情報系の実稼働システムを対象とした
形式手法適用実験報告書

別冊　詳細報告書

平成 24 年 4 月

独立行政法人情報処理推進機構
技術本部ソフトウェア・エンジニアリング・センター
統合系システム・ソフトウェア信頼性基盤整備推進委員会

形式手法技術部会
形式手法適用実証 WG

はじめに

独立行政法人情報処理推進機構技術本部ソフトウェア・エンジニアリングセンター（以下、IPA SEC と略す）「統合系システム・ソフトウェア信頼性基盤整備推進委員会」では、「形式手法適用実証」WG を設置し、実在するシステムの外部設計書を題材として形式手法を適用し、その効果を確認するための実験を行った。本実験では 3 種類の形式手法、Event-B、SPIN、VDM++を使用した。使用する形式手法や適用方法、適用目的ごとに 5 つの実験チームを編成し、並行して実験を実施した。本報告書は、実験チーム単位で実施した実験の内容および結果を詳しくまとめたものである。

本報告書の構成は次のとおりである。

第 1 章から第 5 章までは、以下の通り、各実験チームの報告をまとめたものである。

- 第 1 章：Event-B を使った実験（その１）
- 第 2 章：Event-B を使った実験（その２）
- 第 3 章：Event-B を使った実験（その３）
- 第 4 章：SPIN を使った実験
- 第 5 章：VDM++を使った実験

第 6 章と第 7 章では、以下のような確認を目的として、参考のため追加で行った実験について報告する。

- 第 6 章：システム稼働後の問題を事前発見できる効果の確認
- 第 7 章：形式手法の表現力と検証能力の確認

各章では、それぞれ以下のような構成で報告する。

- 実験の目的
- 実験チームの体制
- 実施した作業内容・手順
- 実験にかかった工数や指摘事項件数などの定量的な結果
- 実験結果の考察

情報系の実稼働システムを対象とした形式手法適用実験報告書
別冊　詳細報告書

独立行政法人情報処理推進機構

目次

1. Event-B を使用した実験（B1 チーム）

1.1 目的

本実験において、何を確認するために、どのような工程成果物に対して、どの形式手法を用いたかを表 1-1 に記す。

表 1-1 目的

採用した手法	Event-B
（特徴）	Event-B は、システムのアクションを表す「イベント」と、システムが守るべき制約を表す「不変条件」を、記述・検証する手法である。Event-B 用のツールである Rodin Platform、およびそのプラグインによって、モデル検査や定理証明で整合性を検証したり、仕様アニメーションで動作を目視確認したりできる。
確認すること	システムのアクションとデータベースに着目し、画面からのユーザ操作をトリガとして実行される登録/削除といったシステムのアクションが、状態遷移図によって定義されたデータ制約を守ってデータベースの更新を行っているかを確認する。
（確認方法）	登録/削除といったシステムのアクションを「イベント」に、状態遷移図から得られる制約を「不変条件」にし、モデル検査と定理証明を実施する。
確認の対象	画面アクション明細 エンティティ定義 状態遷移図

1.2 体制

本実験を行った作業者のスキル（経験）を表 1-2 に記す。

表 1-2 体制

作業者	スキル （経験）			
	業務 AP 開発	類似 AP 開発	形式手法	採用手法
A	1 年	0 年	3 年※1	2.5 年

- 業務 AP 開発：業務でのアプリケーション開発の経験年数
- 類似 AP 開発：本実験で対象としたアプリケーションに類似したアプリケーションの開発に限った経験年数
- 形式手法：形式手法全般の経験年数
- 採用手法：本実験で採用した形式手法である Event-B に限ったの経験年数

※1：3 年の内訳

1.250 年…トップエスイー※2 での講習（1 回 1.5 時間の講義を 48 回、計 72 時間）

0.250 年…上記講習の修了制作

0.125 年…DSF※3 内での講習（1 回 4 時間の講義を 8 回、計 32 時間）

1.375 年…DSF での実務

※2：トップエスイー

http://www.topse.jp/

※3：DSF

http://www.nttdata.co.jp/dsf/

1.3 作業

1.3.1 作業の概要

本実験では、図 1-1 のような一連の作業を行った。

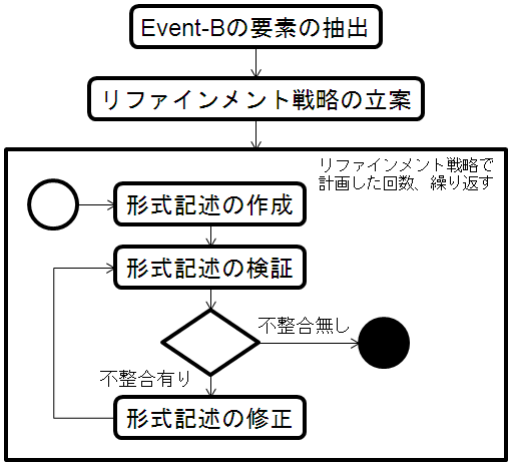


図 1-1 作業の流れ

この作業の流れは、『形式手法適用手順（Event-B 編）【リリース版】』 [1]の「適用法 1」

をもとに、本実験の目的外である「工程成果物への反映」を除いたものである。

本実験における各作業の内容を表 1-3 に記す。

表 1-3 作業の内容

作業名	内容
Event-B の要素の抽出	与えられた外部設計書の全体について、文書の構成や内容を理解し、形式手法を適用する対象を選定する。形式手法を適用する対象に選定したものについて、「イベント」にするシステムのアクション、「不変条件」にするデータ制約、およびそれらを記述するのに必要な「キャリアセット」や「変数」などを抽出する。
リファインメント戦略の立案	抽出した Event-B の要素について、その詳細さで順序付けし、より抽象的な要素から順に形式記述を作成・検証していくよう計画する。
形式記述の作成	抽出した Event-B の要素を、Event-B の文法に従って、形式記述にする。
形式記述の検証	形式記述に対して、モデル検査と定理証明を実施する。
形式記述の修正	検証の結果、形式記述に不整合が発見された場合、それを修正する。

1.3.2 Event-B の要素の抽出

1.3.2.1 与えられた外部設計書の理解

本実験で最初に行ったことは、実験の対象となった外部設計書の内容を理解することである。実験の対象となった外部設計書は、本実験の作業者が作ったものではないため、まずその内容を大まかにでも知る必要がある。

まず、与えられた外部設計書の規模を粗く調べてみたところ、ファイル数にして 400 近くあり、この時点で与えられた外部設計書の全体を実験の対象とするのは現実的ではないと判断した。後に詳細を調べてみると、全体で 16,320 ページにおよぶ外部設計書であった。

1.3.2.2 形式化範囲の絞り込み

先にも述べたように、与えられた外部設計書の全体は、全 16,320 ページにおよぶ膨大なものであったため、限られた実験期間内に一名の作業者で全てを扱うことは困難であり、

形式手法を適用する対象を絞り込む必要があった。本実験では、形式手法を適用する対象を、次の観点で絞り込んだ。

- システム全体のうち、ソフトウェアの機能要件に関する設計
- システム全体のうち、適当な大きさのサブシステム
- データベース上のテーブルのレコードを更新するアクション
- アクションの制御フローに影響するテーブルの属性

はじめに、システムの全体から、ソフトウェアの機能要件に関する設計のみに絞り込んだ。本実験で採用した形式手法 Event-B では、ハードウェアの細かな構成や物理配置などに起因する指摘事項の抽出には向かないであろうと判断したためである。同様に、パフォーマンスやユーザビリティといった非機能要件についても、Event-B には向かないであろうと判断し、形式化範囲から除外した。

次に、先に絞り込んだ範囲から、適当な大きさのサブシステムを選定した。対象システムは「サイト」と呼ばれる九つのサブシステムと、サイト共通処理、共通基盤によって構成されていた。九つのサブシステムは、多くの点で似通った設計がなされているサブシステムであり、どれを切り出しても作業工数は変わらなそうであった。サイト共通処理と共通基盤は、与えられた外部設計書にはあまり多くの情報がなく、上手く形式手法の適用対象を切り出せなかった。以上の分析から、九つのサブシステムから適当に一つのサブシステムを選定することとした[1]。

次に、選定したサブシステムのアクション一覧（図 1-2）と、各アクションの定義から、データベース上のテーブルのレコードを更新するアクション（登録、削除、変更、修正、強制削除、基本情報設定）と、レコードを読み込むだけのアクション（閲覧、検索）に大別し、レコードを更新するアクションのうち「基本情報設定」を除いた五つのアクションを形式化範囲とすることにした。

[1] 正確には、さらにサイトにある八つの機能の中から一つの機能にまで絞っているが、説明が冗長になるため割愛する。

図 1-2 サブシステムのアクション一覧

レコードを更新するアクションもレコードを読み込むだけのアクションも、どちらも形式化範囲に含めることもできたが、本実験でレコードを更新するアクションのうち「基本情報設定」を除いた五つのアクションを形式化範囲としたのは、次の理由からである。

- レコードの属性が守るべき制約を示した「状態遷移図」が存在したこと

「状態遷移図」には、属性が取り得る値の組み合わせ（状態）と、その変化のパターン（遷移）が記述されていた。具体的には図 1-3 のように、「書類」と呼ばれるデータについて、「公開状態」や「削除状態」といった項目があり、それらの値の組み合わせによって状態が定義されるといったことが書かれていた。

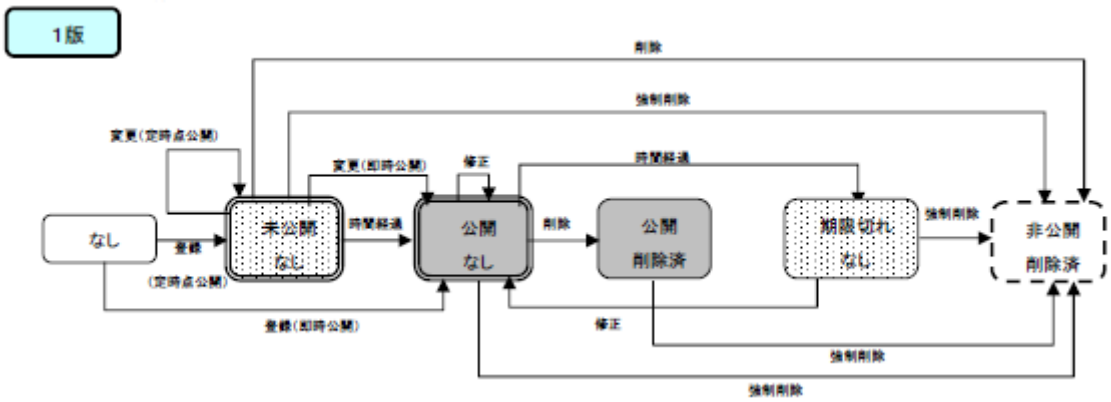
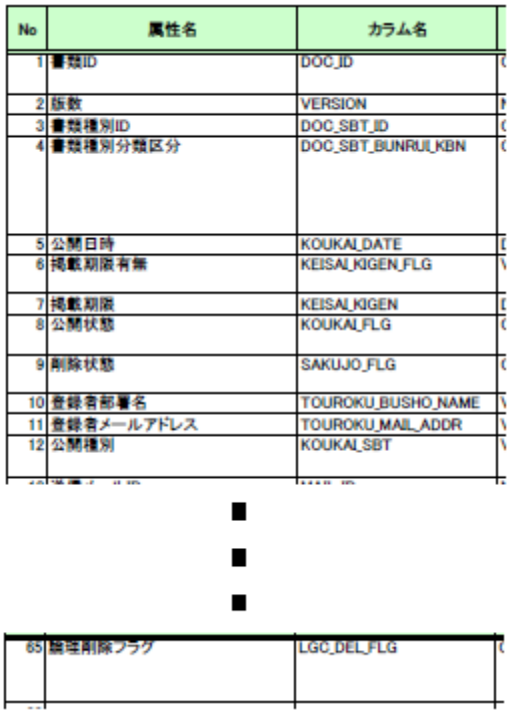


図 1-3 書類の「状態遷移図」

また、「状態遷移図」上の「書類」は、「エンティティ定義」により、データベース上の「書類 ID」、「公開状態」、「削除状態」といった属性を持つ、図 1-4 のようなテーブルのレコードに対応していた。この対応関係の良さから、データベース上のテーブルのレコードを更新するアクションが守るべき制約が導きやすく、本実験で採用した Event-B でのイベント（＝アクション）と不変条件（＝守るべき制約）に、上手く対応すると判断した。なお、レコードを更新するアクションでありながら対象から取り除かれた「基本情報設定」は、「状態遷移図」に書かれた「書類」に関する操作ではなかったため、形式化範囲から除外した。



No	属性名	カラム名
1	書類ID	DOC_ID
2	版数	VERSION
3	書類種別ID	DOC_SBT_ID
4	書類種別分類区分	DOC_SBT_BUNRUI_KBN
5	公開日時	KOUKAI_DATE
6	掲載期限有無	KEISAI_KIGEN_FLG
7	掲載期限	KEISAI_KIGEN
8	公開状態	KOUKAI_FLG
9	削除状態	SAKUJO_FLG
10	登録者部署名	TOUROKU_BUSHO_NAME
11	登録者メールアドレス	TOUROKU_MAIL_ADDR
12	公開種別	KOUKAI_SBT
65	論理削除フラグ	LGC_DEL_FLG

図 1-4 「書類」の属性

次に、「書類」の属性のうち、アクションの制御フローに影響するものだけを選別した。本実験の対象システムは、情報の授受が目的であったため、大半の属性が人に読ませるためのものであり、制御フローに影響する属性は一部であった。本実験では、先の「状態遷移図」に書かれた状態の制約を全てのアクションが守っていることを確認したかったため、人が読むだけの属性を捨象し、アクションの制御フローに影響する属性だけを検証することとした。具体的には、図 1-5 のアクション定義があったとき、「公開日時チェック」という分岐で「公開種別」という属性を使用していたため、「公開種別」を制御フローに影響する属性と判断し、形式化範囲に加えている。

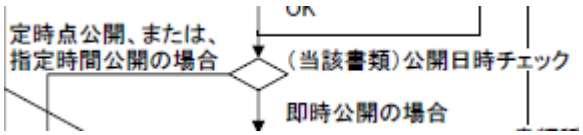


図 1-5 アクションの制御フローに影響する属性

ここまでの分析で、次のアクション、テーブル、属性が、本実験における形式手法の適用対象として選別された。

- 書類を登録するアクション
- 書類を削除するアクション
- 書類を変更するアクション
- 書類を修正するアクション
- 書類を強制削除するアクション
- 書類の情報を格納するテーブル
- 書類の ID を表す属性
- 書類の版数を表す属性
- 書類の公開状態を表す属性
- 書類の削除状態を表す属性
- 書類の論理削除フラグを表す属性
- 書類の公開種別を表す属性

なお、実際の開発ではこういった観点の他に、対象の重要度も考慮して絞り込みを行うべきだが、本実験では実験の作業者とシステムの開発者が別人であることもあり、対象の重要度を的確に判断できないことや、対象の重要度を判断しなくても、本実験の目的である形式手法による改善事項の指摘効果は言えることから、対象の重要度といった観点は含めずに、対象の絞り込みを行った。

また、上述の絞り込み作業では、制御フローに影響する属性をいくつか見逃している。これは、与えられた外部設計書に対する理解が実験の作業者に足りていなかったことに起因した実験上のミスであり、この属性の見逃しが改善事項の見逃しに繋がっている可能性がある。

