_2 にエッジを持ち，かつ So へ出力するタスク群が $S\{Ta\}$ 候補である．図 4 では，全てのタスクが Ds_1 および Ds_2 からのエッジを持つ．しかし，So への出力に着目すると，Ta_4 は該当するエッジがな

表 7: 新規フライト予約の CRUD 図（一部分）

機能セット	タスク		エンティティ	
			Ds_1 フライト情報	Ds_2 予約情報
新規フライト予約	Ta_1	フライト検索	R	
	Ta_2	フライト登録	RU	C
予約変更	Ta_3	予約情報確認		R
キャンセル	Ta_4	予約情報修正	RU	UD
予約リスト予約グラフ	Ta_5	注文状況		R
同期処理	Ta_6	最新情報取得	CU	

表 8: 新規フライト予約の中間拡張 CRUD 図

タスク	データストア		源泉			
	Ds_1	Ds_2	So_1	So_2	So_3	So_4
Ta_1						
Ta_2	U	C	In			

いため，$S\{Ta\}$ 候補には入らない．

$S\{Ta\}$ 候補のうち、表 3 の ”○” がつく組合せに相当する Ta_i が，ルール 2 で特定したタスクとなる．本節の例の場合，$P\{Ta\}$ での操作は，C と U であるため，$S\{Ta\}$ 候補の中で C の操作をする Ta_i 以外は全てルール 2 で特定したタスクとなる．

これらに該当する Ta_i と Ds への CRUD 操作，そして So への Out を追記し，表 9 を完成させる．

4.4 ルール 3：手順 順序組合せテストケースの抽出

表 9 の拡張 CRUD 図から変更タスクと波及タスクの組合せを抽出する．抽出した変更タスクと波及タスクの組合せに対して，データストアに対する操作を明記したものは以下のとおりとなる．

- $Ta_2/Ds_1U \xrightarrow[Ds_1]{} Ta_1R$
- $Ta_2/Ds_1U \xrightarrow[Ds_1]{} Ta_2/Ds_1U$

表 9: 新規フライト予約の拡張 CRUD 図

タスク	データストア		源泉			
	Ds_1	Ds_2	So_1	So_2	So_3	So_4
Ta_1	R		Out			
Ta_2	RU	C	$InOut$			
Ta_3		R			Out	
Ta_5		R		Out		
Ta_6	CU					Out

- $Ta_2/Ds_1U \xrightarrow[Ds_1]{} Ta_6U$
- $Ta_2/Ds_2C \xrightarrow[Ds_2]{} Ta_3R$
- $Ta_2/Ds_2C \xrightarrow[Ds_2]{} Ta_5R$

これらの変更タスクと波及タスクの操作の順序組合せがテストケースとなる．抽出した順序組合せが持つ入力の条件や出力の特性を仕様から抜き出して論理的テストケースとしてまとめる．表 10 に論理的テストケースとしてまとめた結果を示す．

表 10: 順序組合せテストによる論理的テストケース

新規フライト予約

No	論理的テストケース	順序組合せ
1	フライト予約後の空き情報問合せによる同一フライトの参照	$Ta_2/Ds_1U \xrightarrow[Ds_1]{} Ta_1R$
2	フライト予約後の再度同一フライトの予約	$Ta_2/Ds_1U \xrightarrow[Ds_1]{} Ta_2/Ds_1U$
3	フライト予約後の同期処理によって最新のチケット残数の計算	$Ta_2/Ds_1U \xrightarrow[Ds_1]{} Ta_6U$
4	既存注文開く画面での予約したフライトの参照	$Ta_2/Ds_2C \xrightarrow[Ds_2]{} Ta_3R$
5	注文件数グラフ・注文履歴の一覧への新規予約フライト予約の反映	$Ta_2/Ds_2C \xrightarrow[Ds_2]{} Ta_5R$

表 11: 新規フライト予約の画面遷移表（一部分）

遷移→ 前状態↓	フライト検索ボタン	キャンセル Or	OK	注文を開く	グラフ	新規予約	Or 終了	履歴	XOr 終了
フライト予約画面	[出発日・From・To 入力済] フライトテーブル			注文を開く画面	グラフ	[フライト便・クラス・人数・搭乗者・入力済] フライト予約画面		履歴	
フライトテーブル		フライト予約画面	[フライト選択済] フライト予約画面						
注文を開く画面			[予約番号] フライト予約画面						
グラフ							フライト予約画面		
履歴									フライト予約画面
ログイン			フライト予約画面						

5 考察

次に，提案手法で抽出したタスク間の順序組合せと，*AS* のテストケースを設計する既出の手法である状態遷移テストで抽出されるテストケースの比較を行う. 状態遷移テストのテストベースとなるフライト予約システムの画面遷移表（表 11）を使って, 順序組合せの確認が可能な網羅基準である S1 網羅基準を適用する．表 11 は，4 節の提案手法適用例と適用範囲を合わせるために，新規フライト予約を行うために必要な画面と隣接する画面遷移に該当する表としている．仕様の詳細度合いは，DFD，ER 図，CRUD 図と画面遷移表では同等にしている．それは，画面遷移のイベントでのガード条件に記載したデータが DFD のエッジに記載したデータ，ER 図のエンティティの属性と一致していることから確認できる．S1 網羅基準を適用すると 28 の状態遷移パスとなる．28 の状態遷移パスのうち，対応する提案手法で抽出した順序組合せは，表 10 で示すテストケース No.1，2，4，5 の 4 つであった．これらは, 画面遷移表のフライト予約状態での登録イベントを起点にするもののみであった．順序組合せに該当しない状態遷移パスは，互いのタスクで同一のデータを介して処理をするといったことがない．例えば，フライト検索をした後にキャンセルをするとフライト予約画面に遷移するパスは，前の処理の結果によって影響を及ぼさない．

