第２分科会

『場』の発見－暗黙の要件を見つける方法論－

Discovering the “BA" : A methodology for finding tacit requirements

主査：　早川　勲　（株式会社 山武）
副主査：　板倉　稔　（株式会社 イネーブル・ツリー）
リーダー：　松村　英孝　（東京海上日動システムズ株式会社）
研究員：　平田　博義　（三菱電機マイコン機器ソフトウエア株式会社）
安井　章和　（三菱電機コントロールソフトウェア株式会社）
根上　勇一　（アンリツ株式会社）
中村　守利　（株式会社インテック）

概要

本論文では、要件定義の漏れを無くす方法を研究した。状態遷移図と状態遷移マトリクスを用いて、要件定義の漏れを見つけ出すある方法論を用いて簡単な例で実践した。この結果、「機械的に要件の漏れを見つけ出せる」「仕様変更の影響範囲が明確になる」「処理の再利用が可能となる」「開発者、ユーザが理解しやすい」といった効果を見出した。要件定義の漏れを無くす有効な手段として紹介する。

Abstract

In this article, we studied a method to find a leak of the requirements. We practiced it in a simple example using a methodology to find a leak of the requirements. As a result, we found an effect that, "It is possible to find a leak of the requirements mechanically.", "It becomes clear to change the specification.", "It is possible to reuse the processing.", "It is easy to understand between the developer and user." We introduce the effective method to find a leak of the requirements.

1.　課題

要件定義の漏れは、開発者、ユーザ共に不利益をもたらす。開発者側は、開発工程の終盤になって仕様追加･変更が生じることで納期遅延やコストオーバーとなる。ユーザ側は、納期遅延によって業務計画の見直しが発生し、業務内容によっては重大問題に発展する。

要件定義が漏れる主因は、開発者とユーザ間で合意を取らずに仕様が曖昧なまま開発を進めることにある。ユーザが精通している業務内容を経験の無い開発者が完全に理解するのは困難であり、開発者は、どこまでを要件の範疇としたら良いのかわからない。また、ユーザは何を要件として定義すれば自分達が望むシステムになるのかわからない。開発者とユーザ間でシステム化するにあたっての業務課題を明確にし､共有しなければならない。本研究では、要件定義の漏れを方法論で解決することを課題とした。

2.　目的

要件定義の漏れを無くすために以下が可能な方法論を見つけることを目的とした。

① 開発者とユーザが互いに理解出来るモデルであること。
② 要件を形式的に見つけ出せること。

3. 方法論

本研究では、実世界に存在するありとあらゆるシーンを『場』と呼ぶ。券売機を例にすると、「お金を入れる」「行き先を選ぶ」といった『場』がある。

『場』を発見する手法として、2007 年 第 26 回ソフトウェア品質シンポジウム経験論文で発表された G2BizMo（GENCHI-GENBUTSU Business Modeling；究極のシステム開発技法）[1]の適用を試みた。

G2BizMo は、次の手順でビジネスモデルを分析する。

① ビジネス・フローを描く。（実世界を写像する）
② ビジネス・フローを状態遷移図にする。（写像した実世界を形式変換する）
③ 状態遷移図をレビューする。
（3 つの観点(飛越し、出戻り、状態とイベントの存在有無)でレビューする）
④ 状態遷移図を状態遷移マトリクスにする。（機械的に形式変換する）
⑤ 状態遷移図上で、個々の状態とすべてのイベントの組合せをレビューする。
（形式化された方法でレビューする）
⑥ 状態とイベントの組合せに処理を定義する。
（名前を付ける場所が形式的に分かる）
⑦ ER 図を作り、エンティティの項目の実世界の意味を定義する。（既存方法論）
⑧ イベントの画面、処理を定義する。（既存方法論）