1.3.2.3 実験の作業者と対象システムの開発者による Q&A

与えられた外部設計書の理解や形式化範囲の絞り込みには、対象システムの開発者との Q&A が大いに役立った。一般的に外部設計書は、開発した企業、チーム、個人の裁量や文化が色濃く反映されていることがあり、そういった背景を知らずに読んでも、文面からだけではうまく内容を理解できないことがある。本実験では、実際に対象システムを開発し

たチームとの Q&A を通して、記述内容や図表の読み方などを確認しながら理解を進めていった。

1.3.2.4　イベントの抽出

与えられた外部設計書には、典型的なウォータフォールモデルの開発プロセスで作られる外部設計書よりは、内部設計書に近い詳細さで記述されているものもあった。例えば「アクション定義」では、『機能要件の合意形成ガイド』 [2]で言うところの「システム化業務説明」に相当する記述はなく、より詳細なフローチャートの記述があった。そこで、フローチャートの内容から、システム化業務相当の情報を読み解きながら、次のルールでEvent-Bの「イベント」、およびその構成要素である「ガード条件」と「コマンド [2]」を抽出していった。

1. 正常系（処理を行い、結果を返すフロー）と、異常系（処理をロールバックし、エラーを返すフロー）に分類し、正常系を「イベント」とする
2. フローチャート中の分岐を「ガード条件」とする
3. フローチャート中の処理を「コマンド」とする

この一連の流れを、図 1-6 のような書類を削除するアクションのフローチャートを用いて具体的に説明する。

[2] Event-B の用語に従うと正確には「アクション」だが、本実験では既に与えられた外部設計書で「アクション」という用語が使われているため、本稿では Event-B のアクションを「コマンド」と呼ぶこととしている。

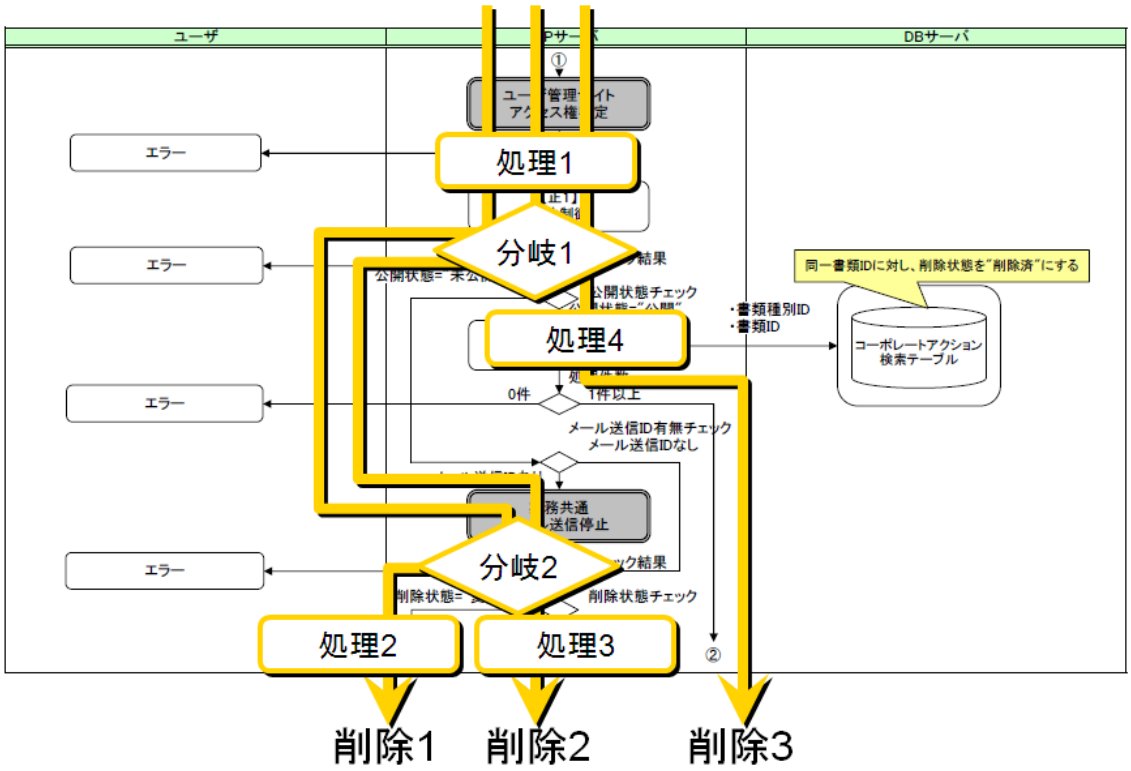


図 1-6 書類を削除するアクションのフローチャート（抜粋）

図 1-6 のフローチャートについて、正常系と異常系の分類を行った結果、矢印が示すような「削除 1」、「削除 2」、「削除 3」の三つの正常系と、「エラー」に到達するいくつかの異常系があることがわかった。異常系は、それが起きてもロールバックされ、処理前の状態にもどるよう設計されていたことから、システムの状態は変わらないため、本実験では捨象することとした。

次に各「イベント」の「ガード条件」を考える。分岐について各線を追ってみると、「削除 1」と「削除 2」は「分岐 1」を左に、「削除 3」は下に流れる。またさらに、「削除 1」は「分岐 2」を左に、「削除 2」は下に流れる。この分岐を通る条件を全て満たす条件[3]が、各「イベント」の「ガード条件」である。

次に各「イベント」の「コマンド」を考える。処理について各線を追ってみると、「削除 1」では「処理 1」と「処理 2」、「削除 2」では「処理 1」と「処理 3」、「削除 3」では「処理 1」と「処理 4」が実行されることがわかる。

以上により、本実験で書類を削除するアクションから抽出された「イベント」は、表 1-4 のようになった。

[3] 実際には、いくつか「エラー」に流れる分岐もあるため、「エラー」に流れないための条件も含まれるが、説明が冗長になるため省略している。

表 1-4 書類を削除するアクションから抽出されたイベント

	ガード条件		コマンド
	分岐１	分岐２	
削除１イベント	左	左	処理 1、処理 2
削除２イベント	左	下	処理 1、処理 3
削除３イベント	下	―	処理 1、処理 4

同様に、他のアクションについてもこの一連の作業を実施し、最終的に本実験では 13 個の「イベント」が抽出された。

1.3.2.5 集合・公理・不変条件の抽出

先に行った対象文書の理解作業の段階で、形式手法の適用対象としたテーブルとその属性について、それぞれをどのような集合・公理・不変条件で定義すればいいかの分析を行った。例えば「書類の ID を表す属性」の値は、登録処理が行われるたびに生成される値であったため、キャリアセットとしての書類 ID と、登録済みの書類 ID を表す変数を用意することとした。「書類の版数を表す属性」は、登録時には 1 版、最大で 99 版であったので、1 から 99 の整数を表す型を用意することとした。その他、「書類の公開状態を表す属性」、「書類の削除状態を表すデータ」、「書類の論理削除フラグを表すデータ」、「書類の公開種別を表すデータ」も同様に、与えられた外部設計書に書かれたデータの仕様に合わせて定義していった。

次にテーブルについて、主キーを構成する属性とその他の属性の関係を、次のように定義した。

- 主キーは、全ての主キーが揃うことでレコードが一意に定まるため、主キーとされた属性の直積を型とした変数として定義した
- 主キー以外の属性は、主キーが決まれば導出可能であるため、先に定義した主キーの直積を型とした変数から、各属性への関数[4]を型とした変数として定義した。

例えば図 1-7 のようなテーブルがあり、「書類 ID」は任意の値、版数は整数、「公開状態」は図 1-8 のような値、「削除状態」は図 1-9 のような値を取る場合で説明する。まず準備として、各属性の型を、「書類 ID」は任意の値を取るのでキャリアセット、「版数」は整数、「公開状態」は図 1-8 より｛未公開、公開、期限切れ、非公開｝の列挙、「削除状態」は図 1-9 より｛なし、変更中、変更有、削除済｝の列挙とする。主キーは「書類 ID」と「版数」なので、「書類 ID」の型と整数の直積とする。「公開状態」は主キー以外の属性であり、取り得る値は｛未公開、公開、期限切れ、非公開｝なので、主キーから｛未公開、公開、期限

[4] NULL が許されない属性なら全域関数、NULL が許される属性なら部分関数。

切れ、非公開｝への関数、「削除状態」も同様の方法で、主キーから｛なし、変更中、変更有、削除済｝への関数とする。なお、実際に Event-B でこれらを定義する場合は、他の定義のしやすさなどとの兼ね合いで変形することもある。例えば主キーのうち「書類 ID」だけを取り出した集合が欲しいケースでは、「書類 ID」の変数を用意し、主キーを「書類 ID」の変数と整数の直積としてもよい。

属性名	主キー
書類ID	○
版数	○
公開状態	
削除状態	

図 1-7 「書類」のテーブル

値
未公開
公開
期限切れ
非公開

図 1-8 「公開状態」の取り得る値

値
なし
変更中
変更あり
削除済

図 1-9 「削除状態」の取り得る値

以上のような分析の結果、本実験では表 1-5 のような集合・公理・不変条件が抽出された。

表 1-5 抽出された集合・公理・不変条件

Sets, Constants, Variables	Axioms, Invariants
書類 ID（DOC_ID）	**キャリアセット**
登録済みの書類 ID（active_DOC_ID）	**登録済みの書類** ID ⊆ **書類** ID
版数（VERSION）	**版数** = 1..99
公開状態（KOUKASI_FLG）	**公開状態** = { **未公開**（InPreparation），**公開**（Public）， **期限切れ**（Expired），**非公開**（Private） }
削除状態（SAKUJO_FLG）	**削除状態** = { **なし**（Unchanged），**変更中**（Changing）， **変更有**（Changed），**削除済み**（Removed） }
論理削除フラグ（LGC_DEL_FLG）	**論理削除フラグ** = { OFF, ON }
公開種別（KOUKAI_SBT）	**公開種別** = { **即時公開**（Instant）， **指定時間公開**（Appointing） }
テーブルの主キー （CA_KENSAKU_TRN_TBL__PRIMARY_KEY）	**テーブルの主キー** ⊆ **登録済みの書類** ID × **版数**
テーブルの公開状態 （CA_KENSAKU_TRN_TBL__KOUKASI_FLG）	**テーブルの公開状態** ∈ **テーブルの主キー** → **公開状態**
テーブルの削除状態 （CA_KENSAKU_TRN_TBL__SAKUJO_FLG）	**テーブルの削除状態** ∈ **テーブルの主キー** → **削除状態**
テーブルの論理削除フラグ （CA_KENSAKU_TRN_TBL__LGC_DEL_FLG）	**テーブルの論理削除フラグ** ∈ **テーブルの主キー** → **論理削除フラグ**
テーブルの公開種別 （CA_KENSAKU_TRN_TBL__KOUKAI_SBT）	**テーブルの公開種別** ∈ **テーブルの主キー** → **公開種別**

最後に、「書類」の「状態遷移図」から、追加の不変条件を抽出した。「状態遷移図」には、大きく分けて次の二つの情報が書かれていた。

- **一つの「書類」が取り得る状態の制約**
- **最新版の「書類」と一つ前の版の「書類」との関係**

一つの「書類」が取り得る状態の制約とは、例えば「第 1 版」で「公開状態」が「未公開」の「書類」について「状態遷移図」を見てみると、「削除状態」が「なし」しかありえない、といった制約である。このとき、「削除状態」が「なし」だからといって、「公開状

態」が「未公開」とは限らないということに注意しなければならない。「削除状態」が定まる場合の「公開状態」の制約は、別途分析する。例えば「第 1 版」で「削除状態」が「なし」の「書類」の場合、「公開状態」は「未公開」、「公開」、「期限切れ」のいずれかである。

最新版の「書類」と一つ前の版の「書類」との関係とは、例えば最新版の「書類」が「公開/なし」である場合、一つ前の版の「書類」は「公開/変更有」となっているというように、同期して遷移する状態の制約のことである。例えば最新版が図 1-10 のように状態遷移し、一つ前の版が図 1-11 のように状態遷移すると定義されており、それぞれの初期状態が「公開/なし」、「公開/変更有」であれば、最新版が「公開/なし」のときは一つ前の版は「公開/変更有」、最新版が「期限切れ/なし」のときは一つ前の版は「期限切れ/変更有」という関係があると言える。

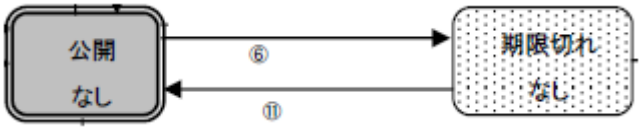


図 1-10 最新版の状態遷移

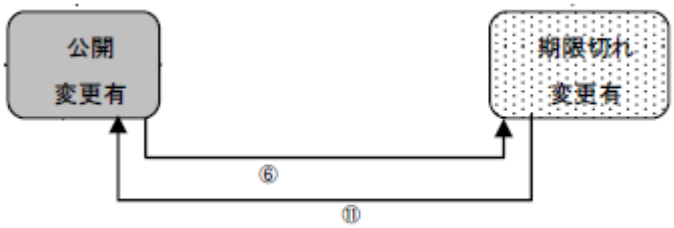


図 1-11 一つ前の版の状態遷移

以上のようにして「状態遷移図」から得られた「不変条件」は、冗長なものを整理して 27 個に及んだ。図 1-12 はその一例で、最新版（active_version）の書類の状態（doc_state）が、「未公開/変更中」（InPreparation_Changing_OFF）である場合、一つ前の版の状態は「公開/変更中」（Public_Changing_OFF）であること、またその逆も成り立つということを表している。

```
∀ doc_id, version
· doc_id ↦ version ∈ active_version ∧ version > 1
⇒ (doc_id ↦ version ↦ InPreparation_Changing_OFF ∈ doc_state
 ⇔ doc_id ↦ version − 1 ↦ Public_Changing_OFF ∈ doc_state)
```

図 1-12 「状態遷移図」から得られた「不変条件」の一例

1.3.3　リファインメント戦略の立案

次に、抽出した Event-B の要素を、その抽象度で順序付けし、抽象度の高い要素から順に形式化するよう、リファインメント戦略を計画した。

抽象度の順序付けの基準と、対応するリファインメントステップは、次のような判断で行った。

- ステップ1.　主キーの要素を最優先とする
- ステップ2.　主キーの要素のみに関係する要素を、次の優先度とする
- ステップ3.　主キーの要素と、他の要素に関係する要素を、次の優先度とする
- ステップ4.　状態遷移図に関係する要素を、最後とする

ステップ 1 では、主キーのみを形式化することで、データベースのテーブルの動きを、大まかに表すことを目的とした。本実験では、「書類 ID」と「版数」が主キーを構成する要素であったため、この二つを対象とした。

ステップ 2 では、主キー以外の要素のうち、他の要素への依存度が小さそうなものを対象とした。本実験では、「公開状態」、「削除状態」、「論理削除フラグ」が、この対象となった。

ステップ 3 では、ステップ 2 で定義した要素に関係する要素を対象とした。本実験では、「公開種別」が、この対象となった。「公開種別」の存在は、「公開状態」の変化を詳細にする。