S1 網羅基準では抽出できないが，本手法によって抽出できたテストケースは，No.3 の $Ta_2/Ds_1U \xrightarrow[Ds_1]{} Ta_6U$ である．このテストケースは，必要なテストケースと考えられる．4 節の適用評価にて利用したテストベースは，変更が入った機能セットに焦点を絞ったものである．4 節では新規フライト予約が該当する。そのため，表 11 では，新規フライト予約に隣接する画面遷移を該当するテストベースとしている．表 10 のテストケース No3 における波及タスクである Ta_6U は，表 7 から同期処理のタスクであることがわかるが，新規フライト予約とは別の機能セットに含まれるタスクであり，フライト予約画面と隣接する画面からなる画面遷移表には現れない．そのため，S1 網羅基準では抽出することができない．

6 おわりに

本論文では，状態遷移を持つソフトウェアにおいて，変更による変更波及がデータベースや外部変数などの保持データを介して生ずる場合のテストに関して，その網羅基準である IDAU 法と, 順序組合せテストケースを抽出する手法を提案した．DFD，ER 図、CRUD 図をテストベースとして，3 つのルールを適用することでテストが必要となる順序組合せを抽出できることを説明した．提案した手法で組合せが抽出できることを確認するため，フライト予約システムの仕様を具体例にして適用を行った．最後に画面遷移図（画面遷移表）から従来手法である状態遷移テストの S1 網羅基準にて抽出した状態遷移パスと提案手法を比較して，テストケース数の削減ができる効果と，S1 レベルの画面遷移の網羅では抽出できないテストケースが抽出できる効果を示した．

提案手法に対する今後の取り組みは 2 つある．1 つは,

適用範囲の明確化である. 変更のパターン（タスク内の制御ロジックの変更, 新しい要素の追加など）に対して, どこまで適用でき, どこからは適用できないかを明らかにする.

もう 1 つは，今回の提案手法のツール化である．実際の開発プロジェクトで扱う規模の大きいデータ設計文書に対して本手法を適用する際には，本手法のルールをツール化するといった方法での適用が必要になる．これらの準備を行い，実践の場に本手法を適用していく．

参考文献

[1] R.S. Arnold, Software change impact analysis, IEEE Computer Society Press, 1996.

[2] B. Li, X. Sun, H. Leung, and S. Zhang, “A survey of code-based change impact analysis techniques,” Software Testing, Verification and Reliability, vol.23, no.8, pp.613–646, 2013.

[3] B. Beizer, Software Testing Techniques, Van Nostrand Reinhold, 1990. (翻訳 :小野間 彰， 山浦恒央 :ソフトウェアテスト技法, 日経 BP 出版センター (1994)).

[4] H. Gomaa, “Designing software product lines with uml.,” SEW Tutorial Notes, pp.160–216, 2005.

[5] L.C. Briand, Y. Labiche, and L. O’sullivan, “Impact analysis and change management of UML models,” Software Maintenance, 2003. ICSM 2003. Proceedings. International Conference onIEEE, pp.256–265 2003.

[6] 小谷正行，落水浩一郎，“UML 記述の変更波及解析に利用可能な依存関係の自動生成,” 情報処理学会論文誌，vol.49，no.7，pp.2265–2291，2008.

[7] J.N. Campbell, “Data-flow analysis of software change,” Oregon Health & Science University, pp.1–118, 1990.

[8] A. Maule, W. Emmerich, and D.S. Rosenblum, “Impact analysis of database schema changes,” Proceedings of the 30th international conference on Software engineeringACM, pp.451–460 2008.

[9] 加藤正恭，小川秀人，“CRUD マトリクスを用いたソフトウェア設計影響分析手法,” 情報処理学会第 73 回全国大会講演論文集，vol.73，pp.249–250，2011.

[10] ISTQB/FLWG, Foundation Level Syllabus, International Software Testing Qualifications Board, 2011.

[11] T. DeMarco, “Structure analysis and system specification,” Pioneers and Their Contributions to Software Engineering, pp.255–288, Springer, 1979.

[12] A.L. Politano, “Salvaging information engineering techniques in the data warehouse environment,” Informing Science, vol.4, no.2, pp.35–44, 2001.

[13] T.Yumoto， K.Uetsuki， T.Matsuodani, and K.Tsuda, “A study on the efficiency of a test analysis method utilizing test-categories based on aut and fault knowledge,” ICACTCM ’ 2014, pp.70–75, 2014.

[14] ISO/SC7/WG26, Software and Systems Engineering-Software-Testing Part 2:Test Processes, ISO/IEC/IEEE29119 2013(E), 2013.