実世界を写像したビジネス・フローは、ユーザにとって分かりやすい。ビジネス・フローを状態遷移図にすると、曖昧な仕様が排除しやすくなる。③で示した 3 つの観点

・ある状態から前の状態に戻ることがあるかの確認（出戻り）
・状態から飛び越えて 1 つ先や 1 つ先の状態へ変化することがあるかの確認（飛び越し）
・状態とイベントが本当に存在するかの確認（状態とイベントの存在有無）

に絞られているため、レビューはユーザを交えて効率良くできる。また、状態遷移マトリクスにすることで、網羅的に『場』の組合せを見出すことができる。

この結果、見えていなかった要件が見えるようになり、漏れが無くなる。開発者とユーザは共通した理解で要件定義ができるようになる。

状態遷移図は、状態遷移マトリクスに機械的に変換することができ、コンピュータ化する処理まで連続的に変換できる。今回は、実装レベルまで確認することはやめ、要件定義の質の向上を確認することとした。

本研究では、一般的なビジネスモデルを例題に①～⑥を適用しようと考えたが、例が簡単なので、①を省略した。

4. 方法論の適用

状態遷移図、状態遷移マトリクスを作成し、『場』を発見することができるか例題を用いて実践した。初期要件への適用と追加要件への適用の 2 つについて説明する。

4.1. 初期要件への適用

4.1.1. ビジネス・フローを状態遷移図にする

ビジネス・フローの作成を省略し、「複数枚選択可能な券売機の処理」を例題に状態遷移

図を作成した。「発券」を管理対象として、G2BizMo に従い状態とイベントで表現した。作成した状態遷移図を図 1 に示す。研究員にて 5 つの状態遷移図が提案されたが、最もシンメトリに近い状態遷移図を採用した。

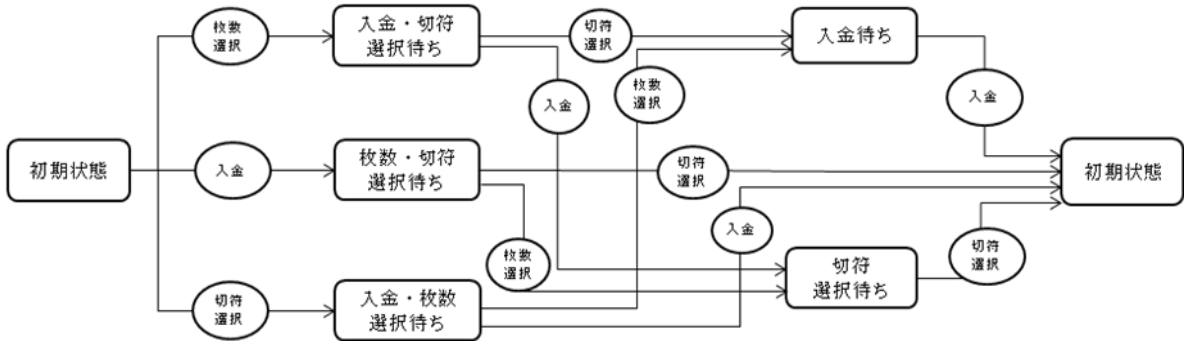

図 1　状態とイベントで定義された状態遷移図

4.1.2. 状態遷移図のレビュー

図 1 では中心となる流れしか表していない。細かな流れとなる横道が存在しているか確認が必要である。それぞれの状態でどんなイベントがあり得るかを考えるため、図 1 の状態遷移図を 3 章で述べた 3 つの観点でレビューをする。

レビューした結果を反映させた状態遷移図を図 2 に示す。出戻りの確認時に図 2 の太線部分、繰り返し可能な処理が抜けていることに気付いた。また、状態とイベントの再確認により、図 2 の点線部分で示す、再選択可能なイベントが存在することに気付いた。取消のイベントも存在したが、全ての状態で起こるため、取消イベントは状態遷移マトリクスで取り込む。