ステップ 4 では、「状態遷移図」の内容を対象とした。なお、ここでのステップ 4 の判断にはミスがあった。「状態遷移図」の情報は、ステップ 2 で定義できるものや、ステップ 3 で定義できるものがあったため、それぞれのステップで形式化した方が抽象度が揃い、ステップ間の分量のバランスもよくなったはずである。この判断ミスにより、ステップ 4 に作業が偏重し、その分量の多さから後の「形式記述の検証」の遅延へと繋がった。

以上のような作業の結果、本実験では表 1-6 のようなリファインメント計画で形式記述を作成することとした。

表 1-6 リファインメント計画（イベントは省略）

Steps	Sets, Constant	Variables
1	書類 ID 版数 = 1 .. 99	登録済み書類 ID ⊆ 書類 ID テーブルの主キー ∈ 登録済み書類 ID × 版数
2	公開状態 = { 未公開, 公開, 期限切れ, 非公開 } 削除状態 = { なし, 変更中, 変更有, 削除済み } 論理削除フラグ = { OFF, ON }	テーブルの公開状態 ∈ テーブルの主キー → 公開状態 テーブルの削除状態 ∈ テーブルの主キー → 削除状態 テーブルの論理削除フラグ ∈ テーブルの主キー → 論理削除フラグ
3	公開種別 = { 即時公開, 指定時間公開 }	テーブルの公開種別 ∈ テーブルの主キー → 公開種別
4		（状態遷移図を参照）

1.3.4 形式記述の作成

リファインメント計画が定まった後、その計画に従って形式記述の作成を実施した。ここでは概ね、これまでの作業で定義してきたものを清書する作業に終始していたため、特に何事もなく作業は進んだ。

✧ 補足

本実験では、Event-B の要素の抽出後、リファインメント計画を立案する前に、一度形式

記述の試し書きを行った。その目的は、抽出した Event-B の要素をツールで確認することである。特に公理・不変条件は、その表現に数式が出てくるため、ツールによる文法チェックが有効である。

1.3.5 形式記述の検証

次に、作成した形式記述に対して、モデル検査と定理証明を実施した。本節では、モデル検査と定理証明の実施作業上の話題のみを取り上げ、ここで抽出された指摘事項の詳細については、1.4.4 節で説明する。

✧ Event-B のモデル検査

Event-B では、Rodin Platform 用のプラグインの一つである ProB を使うことで、モデル検査を実施できる。ProB では、LTL 式を指定するモデル検査の他、Event-B の不変条件を検証することもできる。本実験では、LTL 式は指定せず、Event-B の不変条件のみを検証した。

また ProB は、モデル検査に必ず付きまとう状態爆発の問題に対して、検証する集合の要素数などを制限するといった方法で回避を試みる。本実験では、ProB のデフォルト値を利用してモデル検査を行った。

✧ Event-B の定理証明

Event-Bでは、Rodin Platformが持つ自動定理証明器と対話的証明インターフェースの他、プラグインとして手に入る自動定理証明器であるAtelier B Provers、およびSMT Solver[5]を、定理証明に利用できる。Event-Bはこのように自動定理証明の環境が充実していることもあり、本実験でも全 848 個の証明課題のうち、約 93%にのぼる 789 個が、ツールにより自動証明された。

✧ 実験上の反省点

本実験でのモデル検査の導入は、実験の作業者の知識不足により当初の予定には入れておらず、実験の途中から導入を開始した。本来なら初めから、定理証明を実施する前に、まずモデル検査を実施するように計画し、実験を実施すべきであった。

また、本実験で自動定理証明器の一つとして利用した SMT Solver プラグインには、特定の条件下で証明が実施できなくなる不具合が存在していた。この不具合は、形式記述の修正を行った後の再検証の際、過去に行った証明の結果が利用できず、多くの証明課題を再び証明しなおさなければならなくなるという事態として、大きな影響を及ぼした。おおよそ、定理証明にかけた時間の約 6 割は、この不具合による、本来ならば不要の再証明の作業であった。

[5] 本実験では、SMT Solver は Alt-ergo を利用した。

1.3.6　形式記述の修正

次に、検証の結果わかった不整合について、形式記述の修正を行う。本実験では、修正方法は実験の作業者が、他の設計内容との整合性や Q&A から考え、修正を実施した。

以降、修正すべき不整合がなくなるまで、「形式記述の検証」と「形式記述の修正」を繰り返す。本実験では、合計二回の修正を行った。

1.4　実験結果

本章では、実験にかかった工数、実験対象の規模、抽出した指摘事項の数などから、検証の生産性、改善事項の指摘効率について考察する。

1.4.1　工数

表 1-7 工数

作業名	工数（人 H）	
Event-B の要素の抽出	34.0	※1
リファインメント戦略の立案	1.5	
形式記述の作成	9.5	
形式記述の検証（モデル検査）	5.5	
形式記述の検証（自動定理証明）	10.0	※2
形式記述の検証（対話的定理証明）	39.0	※2 ※3
形式記述の修正	8.0	
	計 107.5	

※1：約 5 割の時間は、対象文書の理解に費やした工数

※2：約 6 割の時間は、SMT Solver プラグインの不具合による手戻り

※3：約 2 割の時間は、モデル検査を併用していなかった序盤のもたつき

1.4.2　規模

1.4.2.1　実験対象の規模

形式記述の対象とした文書の規模は、次のとおりである。

表 1-8 実験対象の規模

	個数	ページ数	
業務	5	100	20 ページ/業務
テーブル	1	3	
状態遷移図	1	7	※
		計 110	

※：状態遷移図のページ数には、状態遷移図に関連する補足情報のページも含む

なお、形式記述の対象とはしなかったが、本実験で形式化範囲を絞り込むために読んだページ数は 707 ページ、読み飛ばしたページ数も含めた全ページ数は 16,320 ページである。

1.4.2.2 作成した形式記述の規模

本実験で作成した形式記述の規模は、次のとおりである。

表 1-9 形式記述の規模

	step1	step2	step3	step4	全体※2
行数※1	142	458	614	1,012	1,084
イベント数	7	10	13	13	13
変数数	3	6	7	8	8
不変条件数	4	3	1	53	61
証明課題数	32	62	15	739	848
自動的に証明された証明課題数	32	62	15	680	789
対話的に証明した証明課題数	0	0	0	59	59
自動的に証明された証明課題の割合	100%	100%	100%	約 92%	約 93%
対話的に証明した証明課題の割合	0%	0%	0%	約 8%	約 7 %

※1：Rodin Platform の Pretty Print に表示されたものを計測。コメントは含まない。

※2：全体の行数は「step4 の MACHINE の行数」+「step1～3 の INVARIANTS の行数」+「全 step の CONTEXT の行数」で計算。全体のイベント数と変数数は、最後のリファインメントである step4 の値を採用。全体の不変条件数と証明課題数は、全 step の合計で計算。

1.4.2.3 生産性

外部設計書の規模と形式手法の適用にかかった工数との関係を次に示す。

- 形式記述の生産性[6] ＝ 107.5 [人H] ÷ 110 [ページ] ≒ 0.98 [人H／ページ]
- 仕様理解の生産性[7] ＝ 107.5 [人H] ÷ 707 [ページ] ≒ 0.15 [人H／ページ]

[6] 形式記述の対象とした 110 ページをベースとした生産性

1.4.3　形式手法による指摘事項

1.4.3.1　指摘事項種別

本実験において、指摘した指摘事項の数を表 1-10 に示す。

表 1-10 指摘事項種別

指摘事項種別※	個数
設計内容の衝突	2
設計内容の不足	3
設計内容の曖昧さ	0
誤字・脱字	1
	計 6

※：指摘事項種別

- 設計内容の衝突：設計書に書かれている内容に矛盾・不整合があること
- 設計内容の不足：本来設計書に書かれるべきことが書かれていないこと
- 設計内容の曖昧さ：同一のものを複数の表現で記述する表記ゆれや、複数の解釈ができる文が存在すること
- 誤字・脱字：誤字・脱字などの表記に関する指摘

1.4.3.2　作業と指摘事項の関係

指摘事項が、形式手法を適用する際のどの作業で見つかったかを表 1-11 に示す。

表 1-11 作業と指摘事項の関係

作業	本実験における指摘事項の個数				
	合計	設計内容の衝突	設計内容の不足	設計内容の曖昧さ	誤字・脱字
Event-B の要素の抽出	3	1	1	0	1
リファインメント戦略の立案	0	0	0	0	0
形式記述の作成	1	0	1	0	0
形式記述の検証	2	1	1	0	0
形式記述の修正	0	0	0	0	0

[7] 形式化範囲を絞り込むために読んだページを含んだ 707 ページをベースとした生産性

1.4.3.3　指摘事項の抽出効率

指摘事項の個数と形式手法の適用にかかった工数との間には次の関係がある。

- 抽出効率 ＝ 107.5 [人 H]／6[個数] ≒ 17.9[人 H／個数]

1.4.4　抽出した指摘事項の例

本節では、本実験で抽出した六つの指摘事項のうち、三つをピックアップして、その詳細を説明する。

なお、ピックアップしなかった残りの三つの指摘事項は、いずれも Event-B を適用する最初の段階である「Event-B の要素の抽出」の作業中に抽出されたものである。すなわち、実験の作業者が、Event-B を適用しようという観点でレビューしたことにより抽出された指摘事項である。

1.4.4.1　実行されない処理

本件は、Event-B のイベントを記述する際、そのガード条件に矛盾を書かざるをえない状況に陥ったことをトリガーに指摘されたものである。

本件の具体的な事象について、表 1-12 を用いて説明する。ポイントは、フローチャートにある「公開状態チェック」という分岐処理と、「更新処理 5」という更新処理である。一つ目のポイント「公開状態チェック」は、入力された「書類」の属性「公開状態」の値を参照し、その値によって分岐させる処理と定義されている。なお、本件に関係するケースは、「公開状態」の値が「公開」である場合であったので、そのケースのみを表 1-12 に記載している。二つ目のポイント「更新処理 5」は、現在のデータベース上に存在する「書類」のうち、表 1-12 の「画面アクション明細」に書かれている条件にマッチする「書類」を更新する処理である。なお、どのように更新するかについては、本件とは関係ないため、表 1-12 には記載していない。

表 1-12 実行されない処理

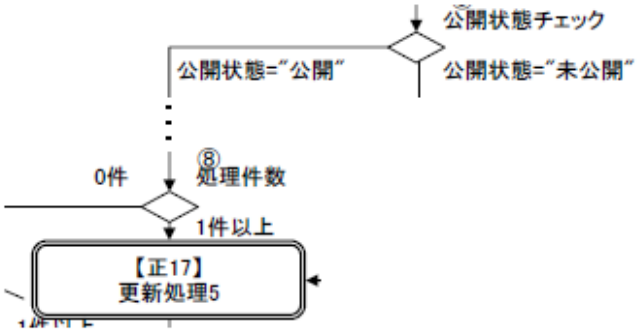
フローチャート	
画面アクション明細	更新処理 5 の実行条件： 1. 書類種別ID=(入力値)書類種別ID And 2. 書類ID=(入力値)書類ID And 3. 版数=(入力値)版数 And 4. 公開状態=非公開 And 5. 削除状態＝削除済

表 1-12 のように二つ並べて書けば、「公開状態」の値が「公開」であるという分岐が先にあり、「公開状態」の値が「非公開」という条件の処理が後にあるため、「更新処理 5」が常に失敗することは一目瞭然だが、実際の設計書ではこれらが離れた場所に書かれているため、レビューで見逃すことも十分あり得る。

次に、本件をどのように発見したかについて説明する。Event-B では両方を、イベントのガード条件という欄に、例えば次のように記述することになる。

doc_id ↦ version ↦ 公開 ∈ 公開状態

doc_id ↦ version ↦ 非公開 ∈ 公開状態

ここでは、入力された「書類種別 ID」と「書類 ID」を「doc_id」、「版数」を「version」とし、「公開状態」という変数の型を、「doc_id」と「version」のペアをキー、公開状態の値をバリューとしたマップとして定義している。上記 Event-B の記述をプログラミング言語風に書き換えると、次のようなイメージである。

```
公開状態.get(doc_id, version) == 公開 &&
公開状態.get(doc_id, version) == 非公開
```

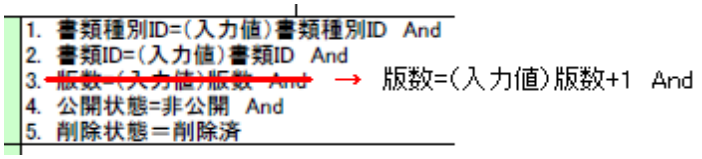
このような条件を記述するタイミングで、矛盾した条件を書いていることに気付き、本件の発見に至った。

なお、本件に対する評価者の評価は、次のようなものであった。

- 画面アクション明細上の「版数=（入力値）版数」が、実際は「版数=（入力値）版数+1」であるという記述の誤りに起因する問題であり、この指摘は改善事項と認められる。
- ただし、本件は設計後からシステム稼働前の間で解決されている。

参考までに、この指摘に対する「画面アクション明細」の改善結果は、表 1-13 のようになる。

表 1-13 指摘事項改善後のアクション詳細定義

画面アクション明細	更新処理 5 の実行条件： 

すなわち、本来あるべき姿を Event-B にすると、次のようになる。

doc_id ↦ version ↦ 公開 ∈ 公開状態

doc_id ↦ version + 1 ↦ 非公開 ∈ 公開状態

1.4.4.2 未定義の状態への状態遷移

本件は、作成した形式記述に対してモデル検査を実施した結果、不変条件に違反する反例が示されたことをトリガーに指摘されたものである。

本件の具体的な事象について、表 1-14 を用いて説明する。ポイントは、「画面アクション明細」に書かれた「公開状態」と「削除状態」に対する属性変更の仕様と、「状態遷移図」に書かれた状態が取り得る「公開状態」と「削除状態」の値の組み合わせである。

表 1-14 未定義の状態への状態遷移

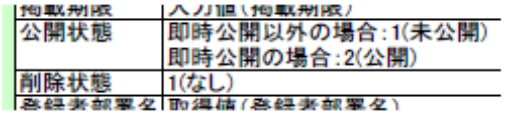
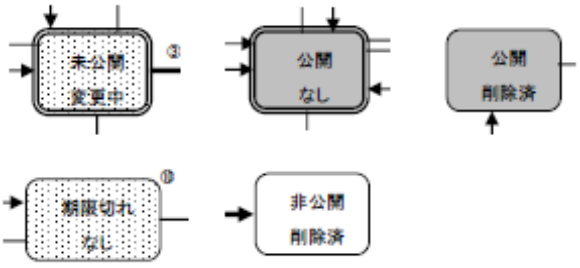
画面アクション明細	属性変更の仕様： 
状態遷移図	状態の定義：  ※上段は公開状態、下段は削除状態

表 1-14 の「画面アクション明細」より、アクション実行後の「公開状態」と「削除状態」の組み合わせには、次の二通りがあることがわかる。

- 未公開、なし
- 公開、なし

次に表 1-14 の「状態遷移図」から状態の定義を列挙すると、次の五通りの「公開状態」と「削除状態」の組み合わせがあることがわかる。

- 未公開、変更中
- 公開、なし
- 公開、削除済
- 期限切れ、なし
- 非公開、削除済

ここで、「画面アクション明細」から得られた「公開状態」と「削除状態」の組み合わせと、「状態遷移図」から得られた「公開状態」と「削除状態」の組み合わせを見比べてみると、「状態遷移図」には「未公開、なし」の組み合わせが定義されていないことがわかる。つまり、このアクションは、「状態遷移図」に定義されていない状態への遷移を許していることになる。