それぞれの状態でどんなイベントがあり得るのかを考えることにより、思いつきではなく『場』を発見することができた。

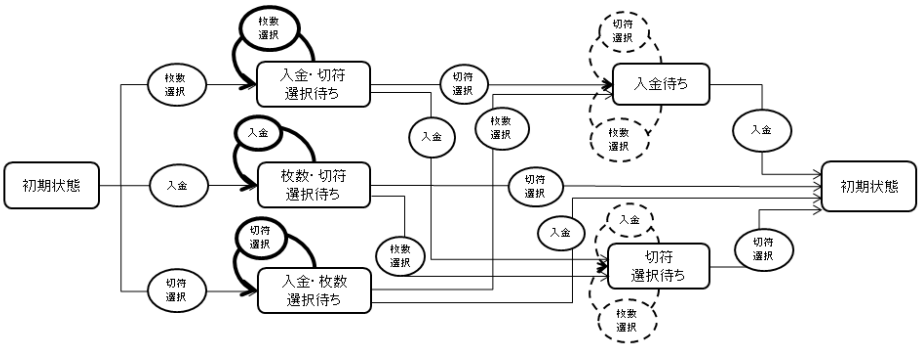

図 2　レビュー後の状態遷移図

4.1.3. 状態遷移マトリクスをレビューする

(1) 状態遷移図を状態遷移マトリクスにする

図 2 の状態遷移図をみると、管理対象が、ある状態（現状態）にいるとき、イベントが来ると次の状態（次状態）に遷移することがわかる。これを表形式にしたものが表 1 である。従って図 2 と表 1 は等価である。

表 1　状態遷移マトリクス

現状態	イベント	次状態
初期状態	枚数選択	入金･切符選択待ち
初期状態	入金	枚数･切符選択待ち
初期状態	切符選択	入金･枚数選択待ち
入金･枚数選択待ち	枚数選択	入金待ち
入金･枚数選択待ち	入金	初期状態
入金･枚数選択待ち	切符選択	入金･枚数選択待ち

(2)状態遷移マトリクスをレビューする

4.1.2 の状態遷移図のレビューでは管理対象の状態を順列に整理した。表 1 の一つ一つの状態に対して、全てのイベントの組合せ（表 2 太線部分）マトリクスを作ることで網羅的に再確認することができる。これは状態とイベントの掛け算となっており、表 2 に示す。

ここで後回しにしていたタイムアウトや、取消操作等の状態遷移図が複雑になるイベントも全て取込む。総当りで確認を行うことにより横道の検討漏れを防ぐことができる。

表 2　状態遷移マトリクスレビュー

現状態	イベント	次状態
初期状態	枚数選択	入金･切符選択待ち
初期状態	入金	枚数･切符選択待ち
初期状態	切符選択	入金･枚数選択待ち
初期状態	取り消し	初期状態
初期状態	タイムアウト	初期状態
入金･枚数選択待ち	枚数選択	入金待ち
入金･枚数選択待ち	入金	初期状態
入金･枚数選択待ち	切符選択	入金･枚数選択待ち
入金･枚数選択待ち	取り消し	初期状態
入金･枚数選択待ち	タイムアウト	初期状態

4.1.4. 状態とイベントの組合せに処理を定義する

表 2 で得られた状態とイベントの組合せで、一意に処理が決まる（参考文献）。そこで、処理を定義した表 3 を作成する。表 3 は、表 2 を太線左部分に転記し、太線右部分に処理を追記する。

以上で、管理対象の状態とイベントの組合せが決まり、その各々で何をすべきかが決まった。今回は方法論の②～⑥を 1 サイクルのみ行ったが、実際の業務では新たな横道が見つからなくなるまで 2，3 回サイクルを回す（参考文献）。