本件は、Rodin Platform の ProB プラグインを使ったモデル検査により、図 1-13 のような反例として示された。図 1-13 の左列はイベント、中列は不変条件、特に Value が⊥となっているものは違反が起きた不変条件、右列は実際に違反が起きるまでのトレースが、下から上の順にフローする書式で書かれている。このトレースを見ると、初期状態から Add1 イベントを実行し、Change4-2 イベントを実行するという手順で、本件を再現できることがわかる。この Change4-2 イベントが、表 1-14 の「画面アクション明細」をもとに記述したイベントであったこと、違反として示された不変条件が、表 1-14 の「状態遷移図」をもとに記述した不変条件であったことから、本件の発見に至った。

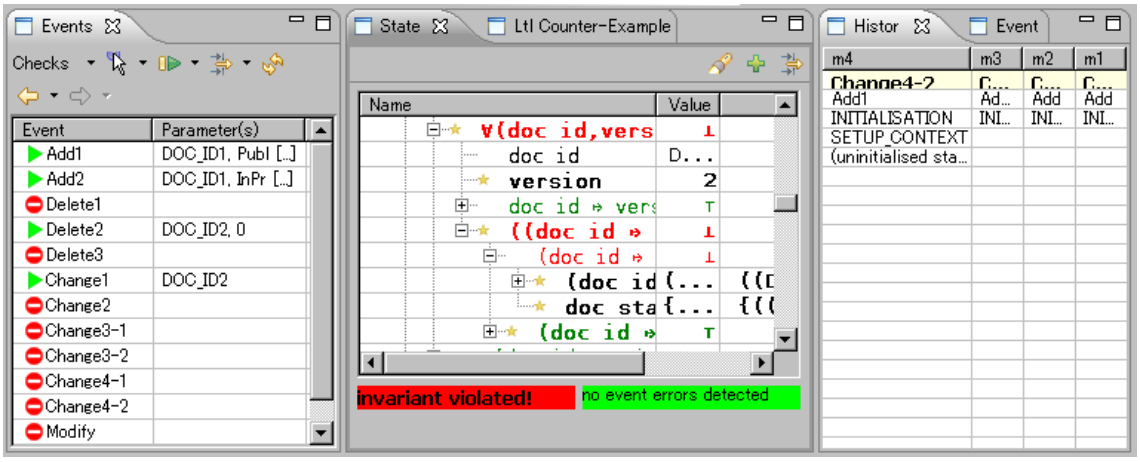
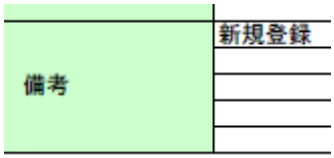
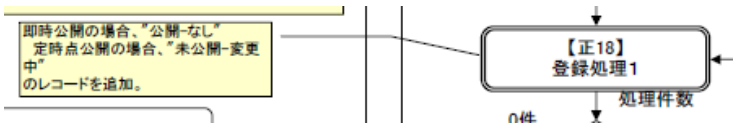
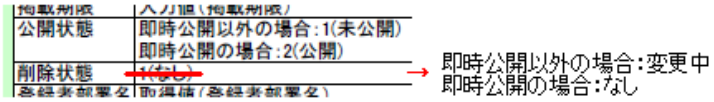


図 1-13 モデル検査の結果

参考までに、このルールに従った読み替えを行うと表 1-15 の「読み替えルールを適用した結果」のようになる。

表 1-15 読み替えルール

画面アクション明細	**備考欄：** 
フローチャート	**吹き出し：**  **※“公開状態—削除状態”の書式で書かれている**
読み替えルールを適用した結果	**属性変更の仕様：**  **※即時公開以外＝定時点公開**

読み替えルールを適用した結果に従えば、アクション実行後の「公開状態」、「削除状態」の組み合わせは次のようになり、必ず「状態遷移図」に定義された状態に遷移するように設計されていることがわかる。

- 未公開、変更中
- 公開、なし

1.4.4.3　未定義のデータに依存した処理

本件は、作成した形式記述に対して定理証明を実施した結果、証明ができなかったことをトリガーに指摘されたものである。

本件の具体的な事象について、表 1-16 を用いて説明する。ポイントは、「状態遷移図」には最新版である N 版と、最新版の一つ前の版である N-1 版の状態についてのみ定義されている点と、表 1-16 のアクションには、二つ前の版である N-2 版の状態が影響するという点である。

表 1-16 未定義のデータに依存した処理

事前状態	（N-2 版） （未定義）	N-1 版 公開/変更中	N 版 未公開/変更中
アクション	↓ （何もしない） ↓	↓ 更新 ↓	↓ 更新 ↓
事後状態	N-2 版 （未定義）	N-1 版 公開/なし	（N 版） 非公開/削除済

※事前状態と事後状態の表記は、上段が「版数」、下段が「公開状態/削除状態」
※版数がカッコ書きになっているものは、「状態遷移図」の定義の範囲外

表 1-16 は、最新版である N 版を削除し、N-1 版を最新版に戻すアクションを表している。ここで、「状態遷移図」では N 版と N-1 版の状態についてのみ定義しており、N-2 版の状態は未定義であることから、このアクションを実行する時点においては、N-2 版の状態は特定できないことに着目してほしい。この状態で N 版を削除し、N-1 版を最新版に戻すと、N-2 版が最新版の一つ前の版として表に出ることになる。つまり、「状態遷移図」に定義された N 版と N-1 版の関係を、アクション実行後は N-1 版と N-2 版が満たさなければならなくなる。それにも関わらず、表 1-16 のアクション定義では、N-2 版に対する状態の変更操作を行っていないため、N-2 版が未定義のまま表に出ることになり、N-1 版と N-2 版が「状態遷移図」の関係を満たさなくなる可能性が出てくる。

本件は、Rodin Platform を使った定理証明により、図 1-14 のように証明できない証明課題の存在として示された。左列が証明木、中列上段が証明に使える定理、中列中段が証明しようとしている論理式、中列下段は操作用のパネル、右列が証明課題の一覧である。図 1-14 では、`Delete3` イベントにおける不変条件 `inv26` を証明しようとしている。中列中段にある `version1 - 2` という文字列が N-2 版のことを表しており、証明しようとしている論理式が N-2 版の状態を証明しなければならないことを示している。ところが、N-2 版は「状態遷移図」に定義されていないため、証明に必要な定理が足りず、証明することができない。ここで証明できなかった `Delete3` イベントが表 1-16 のアクションであり、不変

条件 inv26 が、最新版が「公開/なし」という状態のときの最新版の一つ前の版の状態を定義したものであったことから、本件の発見に至った。

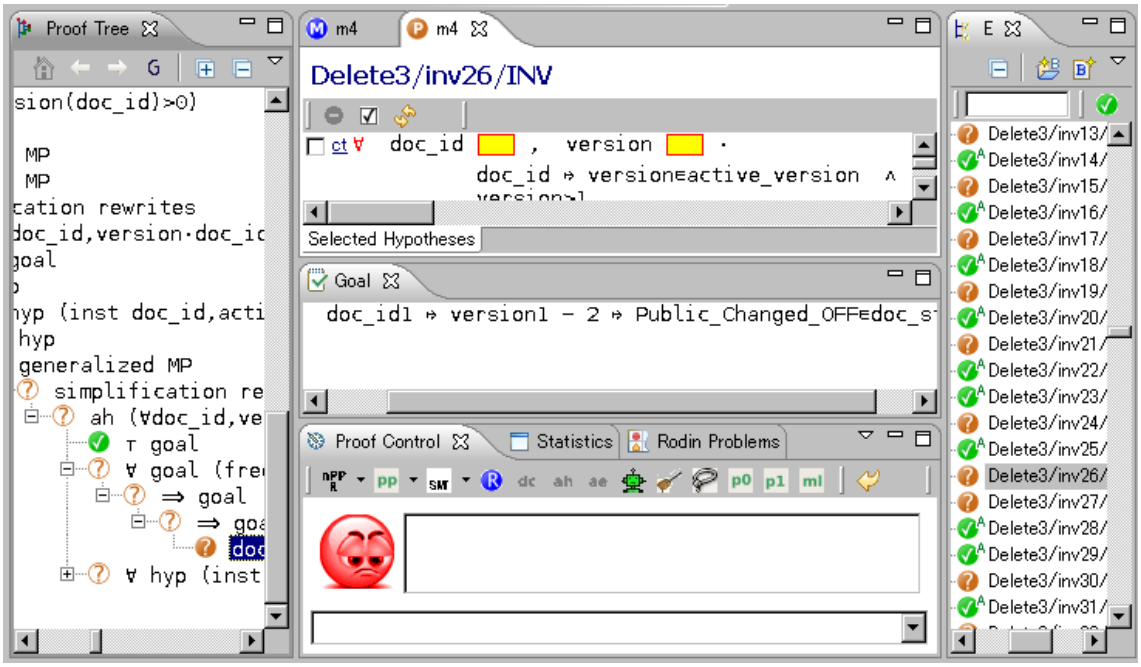


図 1-14 定理証明の結果

1.5 考察

本章では、本実験で行った作業とその結果から得られた知見をもとに、五つの考察について述べる。

1.5.1 経験の少ない作業者でも適用効果は出る

本実験の大きな特徴は、表 1-2 にあるように作業者の実務経験が浅く、しかも一名しかいなかったにもかかわらず、開発者によるレビュー済みの外部設計書から、表 1-10 にあるように、誤字脱字などの些細な指摘を除いても 5 件の改善事項を指摘したことである。0 節や 1.3.3 節にあるような実験上のミスもいくつかあり、決して形式手法の最適な適用事例とは言えない中、こうした結果が出た事は、形式手法の適用が、たとえ上手くないやり方であっても、やりさえすればそれなりの結果が出るという事実と、条件が良ければさらに大きな効果が出ることへの期待を示したと言える。

1.5.2 部分的な適用であっても効果は得られる

本実験では、限られた期間、限られた作業者で実験を終わらせるため、多くの捨象を行

った。この事情は、開発現場の実状と変わらず、実開発に形式手法を適用する場合にも同様に、形式手法を適用する箇所、しない箇所を選択することになるであろう。そのような部分的な適用であっても、本実験で実証されたように、改善事項の指摘効果は得られることが、本実験により実証された。

1.5.3 検証手段の多様さは効果的である

本実験では、形式記述の作成までに行ったレビュー[8]で 4 件、モデル検査で 1 件、定理証明で 1 件の改善事項を見つけている。実施の順はレビュー、モデル検査、定理証明であるため、モデル検査で見つけた 1 件はレビューでは指摘できなかった改善事項、定理証明で見つけた 1 件はレビューでもモデル検査でも指摘できなかった改善事項であると言える。

おそらくモデル検査や定理証明にかけた分の時間をレビューに費やしても、同じ改善事項を指摘できなかったであろう。本実験でモデル検査や定理証明で見つけた改善事項は、複数の設計書を跨いだ二点間の不整合であったり、比較的複雑なロジックを伴うものであったため、レビューで見つけるにはそれなりの「意気込み」や「レビューのスキル・経験」が必要である。仮にレビューの時間が与えられても、レビューで見つけやすい改善事項、例えば些細な誤字脱字などが目につき、深いレビューにまで行きつかなかったかもしれない。

1.5.4 工数改善の余地は十分にある

本実験にかかった工数には、他者の作った設計書を理解するための工数といった、開発者が自ら形式手法を適用する場合にはかからない工数が、およそ 17 人 H 分、含まれている。この分の工数は、何の施策も必要とせず、実開発に適用する時点で軽減される。

また、実験上のミスによる工数増や、ツールの不具合による工数増が非常に多く、プロセスの最適化やツールの成熟により、大幅な工数改善が見込める。正確な値は計測できていないが、形式記述の検証にかかった工数のうち、およそ 35 人 H 強は、改善の見込みがある工数であろう。

すなわち、作業者の能力的な問題を除いても、本実験はおよそ 55.5 人 H の工数にまでは改善の見込みがあったと言える。

1.5.5 実体験により教育効果が得られる

作業者自身の実体験として、本実験で「本物[9]の仕様書」を対象にEvent-Bの適用を経験したことで、Event-Bでどのようなことが書けるか、どのような対象であれば効果的かといっ

[8] 一般的なレビューの効果に加え、形式記述の作成による、いわゆる「用語辞書の作成」の効果や、ツールによる文法チェックの効果が含まれている。
[9] 実験用に作られたものではなく、実務で作られたものであるということ。

た暗黙知を得られたと感じる。暗黙知と書いたのは、例えば「Webシステムに適用できる」だとか「スマートフォン向けのアプリケーション開発に適用できる」といったような、わかりやすい言葉で言い表せないためである。少なくとも、今後別の対象にEvent-Bを適用する場合には、本実験との共通点や差異をベースに、適用可能性や適用効果を予測できるようにはなったと言える。

1.6 まとめ

本実験は、形式手法の一つである Event-B が、外部設計書の改善事項の指摘に役立つかどうかを確認することを目的とし、外部設計書に書かれたシステムのアクションが、同じく外部設計書に書かれたデータの制約を守っているかどうかを検証するというシナリオで、Event-B の適用実験を行った。以降に、本実験の結果をまとめる。

✧ 形式手法で指摘事項を抽出できた

本実験では、110 ページの外部設計書（表 1-8）から、6 件の指摘事項（表 1-10）を、107.5 人 H の工数（表 1-7）で抽出した。また、6 件のうち 2 件は、形式手法に特有のモデル検査、定理証明を使って指摘している（0 節、1.4.4.3 節）。

✧ 決められた手順に従って形式手法を適用できた

本実験では、『形式手法適用手順（Event-B 編）【リリース版】』 [1]をもとにした手順（1.3.1 節）に従って作業を実施し、結果を出した。このことは、形式手法の適用において、決まりきった手順が少なくとも一例あり、それを行使することで結果が出せることを示している。

✧ 多くの知見が得られた

本実験から、1.5 節にあるように、次のような五つの知見が得られた。

- 経験の少ない作業者でも適用効果は出ること（1.5.1 節）
- 部分的な適用であっても効果は得られること（1.5.2 節）
- 検証手段の多様さは効果的であること（1.5.3 節）
- 工数改善の余地は十分にあること（1.5.4 節）
- 実体験により教育効果が得られること（1.5.5 節）

2. Event-B を使用した実験（B2 チーム）

2.1 目的

本実験において、何を確認するために、どのような工程成果物に対して、どの形式手法を用いたかを表 2-1 に記す。

表 2-1　形式手法適用の概要

採用した手法	Event-B	
（特徴）	Event-B では、不変条件を守った状態から、あるイベントを１回実行し、実行後の状態が不変条件を守っていることを検査できる。	
確認すること	業務におけるデータの処理や画面の処理（画面項目定義や入力チェック等）が、データの構造と値に関する制約を守る。	
（確認方法）	データに関する制約と画面のボタン押下（それに伴う業務機能も含む）をそれぞれ Event-B の「不変条件」「イベント」と捉えることにより、一連の画面処理や複数の業務機能が DB の一貫性を守ることを記述・検証する。	
確認の対象	**成果物**	**形式記述**
	基本設計書・画面	Event-B の イベント
	基本設計書・画面、エンティティ定義、ER 図	Event-B の 変数、不変条件
（備考）	上記は「機能要件の合意形成ガイド」 [2]における下記の技術領域・工程成果物に相当する。 ・画面編の画面アクション明細、画面遷移、及びシステム振舞い編のシステム化業務説明 ・画面編の画面入出力項目一覧、及びデータモデル編の ER 図、エンティティ定義	