表 3　状態遷移マトリクスに処理の追加

状態遷移			処理					
現状態	イベント	次状態	ありえない	何もしない	エラー	発券	返金	金額計算
初期状態	枚数選択	入金･切符選択待ち		1				
初期状態	入金	枚数･切符選択待ち		1				
初期状態	切符選択	入金･枚数選択待ち						1
初期状態	取り消し	初期状態		1				
初期状態	タイムアウト	初期状態	1					
入金･枚数選択待ち	枚数選択	入金待ち						1
入金･枚数選択待ち	入金	初期状態				1	1	
入金･枚数選択待ち	切符選択	入金･枚数選択待ち						1
…	…	…						
入金待ち	枚数選択	入金待ち						1
入金待ち	入金	初期状態				1	1	
入金待ち	切符選択	入金待ち						1
入金待ち	取り消し	初期状態		1				
入金待ち	タイムアウト	初期状態		1				

4.2. 要件追加への適用

要件追加として「磁気カードにお金をチャージできる追加機能が発生した」ことを想定し、状態遷移図を作成した。追加した部分を図 3 に太線で示す。表 3 を見てもわかるように、状態の遷移が追加された場合でも、新たに状態とイベントの組合せが増えるのみで、他の状態とイベントの組合せに影響を与えることはない。また、処理は状態とイベントの組合せで決まるので、追加した組合せの処理のみを考えればよい。処理が独立しているため変更箇所以外へ影響を与えることがない。検討中に要件が追加になった場合でも、追加や変更したい部分のみ手を加えればよいので容易に対応できることがわかった。

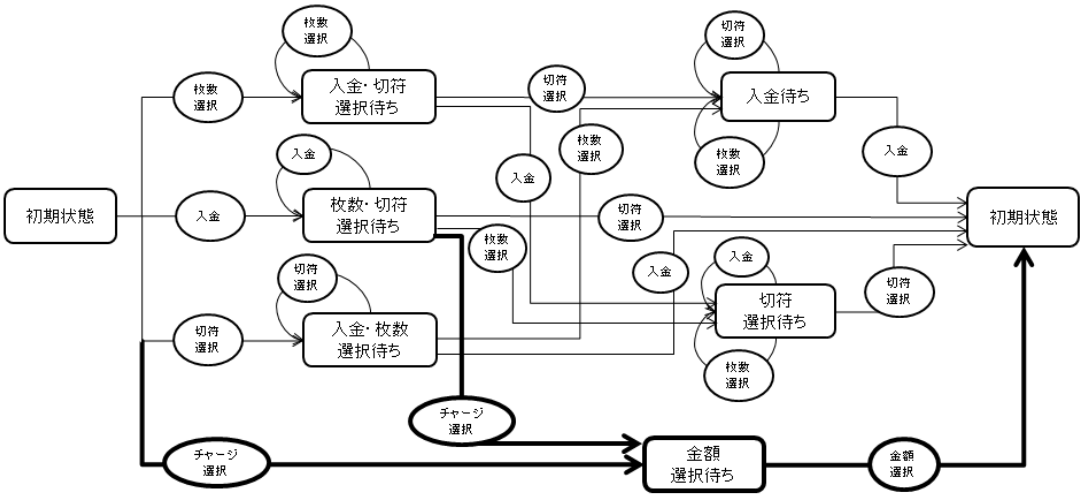

図 3　要件追加反映後の状態遷移図

4.3. 適用のポイント

我々が、G2BizMo を適用したときに試行錯誤した事例に基づき、適用のポイントをまとめる。

(1) 状態遷移図と条件分岐図

ビジネス・フローを状態遷移図で表現する際に、図 4 で示したように条件分岐図を描いてしまうことがあった。条件分岐図は、条件による処理の流れを表現した、言わばフローチャートである。一方、状態遷移図はイベントにより管理対象の状態が遷移する様子を表した図であり両者は別物である。

状態遷移図で分析をしているつもりでも、機能中心に考えているため、状態遷移図ではなく条件分岐図を描いている場合がしばしばある。状態遷移図を作成する場合、この点に注意してほしい。

状態遷移図に馴染みの無い人は、少し練習が必要である。我々の場合、二度目の例題では、全員が状態遷移図を書けるようになった。コツは、「処理」を考えるのではなく、管理対象の変化に着目し、変化のトリガとなるイベントを見つけ出すことである。