2.2 体制

本実験を行った作業者の役割やスキルなどを表 2-2 に記す。

表 2-2　作業者の役割とスキル

作業者	役割*	スキル　（経験）			
		業務 AP 開発	類似 AP 開発	形式手法	採用手法
A	形式記述者	15 年	0 年	3 年	3 年
B	形式記述者	0 年	0 年	1 年	1 年
C	形式検証者	5 年	0 年	7 年	1 年

＊ 役割とその担当する作業は以下のとおり。

- 形式記述者 ： 『形式記述』の作成と修正を行う
- 形式検証者 ： 『形式記述』に関する検証を行う

2.3 作業

2.3.1 作業の概要

本実験では、図 2-1 に示すような一連の作業を行った。

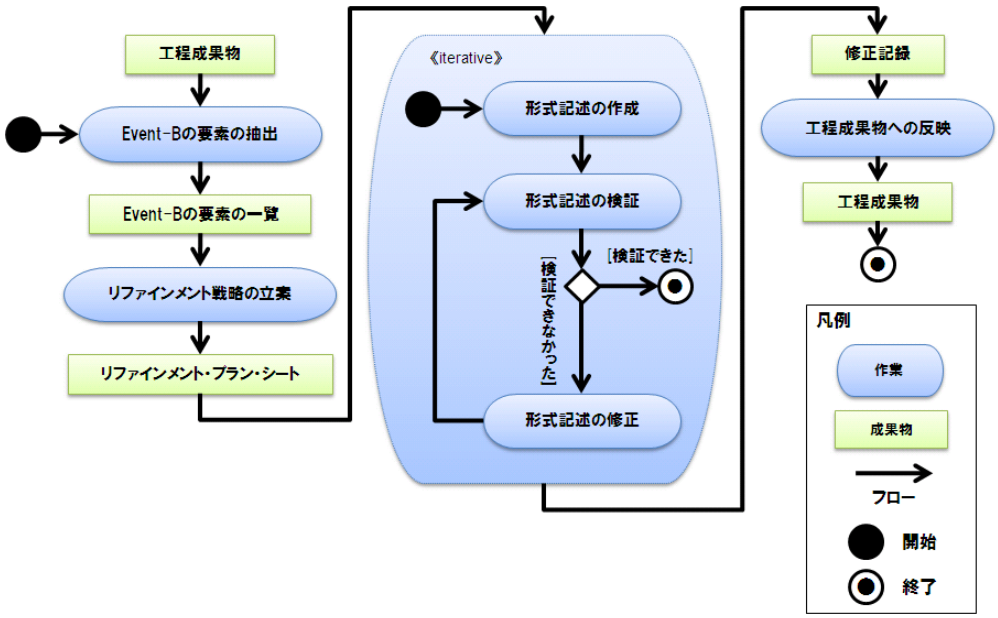


図 2-1　作業の概要

各作業の内容については『形式手法適用手順（Event-B 編）【リリース版】』 [1]（適用法 2）を参照のこと。

2.3.2　関係と多重度の記述

✧ **課題**

本実験の対象となった ER 図では、本来存在するエンティティ間の関係や関係上の多重度が、表記の簡略化等を理由に省略されている場合がある。データの整合性を検証するためには、省略された関係や多重度を推測してから形式記述を行う必要がある。

✧ **対策**

エンティティ間で同一のデータ項目を持つ場合、それらのエンティティ間には関係があると考えられため、それを形式記述上の不変条件として抽出する。関係上の多重度については業務内容から類推することで、不変条件へ反映させる。省略されたエンティティ間の関係と多重度の例を図 2-2 に示す。

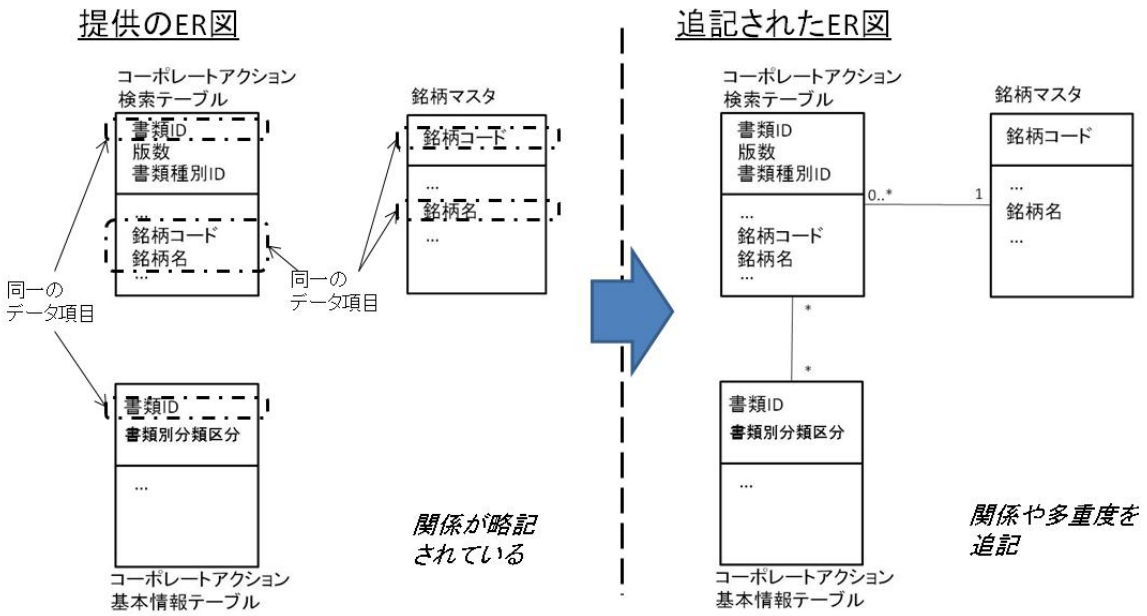


図 2-2 エンティティ間の関係や多重度の追記

✧ **結果**

適用法に従いデータ制約が保存されることを検証するための形式記述が作成可能となった。

2.3.2.1 Event-B の形式記述

前節で述べた対策に従って作成した Event-B の形式記述（変数定義、事前条件定義）を図 2-3 に示す。

```
MACHINE
    m0
SEES
    c1
VARIABLES
    corp        //    エンティティ(コーポレートアクション情報)
    shoruiID        //    属性(書類 ID)
    shoruishubetsuID        //    属性(書類種別 ID)
    hansu        //    属性(版数)
    koukaijoutai        //    属性(公開状態)
    sakujojoutai        //    属性(削除状態)
    meigaracode        //    属性(銘柄コード)
    meigaramei        //    属性(銘柄名)
    corptsuchi        //    エンティティ(コーポレートアクション通知種別)
    shoruiIDT        //    属性(書類 ID)
    tsuchishubetsu        //    属性(通知種別 ID)
    meigara        //    エンティティ(銘柄マスタ)
    meigara_rel        // 関係(コーポレートアクション情報と銘柄マスタ)
    currentScreen        //    現画面
    a0201Meig□ramei        //    画面項目(a0201: 銘柄名)
    a0201MeigaraID        //    画面項目(a0201: 銘柄コード)
    a0201Tsuchishubetsu        //    画面項目(a0201: 通知種別)
    a0202Meigaramei        //    画面項目(a0202: 銘柄名)
    a0202MeigaraID        //    画面項目(a0202: 銘柄コード)
    a0202Tsuchishubetsu        //    画面項目(a0202: 通知種別)
    a0901SearchResult        // 画面項目(a0901: 銘柄名検索結果)
    a0901SelectedIte        //画面項目(a0901: 選択された銘柄)
INVARIANTS
    inv1    :    corp ⊆ CORP        //型定義
    inv2    :    shoruiID ∈ corp ↣ SHORUIID        // 型定義
    inv3    :    shoruishubetsuID ∈ corp ↣ SHORUISHUBET        //
```

```
                SUID                                                コーポレートアクション情報の型定義（属性）
inv4    :   hansu ∈ corp → ℕ                    //  コーポレートアクション情報の型定義（属性）
inv5    :   koukaijoutai ∈ corp → KOUKAI        //  コーポレートアクション情報の型定義（属性）
inv6    :   sakujojoutai ∈ corp → SAKUJO        //  コーポレートアクション情報の型定義（属性）
inv7    :   meigaracode ∈ corp → MEIGARACODE    //  コーポレートアクション情報の型定義（属性）
inv8    :   meigaramei ∈ corp → MEIGARAMEI      //  コーポレートアクション情報の型定義（属性）
inv9    :   corptsuchi ⊆ CORPTSUCHI             //  コーポレートアクション通知情報の型定義（エンティティ）
inv10   :   shoruiIDT ∈ corptsuchi ↣ SHORUIID   //  コーポレートアクション通知情報の型定義（属性）
inv11   :   tsuchishubetsu ∈ corptsuchi → TSUCHISHUBETSU    //  コーポレートアクション通知情報の型定義（属性）
inv12   :   meigara ⊆ MEIGARA                   //  銘柄マスタの型定義（エンティティ）
inv13   :   meigara_rel ∈ corp → meigara        //  関係（コーポレートアクションと銘
```

```
                                                    柄マスタ)の型定義
    inv14   :   currentScreen ∈ SCREEN          //   現画面の型定義
    inv15   :   a0201MeigaraID ∈ MEIGARACODE    //   画面項目
                                                    の型定義
    inv16   :   a0201Meigaramei ∈ MEIGARAMEI    //   画面項目
                                                    の型定義
    inv17   :   a0201Tsuchishubetsu ∈ TSUCHISHUBET  //
                SU                                  画面項目の
                                                    型定義
    inv18   :   a0202MeigaraID ∈ MEIGARACODE    //   画面項目
                                                    の型定義
    inv19   :   a0202Meigaramei ∈ MEIGARAMEI    //   画面項目
                                                    の型定義
    inv20   :   a0202Tsuchishubetsu ∈ TSUCHISHUBET  //
                SU                                  画面項目の
                                                    型定義
    inv21   :   a0901SearchResult ⊆ corp        //   画面項目の型定
                                                    義
    inv22   :   a0901SelectedItem ∈ CORP        //   画面項目の型定
                                                    義
```

図 2-3　Event-B の形式記述（変数宣言、不変条件）

2.3.3　イベントの事前条件抽出

✧ **課題**

本実験の対象となったアクション明細は、フローチャート（手続き）の形式で詳細が記述されている。この形式では、Event-B による形式記述のスタイル（イベントのガード条件、アクション記述）との乖離がある。

✧ **対策**

アクション明細のフローチャートの各パスを抽出し、各々のパスに対応するイベントを用意した。各パスを通過するための画面項目や入力値の条件をガード条件とし、パス通過後の DB 項目の値設定や次画面の画面項目への値設定をアクション記述とした。たとえば図 2-4 では、業務データを DB に登録したことを通知するメーリングリストの有り無しなどに応じて、3 つのイベント A0202JikkouB1、A0202JikkouB2、A0202JikkouB3 を抽出している。これらは実行ボタン押下時のアクション明細に対応しており、A0202JikkouB1 は正常に業務

データが登録される場合、A0202JikkouB2 と A0202JikkouB3 は登録前のデータチェック等により業務データが登録されない場合に相当する。

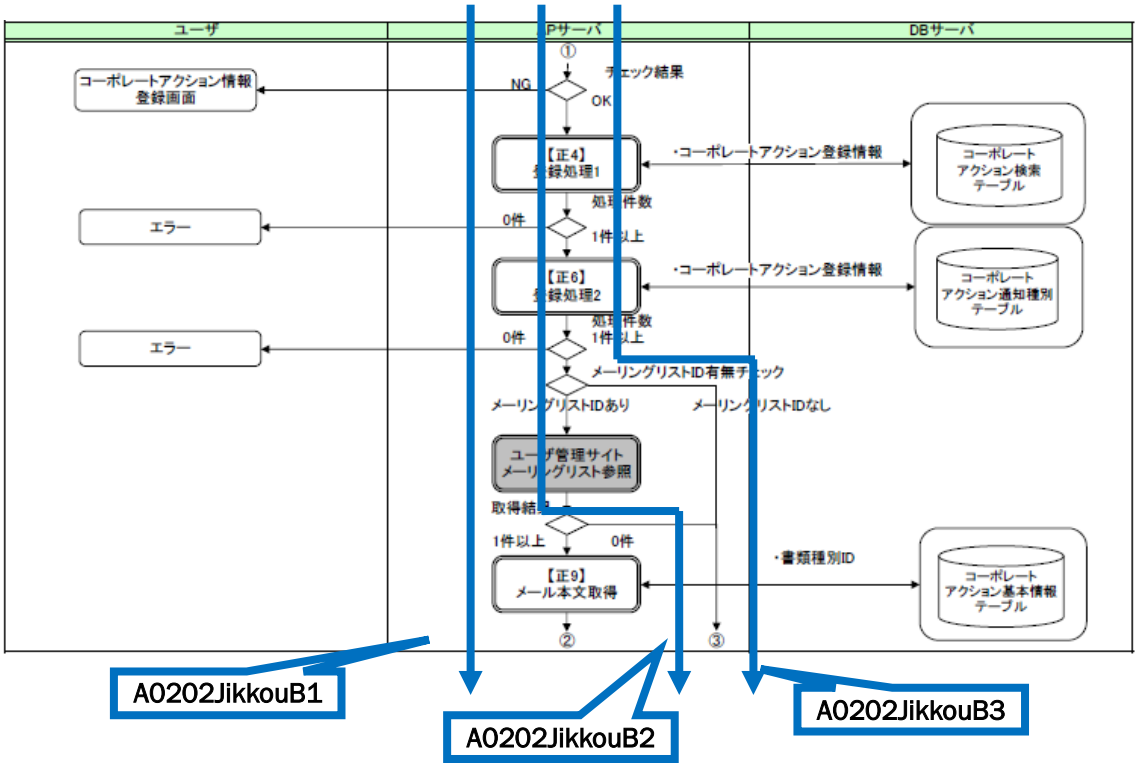


図 2-4 イベントとなる設計情報

✧ 効果

フローチャート形式のアクション詳細から事前条件とアクション記述を抽出することが可能となった。加えて、イベントの場合分けを行うことが可能となった。前節の対策に従い Event-B の形式記述した結果（イベント記述）を図 2-5 に示す。

```
A0202JikkouB1    ≙
STATUS
      ordinary
ANY
      i1
      i2
      c1
      c2
      m1
WHERE
```

```
      grd1   :     currentScreen = a0202
      grd2   :     c1 ∈ CORP
      grd3   :     c1 ∉ corp
      grd4   :     i1 ∈ SHORUIID
      grd5   :     i1 ∉ ran(shoruiID)
      grd6   :     i1 ∉ ran(shoruiIDT)
      grd7   :     c2 ∈ CORPTSUCHI
      grd8   :     c2 ∉ corptsuchi
      grd9   :     i2 ∈ SHORUISHUBETSUID
      grd10   :     i2 ∉ ran(shoruishubetsuID)
      grd11   :     m1 ∈ MEIGARA
      grd12   :     m1 ∈ ran(meigara_rel)
THEN
      act1   :     currentScreen ≔ a0203
      act2   :     corp ≔ corp ∪ {c1
      act3   :     shoruiID ≔ shoruiID ∪ {c1 ↦ i1}
      act4   :   shoruishubetsuID ≔ shoruishubetsuID ∪ {c1 ↦ i2
                 }
      act5   :     hansu ≔ hansu ∪ {c1 ↦ 1}
      act6   :   koukaijoutai ≔ koukaijoutai ∪ {c1 ↦ mikoukai}

      act7   :    sakujojoutai ≔ sakujojoutai ∪ {c1 ↦ nashi}
      act8   :   meigaracode ≔ meigaracode ∪ {c1 ↦ a0202Meigara
                 ID}
      act9   :   meigaramei ≔ meigaramei ∪ {c1 ↦ a0202Meigarame
                 i}
      act10   :     corptsuchi ≔ corptsuchi ∪ {c2}
      act11   :     shoruiIDT ≔ shoruiIDT ∪ {c2 ↦ i1}
      act12   : tsuchishubetsu ≔ tsuchishubetsu ∪ {c2 ↦ a0202Ts
                uchishubetsu}
      act13   :    meigara_rel ≔ meigara_rel ∪ {c1 ↦ m1}
END
```

図 2-5 Event-B の形式記述（イベント）

2.4 実験結果

2.4.1 工数

本実験において、各作業にかけた工数を表 2-3 に示す。

表 2-3 各作業の工数

作業名	工数
Event-B の要素の抽出	45.5 人 H
形式記述の作成	17.0 人 H
形式記述の修正	2.0 人 H
工程成果物への反映	0.0 人 H

2.4.2 設計書と成果物

2.4.2.1 規模

本実験で用いた、本書の対象範囲に関する設計書、作成した中間成果物や形式記述の量を表 2-4 に記す。

表 2-4 作成した中間成果物や形式記述の量

入出区分	成果物	規模
入力	基本設計書	頁数 62、機能数 2、画面数 4、エンティティ数 3
中間	Event-B 要素一覧	頁数 10(パワーポイント)
出力	形式記述	行数 443、証明課題数 25 （内、自動証明数 25）

2.4.2.2 作業効率

前節の入力設計書の規模と形式手法の適用にかかった工数との間には以下の関係がある。

- 作業効率(1) ＝ 64.5[人 H]／106 [翻訳した設計書ページ数] ＝ 0.608[人 H／翻訳した設計書ページ数]
- 作業効率(2) ＝ 64.5[人 H]／287 [参照した設計書ページ数] ＝ 0.225 [人 H／参照した設計書ページ数]

2.4.3　形式手法による改善事項

2.4.3.1　改善事項種別

本実験において、検出した改善事項の数を表 2-5 に示す。

表 2-5 検出した改善事項の数

不具合種別*	個数
設計内容の衝突	3 個
設計内容の不足	0 個
設計内容の曖昧さ	0 個
誤字脱字など	0 個
合計	3 個

*改善事項種別

- 設計内容の衝突：（データ構造や判断の条件など）同じ意味を持つべき事柄が記述箇所によって異なる意味をもつように書かれている
- 設計内容の不足：本来あるべき設計がなされていない （複数個所の記述に食い違いがあり、一方の記述の不足により、他方で想定外の状況になることが、想像できる）
- 設計内容の曖昧さ：設計書の書き方が曖昧であるため、人によって異なる解釈が生じる
- 誤字脱字など：誤字・脱字などの表記に関する指摘

2.4.3.2　作業と改善事項の関係

前節の改善事項が、形式手法を適用する際のどの作業で見つかったかを表 2-6 に示す。

表 2-6 作業と改善事項の関係

作業	本実験における改善事項の数			
	合計	曖昧さ	衝突	不足
要素の抽出	0 個	0 個	2 個	0 個
リファインメント計画の策定	0 個	0 個	0 個	0 個
形式記述の作成	0 個	0 個	1 個	0 個
形式記述の検証	0 個	0 個	0 個	0 個
形式記述の修正	0 個	0 個	0 個	0 個
合計	0 個	0 個	3 個	0 個

2.4.3.3　改善事項の指摘効率

前節の改善事項の個数と形式手法の適用にかかった工数との間には以下の関係がある。

- 発見効率（工数ベース）＝3[個数]／64.5[人 H] ＝ 0.05[個数／人 H]
- 発見効率（不具合ベース）＝ 64.5[人 H]／3[個数] ＝ 21.5 [人 H／個数]

2.4.3.4　指摘した改善事項の例

✧ 改善事項の事象

コーポレートアクション情報エンティティの登録について、コーポレートアクション情報登録画面の画面定義書に従って、システム上存在しない"銘柄名","銘柄コード","国名"（以下銘柄情報と呼ぶ）を入力した場合、コーポレートアクション情報確認画面の確認ボタン押下により、上記のような銘柄情報を持つコーポレートアクション情報がコーポレートアクション検索テーブルに登録される。ER 図より、コーポレートアクション検索テーブル内の銘柄情報は銘柄マスタテーブル内に存在するべきなので、前記状態は不正である。

✧ 改善事項の想定原因

登録画面において、銘柄情報は銘柄一覧ポップアップ画面によって取得することが可能であるが、銘柄情報についての画面項目が入力可能となっているために自由に編集されうる。この事項に加え、コーポレートアクション情報登録アクションにおいて、入力値である銘柄情報がシステム上存在するかどうかの入力チェックをしていないことが原因と考えられる。図 2-6 に関係する画面遷移とコーポレートアクション情報登録アクションについて示す。

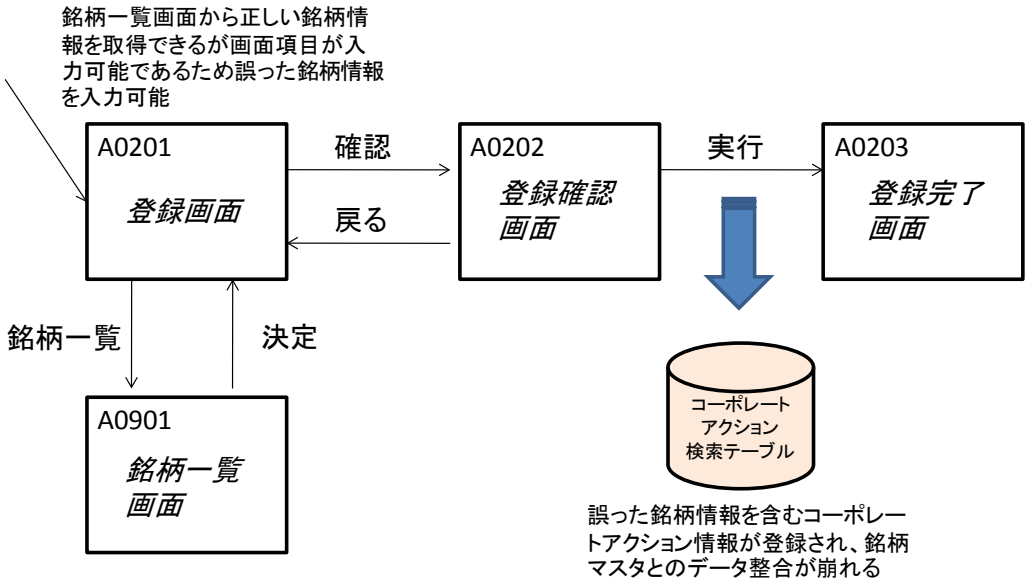


図 2-6　改善事項の想定原因

2.5　考察

2.4.3.4 節で示した改善事項の例は、実際には修正が必要ではないと判断された。これは、提案した改善事項での ER 図で示されるべきデータ制約（多重度）の仮定が実際には誤っていたということに起因する。

本実験は、『形式手法適用手順（Event-B 編）【リリース版】』 [1] の適用法２に基づいているが、2.3.2 節で示した通り、ER 図はしばし関係や多重度が省略されるため、データの制約を形式的に記述するためには、設計記述の不足を補わなくてはならない。その設計記述の不足を補う際に、設計元への確認がうまくいかなかったり形式記述者の思い込みがあった場合、実際の設計情報とは異なる形式記述となってしまい、その後の形式検証の結果が有用なものではないという状況になった。

このような状況は、形式手法適用手順のような既存の設計書から形式記述を作成するようなアプローチでは容易に起こりうる課題であることが、本実験を通じて確認された。また、他の設計記述不足の補足が必要な例は、システムが扱うデータの状態遷移図にも見受けられた。このデータは版（0 以上 99 以下）を伴う状態遷移を持っているが、実際の設計記述では 1 版と 2 版の間での遷移が記述されていた。しかし形式記述を作成する際は、n 版（と n+1 版）をベースとした状態遷移の情報が必要であり、設計情報が不足しているため、この件についても設計元に対しての Q&A が正確に行われなければ正確な形式記述および検証結果が得られないものとなっている。

一方で、本実験で指摘した改善事項におけるデータ制約の仮定が正しかったとすれば、

『形式手法適用手順（Event-B 編）【リリース版】』 [1]の適用法２に基づいて、画面設計（画面項目の制約や画面遷移）やデータ設計の Event-B による形式化が行われ、画面設計とデータ設計の整合性確認を、Event-B の解析によって確認可能であるということが本実験を通して確認された。

2.6 まとめ

本実験では、画面やデータに関する基本設計書から Event-B の形式記述を作成し、画面やデータに関する制約が整合することを Event-B の形式検証によって行った。『形式手法適用手順（Event-B 編）【リリース版】』 [1]の適用法２に従い、画面項目の型に関する制約、入出力可能かに関する制約､ボタン押下時のアクション明細の内容(DB 項目への検索や更新、画面項目への情報転記など)、およびデータ項目の型やエンティティ間の関係を形式記述として反映させ、それらの整合性を形式検証で確認することができた。

3. Event-B を使用した実験（B3 チーム）

3.1 目的

本実験において、何を確認するために、どのような工程成果物に対して、どの形式手法を用いたかを表 3-1 に記す。

表 3-1 実験の目的

採用した手法	Event-B	
（特徴）	Event-B では、不変条件を守った状態から、あるイベントを１回実行し、実行後の状態が不変条件を守っていることを検査できる。	
確認すること	「状態遷移図」に記載された書類の状態遷移に対して、「画面アクション明細」に記載された処理が整合性を保っているかどうかを確認する。	
（確認方法）	「状態遷移図」に記述されている書類の「状態」と「状態を変化させる操作」を Event-B の「公理」とし、「画面アクション明細」を Event-B の「イベント」とし、各々の「イベント」が「状態を変化させる操作」のどれかに対応するということを Event-B の「不変条件」として捉えることにより、「画面アクション明細」に記載された処理が「状態遷移図」に対して整合性を保っていることを検証する。	
確認の対象	**成果物**	**形式記述**
	「状態遷移図」	コンテキストの定数、公理、マシンの不変条件
	「画面アクション明細」	マシンの イベント
（備考）	上記は「機能要件の合意形成ガイド」 [2]における下記の技術領域・工程成果物に相当する。 ・システム化業務フロー ・システム化業務説明 ・画面アクション明細	

3.2 体制

本実験を行った作業者の役割やスキルなどを表 3-2 に記す。

表 3-2　作業者の役割とスキル

作業者	役割*	スキル （経験）			
		業務 AP 開発	類似 AP 開発	形式手法	採用手法
A	形式記述者	13 年	5 年	1.5 年	1.5 年

* 役割とその担当する作業は以下のとおり。

- 形式記述者 ： 『形式記述』の作成と修正を行う
- 形式検証者 ： 『形式記述』に関する検証を行う

3.3 作業

3.3.1 作業の概要

本実験では、図 3-1 に示す一連の作業を行った。

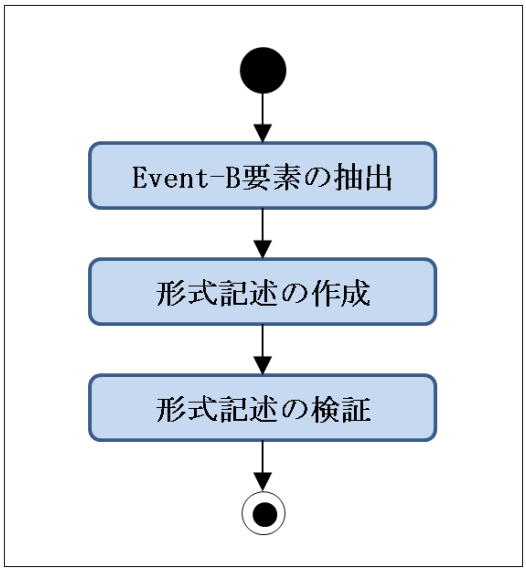


図 3-1　本実験における作業手順

各作業の内容を表 3-3 に記す。

表 3-3　各作業の内容

作業名	内容
Event-B 要素の抽出	・　「状態遷移図」に記述されている「状態」と「状態を変化させる操作」を抽出する。 ・　定義した「状態」と「状態を変化させる操作」に不整合がないことを検証するための要素を抽出する。 ・　「画面アクション明細」から抽出する「イベント」が、「状態遷移図」における「状態を変化させる操作」を実行する「イベント」に対して不整合がないことを検証するための「不変条件」と、それに用いる変数を抽出する。 ・　「画面アクション明細」から「イベント」を抽出する。
形式記述の作成	抽出した Event-B の要素に基づき「形式記述」を作成する。
形式記述の検証	証明課題を証明することにより、「画面アクション明細」に記載された処理が「状態遷移図」に対して整合性を保っていることを検証する。

3.3.2　Event-B 要素の抽出

3.3.2.1　「状態遷移図」からの Event-B 要素の抽出

✧ **課題**

「状態遷移図」では、1 版だけの「状態遷移図」と 1 版から 2 版を生成したときの「状態遷移図」についてだけ記述されているが、3 版以降の任意の版についても整合性の検証を行う必要がある。

✧ **対策**

3 版以降の任意の版についても整合性の検証を行う必要があるという課題に対しては、次の対策を実施した。即ち、1 版から 2 版を生成したときの「状態遷移図」を n 版から n+1 版を生成したときの「状態遷移図」として捉えることにより、3 版以降の全ての版の書類の「状態」と「状態変化」についても Event-B 要素抽出の対象とした。

また、全ての版について形式記述を作成する際、1 版から n 版までの全ての状態遷移を一つの「状態遷移定義」としてまとめて記述する方が合理的である。そのため、次の対策を実施した。即ち、同時に存在する 1 版から n 版までのそれぞれの状態の組を書類の状態として考え、それらを Event-B の「公理」において「状態」として定義した。例えば、図 3-2 に示すように、1 版が s_1、2 版が s_2、…、n 版が s_n という状態であってそれらが同時に可能