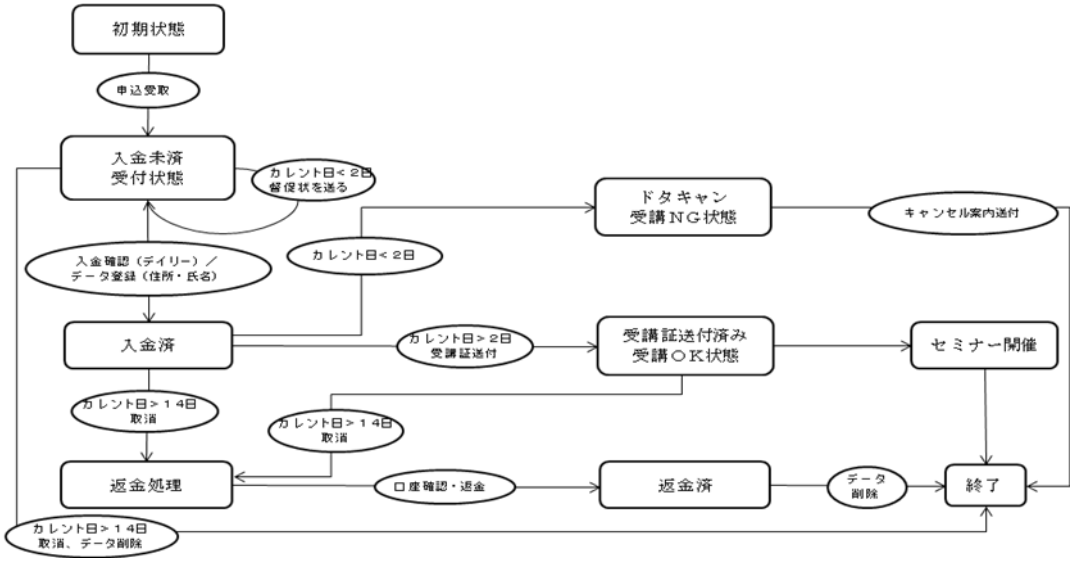

図 4　状態とイベントで定義されていない状態遷移図

(2) シンプルになってないと理解できない

状態遷移図を描いていると、イベントが複雑に交差し、パッと見て理解できない図になることが多かった。パッと見て理解できる図、すなわちシンプルな図を描くコツは、状態やイベントを見つけ出したら、対になる状態やイベントが存在しないかを考えることである。図は「シンメトリ」を保つよう心掛けると美しくシンプルな図を描くことができる。

美しい図が描けない場合は、仕様が整理できていない可能性がある。

(3) 管理対象の見極め

状態遷移マトリクスを作成すると状態とイベントの組合せが爆発的に膨れ上がる。このような場合は、管理対象の設定を誤っている可能性がある。独立したライフサイクルを持つ管理対象を見つけ出し、個々に状態遷移マトリクスを作成すると爆発が防げる。

5.　方法論の特徴と効果

4 章までで述べた検証の結果、いくつかの検討漏れを発見することができた。今までの個人の勘や経験に依存していた要件定義と異なり、状態遷移図と状態遷移マトリクスを利用することで網羅的に要件の横道を検討することができた。つまり『場』を発見することで、気付いていない要件を発見することができた。以下に方法論の特徴と効果を述べる。

5.1. 特徴

5.1.1. 初めに処理ありきではない

ユーザから要件を出されると、まず「処理」を考えがちであるが、本方法論を適用する場合は、まず「管理対象」を見極めることから始まる。「管理対象」の「状態」と「イベント」で「処理」が決まる。処理は結果なので、始めに処理ありきではない。

5.1.2. 業務処理条件と処理自体が独立する

G2BizMo では、ビジネスは、管理対象の状態とイベントの組合せで処理と次状態が一意に決まるものであるという原則（参考文献）に基づき、状態遷移マトリクスを作成し、管理対象の状態とイベントの交点で処理を検討する。その結果、業務処理条件は状態遷移図（表 3 状態遷移マトリクスの状態遷移部）で表される。処理自体は表 3 の処理部で決まる。このように、業務処理自体と処理条件が独立する。