な状態であるとすると、$(s_1,s_2,\cdots,s_n)$という組を書類の状態とした。

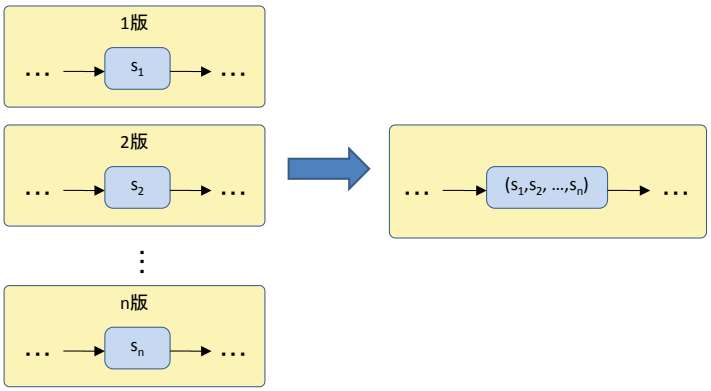


図 3-2　書類の「状態」の表現

また、操作は、同時に存在する 1 版から n 版までの全てに対して同時に作用すると考え、それらを Event-B の「公理」において「状態を変化させる操作」として定義した。例えば、図 3-3 に示すように、操作 a が 1 版に対しては「状態」を s_1 から t_1 に変化させ、同じ操作 a が 2 版に対しては「状態」を s_2 から t_2 に変化させ、3 版以降も同様に、i 版に対しては「状態」を s_i から t_i に変化させる場合を考える。このような操作 a は、書類の「状態」を$(s_1,s_2,\cdots)$から$(t_1,t_2,\cdots)$に変化させる「操作」であると考えた。ここで、操作 a が k 版に対して「状態」を変化させない場合は、$s_k=t_k$ であるとする。

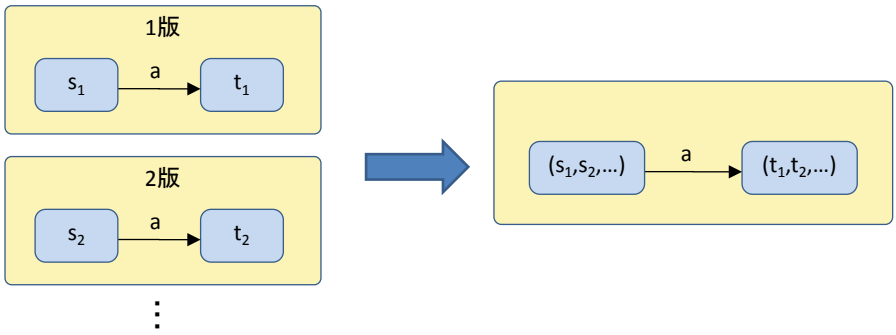


図 3-3　書類の「状態を変化させる操作」の表現

✧ 結果

1 版から n 版までの状態遷移を一つの書類の「状態遷移定義」として考え、書類の「状態」「状態を変化させる操作」を Event-B のコンテキストにおいて「定数」として記述し、それらの「状態」「状態を変化させる操作」が満たすべき条件を「公理」として記述した。

3.3.2.2 「画面アクション明細」からの Event-B 要素の抽出

✧ **課題**

「画面アクション明細」に記載された処理が「状態遷移図」に対して整合性を保っていることを検証するために、「画面アクション明細」からEvent-Bの「イベント」を抽出し、それが、「状態遷移図」における「状態を変化させる操作」に対するイベント（以下「操作イベント」）[10]として正しく状態を変化させているということを証明したい。そのためには、「イベント」の粒度が「操作イベント」の粒度と一致していると都合がよい。しかし、「画面アクション明細」は必ずしも「操作イベント」ごとに記述されているわけではないため、一般には、一つの「画面アクション明細」から複数の「イベント」を抽出する必要がある。

✧ **対策**

「イベント」を抽出する対象として、書類の「状態」を変化させる「画面アクション明細」のみを考えることとした。その上で、「画面アクション明細」が「状態遷移図」において一つの「操作イベント」に対応する場合は一つの「イベント」として抽出した。一方、二つ以上の「操作イベント」に対応する場合は、図 3-4 に示すように「画面アクション明細」における条件分岐などの制御構造に着目し、複数の「イベント」として抽出した。

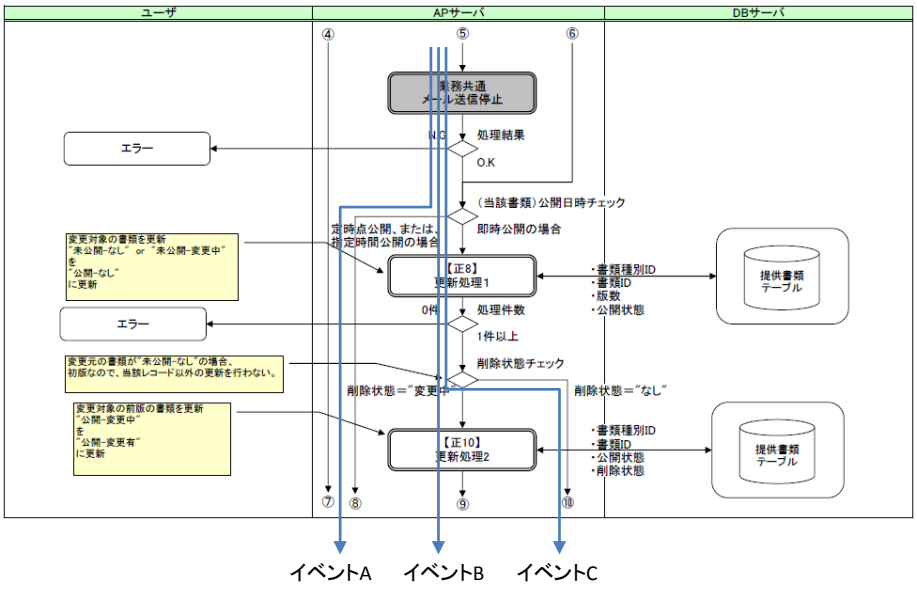


図 3-4 「画面アクション明細」からの「イベント」の抽出

[10] Event-B の構文に含まれる Events と混同することを避けるため、「状態遷移図」における「状態を変化させる操作」を実行する「イベント」をここでは「操作イベント」と呼ぶことにする。

✧ **結果**

「画面アクション明細」から Event-B の「イベント」を抽出し、「定数」である「操作イベント」に対応付けた。また、「画面アクション明細」や関連する設計資料から書類の事前状態と事後状態を抽出し、それらが、「イベント」と対応する「操作イベント」が実行する「状態を変化させる操作」の事前状態と事後状態に一致しているということを「不変条件」として記述した。

3.3.3 形式記述の作成

3.3.3.1 Event-B の形式記述

Event-B 形式記述は、前節までの考え方で抽出した Event-B の要素を用いて表 3-4 に示すように構成した。

表 3-4 Event-B 形式記述の構成

Event-B コンポーネント	識別子	記述内容
CONTEXT	c0	「状態」の Event-B 形式記述
CONTEXT	c1	「状態を変化させる操作」の Event-B 形式記述
CONTEXT	c2	「状態遷移図」の検証のための Event-B 形式記述
CONTEXT	c3	証明に必要な補題の Event-B 形式記述
MACHINE	m0	「画面アクション明細」から抽出した「イベント」を検証するための Event-B 形式記述

✧ **「状態」の Event-B 形式記述**

書類の「状態」は、「公開状態」、「削除状態」、「版数」の組合せで表現した。書類の「状態」を構成するそれらの属性の型を集合として定義し、それらの属性の値を定数として定義した。また、属性の型と値の関係を公理として定義し、書類の「状態」の型（または構造）を属性の組合せで定義した。図 3-5 に Event-B 形式記述を示す。

```
CONTEXT
    c0
SETS                                    書類の「状態」を構成する属性の型
    KOUKAI_FLG          //    公開状態
    SAKUJO_FLG          //    削除状態
```

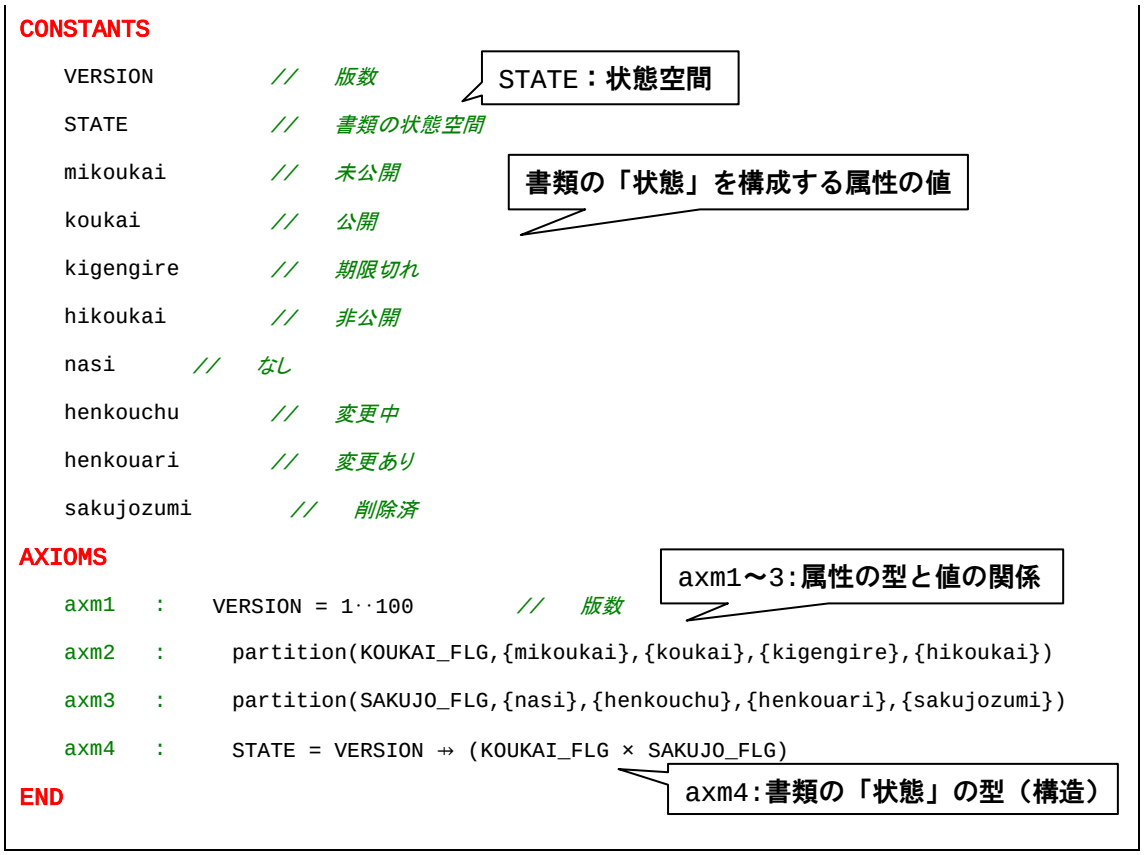


図 3-5 「状態」の Event-B 形式記述

✧ 「状態を変化させる操作」の Event-B 形式記述

書類の「操作イベント」を定数として定義した。また、「状態 A」と「状態 B」の順序対を「状態 A を状態 B に変化させる操作」として考え、それらの各々がある「操作イベント」に所属するという関係を公理として定義した。図 3-6 に Event-B 形式記述を示す。

```
CONTEXT
    c1
EXTENDS
    c0
CONSTANTS
    EV
    evid
    touroku_teijitenKoukai          //    登録(定時点公開)
    touroku_sokujiKoukai            //    登録(即時公開)
    jikanKeika              //    時間経過
    henkou_teijitenKoukai           //    変更(定時点公開)
    henkou_sokujiKoukai             //    変更(即時公開)
    shusei              //    修正
    sakujo              //    削除
    kyoseiSakujo            //    強制削除
AXIOMS
    axm1    :       EV = STATE → STATE
    axm2    :       evid ∈ EV
    axm3    :       evid = STATE ◁ id
    axm4    :       touroku_teijitenKoukai ∈ EV
    axm5    :       touroku_sokujiKoukai ∈ EV
    axm6    :       jikanKeika ∈ EV
    axm7    :       henkou_teijitenKoukai ∈ EV
    axm8    :       henkou_sokujiKoukai ∈ EV
    axm9    :       shusei ∈ EV
    axm10   :       sakujo ∈ EV
    axm11   :       kyoseiSakujo ∈ EV
    axm12   :       ∅↦{1↦(mikoukai↦nasi)}∈touroku_teijitenKoukai
    axm13   :       ∅↦{1↦(koukai↦nasi)}∈touroku_sokujiKoukai
    axm14   :       {1↦(mikoukai↦nasi)}↦{1↦(koukai↦nasi)}∈jikanKeika
    axm16   :       ∀n·n≥2 ⇒ ((1‥n−2×{koukai↦henkouari})∪
                    {n−1↦(koukai↦henkouchu),n↦(mikoukai↦henkouchu)})↦((1‥
                    n−2×{koukai↦henkouari})∪{n−1↦(koukai↦henkouari),n↦(koukai↦nasi)})∈
                    jikanKeika
    axm18       ∀n·n≥1 ⇒ ((1‥n−1×{koukai↦henkouari})∪{n↦(koukai↦nasi)})↦((1‥
```

EV:「操作イベント」の集合

「操作イベント」

axm1:「操作イベント」の集合

axm2～11:「操作イベント」が「操作イベント」の集合に所属する関係

evid は遷移しない操作

axm12～49:「状態 A を状態 B に変化させる操作」が「操作イベント」に所属する関係

: n−1×{kigengire↦henkouari})∪{n↦(kigengire↦nasi)})∈jikanKeika

axm19 : {1↦(mikoukai↦nasi)}↦{1↦(mikoukai↦nasi)}∈henkou_teijitenKoukai

∀n·n≥2 ⇒ ((1‥n−2×{koukai↦henkouari})∪

axm21 {n−1↦(koukai↦henkouchu),n↦(mikoukai↦henkouchu)})↦((1‥

: n−2×{koukai↦henkouari})∪{n−1↦(koukai↦henkouchu),n↦(mikoukai↦henkouchu)})