5.1.3. 方法論が確立している

3 章から 4 章で述べた通り、方法論が確立している。従来は勘と経験しか無かった要件定義工程に方法論が導入されたので、精度の高い要件定義を形式的でかつ、再現性の高いやりかたで実現することができた。

5.2. 効果

5.2.1. ユーザ、開発者が理解しやすい

G2BizMo で描いた状態遷移図は実世界の構造を管理対象とその状態､及びイベントで図示したものである。つまり、状態遷移図は実世界の管理対象のライフサイクルを図示しているので、ユーザに理解しやすい。

また、開発者側としても状態遷移図や状態遷移マトリクスは馴染み深い。今回は組み込み系、エンタープライズ系など分野が異なる研究員 5 人で「セミナー受講」「券売機」の事例で G2BizMo を利用した。その結果、一度目の例題では方法論を理解しきれなかったが、二度目の例題で全員が理解できた。その証拠に見落としがちな要件を見つけ出すことができた。

5.2.2. 機械的に要件の漏れを見つけ出せる（網羅性が高い）

4.1.2、4.1.4 での検証の結果、いくつかの検討漏れを発見することができた。今までの個人の勘や経験に依存していた要件定義と異なり、状態遷移図と状態遷移マトリクスを利用してレビューすることで網羅的に要件定義が必要な『場』を発見する。これらの『場』を検討することにより、「要件」として認識がない「要件」を発見するきっかけとなった。

5.2.3. 要件変更の影響範囲が明確になる（修正による影響の局所化）

要件変更にも様々なものがあるが、4.2 で述べたような要件追加ではライフルサイクルの変更や処理の追加を伴い複雑なものとなる。一般的には処理自体に業務条件が含まれて

いるので、何をどこまで修正すればよいのかわかりづらい。本方法論を利用すれば表 3 のように業務処理条件と処理自体が分かれる。これにより管理対象の状態が増減することと対応する処理を定義するだけで、既存の処理手続き自体に影響を及ぼさない。

5.2.4.処理の再利用が可能となる

5.1.2 でも述べた通り､本研究で使用した方法論は業務処理条件と処理自体を独立させることができる。例えば、表 3 の「金額計算」処理は複数の状態とイベントの組合せから使われる。これは処理の中に業務処理条件の分岐を含まないので、新しい状態やイベントが増えた場合でも処理の再利用が可能となる。

5.2.5.経験のない人でも上流工程を担当できる

本方法論で要件定義の漏れを見つけることができた。今までは勘と経験に依存していた要件定義を形式に則ってできるようになる。つまり、あまり経験のない人も上流工程を担当できるようになる。

6. 結論

4 章、5 章で述べたように、G2BizMo を適用することで、勘や経験に頼ることなく、要件の漏れを発見できた。

検証に用いた事例は簡単なものであり、要件の漏れは無いと考えていたが、検証を開始してみると、要件定義が曖昧である箇所を発見する結果となった。このことより、簡単であると考えがちな仕様にさえ曖昧な要件定義が潜んでいることがわかった。

7. 終わりに

本方法論は、管理対象がライフサイクルを持つことを前提としているため、統計処理のような管理対象が変化しない要件定義には不向きである。この分野は、部分集合を扱う SQL で解決していると考える。

すべての要件は､「ユーザの頭の中」にあるが、ユーザは何をどこまで提示すれば、開発者の求める要件となるのかがわからない。この方法論は「暗黙知」の発見能力の高い方法論である。要件定義の漏れを無くすには本方法論の適用を推奨する。

参考文献

[1] 板倉稔, “究極のシステム開発技法 G2BizMo－実世界からコンピュータ世界への連続変換モデル技法－”, 第 26 回ソフトウエア品質シンポジウム発表報文集, pp187-194, 日科技連, 2007