∈henkou_teijitenKoukai

∀n·n≥1 ⇒ ((1‥n−1×{koukai↦henkouari})∪{n↦(koukai↦nasi)})↦((1‥

axm23 n−1×{koukai↦henkouari})∪{n↦(koukai↦henkouchu),n+1↦(mikoukai↦henkouchu)})∈

: henkou_teijitenKoukai

axm25 ∀n·n≥1 ⇒ ((1‥n−1×{koukai↦henkouari})∪

: {n↦(koukai↦nasi),n+1↦(hikoukai↦sakujozumi)})↦((1‥n−1×{koukai↦henkouari})∪

{n↦(koukai↦henkouchu),n+1↦(mikoukai↦henkouchu)})∈henkou_teijitenKoukai

axm26 : {1↦(mikoukai↦nasi)}↦{1↦(koukai↦nasi)}∈henkou_sokujiKoukai

∀n·n≥2 ⇒ ((1‥n−2×{koukai↦henkouari})∪

axm28 {n−1↦(koukai↦henkouchu),n↦(mikoukai↦henkouchu)})↦((1‥

: n−2×{koukai↦henkouari})∪{n−1↦(koukai↦henkouari),n↦(koukai↦nasi)})∈

henkou_sokujiKoukai

∀n·n≥1 ⇒ ((1‥n−1×{koukai↦henkouari})∪{n↦(koukai↦nasi)})↦((1‥

axm30 n−1×{koukai↦henkouari})∪{n↦(koukai↦henkouari),n+1↦(koukai↦nasi)})∈

: henkou_sokujiKoukai

axm32 ∀n·n≥1 ⇒ ((1‥n−1×{koukai↦henkouari})∪

: {n↦(koukai↦nasi),n+1↦(hikoukai↦sakujozumi)})↦((1‥n−1×{koukai↦henkouari})∪

{n↦(koukai↦henkouari),n+1↦(koukai↦nasi)})∈henkou_sokujiKoukai

axm34 ∀n·n≥1 ⇒ ((1‥n−1×{koukai↦henkouari})∪{n↦(koukai↦nasi)})↦((1‥

: n−1×{koukai↦henkouari})∪{n↦(koukai↦nasi)})∈shusei

axm36 ∀n·n≥1 ⇒ ((1‥n−1×{kigengire↦henkouari})∪{n↦(kigengire↦nasi)})↦((1‥

: n−1×{koukai↦henkouari})∪{n↦(koukai↦nasi)})∈shusei

axm37 : {1↦(mikoukai↦nasi)}↦{1↦(hikoukai↦sakujozumi)}∈sakujo

∀n·n≥2 ⇒ ((1‥n−2×{koukai↦henkouari})∪

axm39 {n−1↦(koukai↦henkouchu),n↦(mikoukai↦henkouchu)})↦((1‥

: n−2×{koukai↦henkouari})∪{n−1↦(koukai↦nasi),n↦(hikoukai↦sakujozumi)})∈

sakujo

axm41 ∀n·n≥1 ⇒ ((1‥n−1×{koukai↦henkouari})∪{n↦(koukai↦nasi)})↦((1‥

: n−1×{koukai↦sakujozumi})∪{n↦(koukai↦sakujozumi)})∈sakujo

axm42 : {1↦(mikoukai↦nasi)}↦{1↦(hikoukai↦sakujozumi)}∈kyoseiSakujo

axm44 ∀n·n≥2 ⇒ ((1‥n−2×{koukai↦henkouari})∪

```
    :       {n−1↦(koukai↦henkouchu),n↦(mikoukai↦henkouchu)})↦(1‥
            n×{hikoukai↦sakujozumi})∈kyoseiSakujo
    axm46   : ∀n·n≥1 ⇒ ((1‥n−1×{koukai↦henkouari})∪{n↦(koukai↦nasi)})↦(1‥
              n×{hikoukai↦sakujozumi})∈kyoseiSakujo
    axm47   :   ∀n·n≥1 ⇒ (1‥n×{koukai↦sakujozumi})↦(1‥n×{hikoukai↦sakujozumi})∈
              kyoseiSakujo
    axm49   : ∀n·n≥1 ⇒ ((1‥n−1×{kigengire↦henkouari})∪{n↦(kigengire↦nasi)})↦(1‥
              n×{hikoukai↦sakujozumi})∈kyoseiSakujo
END
```

図 3-6　「状態を変化させる操作」の Event-B 形式記述

✧ 「状態遷移図」の検証のための Event-B 形式記述

CONTEXT（c0）と CONTEXT（c1）で定義した「状態遷移図」に不整合がないことを検証するための形式記述を定義した。それは、「状態遷移図」に「状態 A」と「状態 B」(「状態 A」＝「状態 B」でも良い）が存在する時、ある順序で並べた「操作イベント」の列を順に実行することにより「状態 A」から「状態 B」に到達可能であるという関係（以下「到達可能関係」)が成立することを証明するための形式記述である。図 3-7 に Event-B 形式記述を示す。

```
CONTEXT
    c2
EXTENDS
    c1
AXIOMS
    axm1    : ∅ ↦ {1↦(hikoukai↦sakujozumi)} ∈ touroku_teijitenKoukai;jikanKeika;jikanK
              eika;shusei;kyoseiSakujo
    axm2    ∅ ↦ {1↦(koukai↦henkouari),2↦(koukai↦nasi)} ∈ touroku_sokujiKoukai;henkou
    :       _teijitenKoukai;sakujo;henkou_sokujiKoukai
    axm3    ∅ ↦ {1↦(koukai↦henkouari),2↦(koukai↦henkouari),3↦(koukai↦nasi)} ∈ tourok
    :       u_sokujiKoukai;henkou_sokujiKoukai;henkou_sokujiKoukai
END
```

axm1〜3：「状態 A」から「状態 B」への「到達可能関係」を theorem として記述
例えば axm3 は状態「いかなる版も存在しない」に対して「登録（即時公開)」「変更（即時公開)」「変更（即時公開)」を続けて実行すると、状態が「1 版：公開-変更有、2 版：公開-変更有、3 版：公開-なし」に変化することを示す。そして、これを証明することが必要になる。

図 3-7　「書類の状態遷移定義」の検証のための Event-B 形式記述

✧ 証明に必要な補題の Event-B 形式記述

証明が円滑に進むために必要な補題を設定した。図 3-8 に Event-B 形式記述を示す。

```
CONTEXT
    c3
EXTENDS
    c2
AXIOMS
    axm1  :  ∀v,s·v⊆VERSION ∧ s∈STATE ⇒ v◁s = {x↦y|x↦y∈s∧x∈v}
    axm2  :  ∀v,s·v⊆VERSION ∧ s∈STATE ⇒ v⩤s = {x↦y|x↦y∈s∧x∉v}
    axm3  :  ∀s,r·s ∈ STATE ∧ r ⊆ KOUKAI_FLG × SAKUJO_FLG ⇒ s▷r = {x↦y|x↦y∈
             s∧y∈r}
    axm4  :  ∀s,r·s ∈ STATE ∧ r ⊆ KOUKAI_FLG × SAKUJO_FLG ⇒ s⩥r = {x↦y|x↦y∈
             s∧y∉r}
    axm5  :  ∀
             s,r·s ∈ STATE ∧ r ∈ KOUKAI_FLG × SAKUJO_FLG ⇒ dom(dom(s)×{r})=dom
             (s)
END
```

axm1～5：証明に必要な補題

図 3-8　証明に必要な補題の Event-B 形式記述

✧ 「画面アクション明細」から抽出した「イベント」を検証するための Event-B 形式記述

「画面アクション明細」から抽出した全ての「イベント」を定義した上で、それらが CONTEXT で定義した「操作イベント」と整合性が取れていることを検証するための「不変条件」と、それに用いる変数を定義した。この「不変条件」は、各々の「イベント」を「操作イベント」に対応付けた上で、「イベント」の事前状態と事後状態のそれぞれと同じ「状態」が CONTEXT で「状態 A」と「状態 B」として定義されており、「状態 A を状態 B に変化させる操作」も CONTEXT に定義されており、さらに、その「操作」が、「イベント」に対応づいた「操作イベント」に所属するという関係である。図 3-9 に Event-B 形式記述を示す。

```
MACHINE
    m0
SEES
    c3
VARIABLES
    state
    statePre
    ev
INVARIANTS
    inv1    :      state ∈ STATE
    inv2    :      statePre ∈ STATE
    inv3    :      ev ∈ EV
    inv4    :      statePre ↦ state ∈ ev
EVENTS
    INITIALISATION    ≙
    STATUS
     ordinary
BEGIN
     act1    :     state ≔ ∅
     act2    :     statePre ≔ ∅
     act3    :     ev ≔ evid
END

    touroku_teijitenKoukai01    ≙          //    登録(指定時間公開)
    STATUS
     ordinary
WHEN
     grd1    :      state = ∅
THEN
     act1    :       ev ≔ touroku_teijitenKoukai
     act2    :       statePre ≔ state
     act3    :       state ≔ {1↦(mikoukai↦nasi)}
END
```

state：「現在の状態」
statePre：「一つ前の状態」
ev：イベントに対応する「操作イベント」

inv1～inv4：「一つ前の状態」を「現在の状態」に変化させる「操作」が、「イベント」に対応付いた「操作イベント」に所属するという関係

act1～act2：「一つ前の状態」と「現在の状態」を（いかなる版も存在しない）初期状態で初期化
act3：「操作イベント」を「状態を変化させない操作」

grd1：「現在の状態」は（いかなる版も存在しない）初期状態

act1：「操作イベント」に「登録_指定時間公開」を代入
act2：「一つ前の状態」に「現在の状態」を代入
act3：「現在の状態」に「1版：未公開-なし」を代入

```
    touroku_sokujiKoukai01    ≙          //    登録(即時公開)
    STATUS
     ordinary
WHEN
     grd1    :     state = ∅
THEN
     act1    :     ev ≔ touroku_sokujiKoukai
     act2    :     statePre ≔ state
     act3    :     state ≔ {1↦(koukai↦nasi)}
END
```

grd1：「現在の状態」は（いかなる版も存在しない）初期状態

act1：「操作イベント」に「登録_即時公開」を代入
act2：「一つ前の状態」に「現在の状態」を代入
act3：「現在の状態」に「1 版：公開

```
    henkou_teijitenKoukai01    ≙          //    変更(指定時間公開、未公開)
    STATUS
     ordinary
ANY
     n
     x3
     s
WHERE
     grd1    :      n ∈ VERSION
     grd2    :      n < 99
     grd3    :      x3 ∈ SAKUJO_FLG
     grd4    :      s ∈ 1‥n−1→KOUKAI_FLG×SAKUJO_FLG
     grd5    :      state = s∪{n↦(mikoukai↦x3)}
THEN
     act1    :      ev ≔ henkou_teijitenKoukai
     act2    :      statePre ≔ state
     act3    :      state(n) ≔ mikoukai↦x3
END
```

grd1～5：「現在の状態」の「n 版」は「未公開」

act1：「操作イベント」に「変更_指定時間公開」を代入
act2：「一つ前の状態」に「現在の状態」を代入
act3：「現在の状態」の「n 版」に「未公開」を代入

```
    henkou_sokujiKoukai01    ≙          //    変更(即時公開、未公開、なし)
    STATUS
     ordinary
ANY
     n
     s
```

```
WHERE
    grd1   :    n ∈ VERSION
    grd2   :    n < 99
    grd3   :    s ∈ 1‥n−1→KOUKAI_FLG×SAKUJO_FLG
    grd4   :    state ∈ STATE
    grd5   :    state = s∪{n↦(mikoukai↦nasi)}
THEN
    act1   :    ev ≔ henkou_sokujiKoukai
    act2   :    statePre ≔ state
    act3   :    state(n) ≔ koukai↦nasi
END
```

grd1～5：「現在の状態」の「n 版」は「未公開-なし」

act1：「操作イベント」に「変更_即時公開」を代入
act2：「一つ前の状態」に「現在の状態」を代入
act3：「現在の状態」の「n 版」に「公開-なし」を代入

```
henkou_sokujiKoukai02   ≙          //   変更(即時公開、未公開、変更中)
STATUS
 ordinary
ANY
    n
    s
    s1
WHERE
    grd1   :    n ∈ VERSIO
    grd2   :    n < 99
    grd3   :    n ≥ 2
    grd4   :    s ∈ 1‥n−1→KOUKAI_FLG×SAKUJO_FLG
    grd5   :    state ∈ STATE
    grd6   :    state = s∪{n↦(mikoukai↦henkouchu)}
    grd7   :    s1 ⊆ s
    grd8   :    ran(s1) = {koukai↦henkouchu}
THEN
    act1   :    ev ≔ henkou_sokujiKoukai
    act2   :    statePre ≔ state
                state ≔ (s\s1)∪
    act3   :    (dom(s1)×{koukai↦henkouari})∪
                {n↦(koukai↦nasi)}
END
```

grd1～6：「現在の状態」の「n 版」は「未公開-変更中」で「1 版～n-1 版」の中で「公開-変更中」のものを含む

act1：「操作イベント」に「変更_即時公開」を代入
act2：「一つ前の状態」に「現在の状態」を代入
act3：「現在の状態」の「公開-変更中」の「版」に「公開-変更有」を代入し、「n 版」に「公開-なし」

```
    henkou_sokujiKoukai02_c    ≙            //    変更(即時公開、未公開、変更中)
    STATUS
     ordinary
ANY
     n
     s1
WHERE
     grd1    :     n ∈ VERSION
     grd2    :     n < 99
     grd3    :     n ≥ 2
                   partition(state,1‥
     grd4    :     n−2×{koukai↦henkouari},{n−1↦(koukai↦henkouchu)},{n↦(mikoukai↦henkouc
                   hu)})
     grd5    :     s1 ∈ STATE          //
     grd6    :     s1 = {n−1↦(koukai↦henkouari),n↦(koukai↦nasi)}
THEN
     act1    :     ev ≔ henkou_sokujiKoukai
     act2    :     statePre ≔ state
     act3    :     state ≔ state   ⊡3 1
END
```

grd1～4：「現在の状態」は「1 版～n-2 版：公開-変更有、n-1 版：公開-変更中、3 版：未公開-変更中」

act1：「操作イベント」に「変更_即時公開」を代入
act2：「一つ前の状態」に「現在の状態」を代入
act3：「現在の状態」の「n-1 版」を「公開-変更有」に、「n 版」に「公開-なし」を代入

```
    henkou_teijitenKoukai02    ≙
    STATUS
     ordinary
ANY
     n
     s1
WHERE
     grd1    :     n ∈ VERSION
     grd2    :     n < 99
     grd3    :     n ≥ 1
                   partition(state,1‥
     grd4    :     n−1×{koukai↦henkouari},{n↦(koukai↦nasi)})
```

grd1～5：「現在の状態」は「1 版～n-1 版：公開-変更有、n 版：公開-なし」であり「n+1 版」は「非公開-削除済」ではない

```
    grd5   :     n+1↦(hikoukai↦sakujozumi)∉state
    grd6   :     s1 ∈ STATE
    grd7   :     s1 = {n↦(koukai↦henkouchu),n+1↦(mikoukai↦henkouchu)}
THEN
    act1   :     ev ≔ henkou_teijitenKoukai
    act2   :     statePre ≔ state
    act3   :     state ≔ state   ◲1
END

   henkou_teijitenKoukai03   ≙
   STATUS
    ordinary
ANY
    n
    s1
WHERE
    grd1   :   n ∈ VERSION
    grd2   :   n < 99
    grd3   :   n ≥ 1
    grd4   :   partition(state,1‥
               n−1×{koukai↦henkouari},{n↦(koukai↦nasi),n+1↦(hikoukai↦sakujozumi)})
    grd5   :   s1 ∈ STATE
    grd6   :   s1 = {n↦(koukai↦henkouchu),n+1↦(mikoukai↦henkouchu)}
THEN
    act1   :   ev ≔ henkou_teijitenKoukai
    act2   :   statePre ≔ state
    act3   :   state ≔ state   ◲
END

   henkou_sokujiKoukai03   ≙
   STATUS
    ordinary
ANY
    n
    s1
WHERE
```

act1：「操作イベント」に「変更_指定時間公開」を代入
act2：「一つ前の状態」に「現在の状態」を代入
act3：「現在の状態」の「n 版」に「公開-変更中」を代入し「n+1」版に「未公開-変更中」を代入

grd1～4：「現在の状態」は「1 版～n-1 版：公開-変更有、n 版：公開-なし、n+1 版：非公開-削除済」

act1：「操作イベント」に「変更_指定時間公開」を代入
act2：「一つ前の状態」に「現在の状態」を代入
act3：「現在の状態」の「n 版」に「公開-変更中」を代入し「n+1」版に「未公開-変更中」を

```
    grd1   :   n ∈ VERSIO
               N
    grd2   :   n < 9
               9
    grd3   :   n ≥
               1
    grd4   :   partition(state,1‥
               n−1×{koukai↦henkouari},{n↦(koukai↦nasi)})
    grd5   :   n+1↦(hikoukai↦sakujozumi)∉stat
               e
    grd6   :   s1 = {n↦(koukai↦henkouari),n+1↦(koukai↦nasi
               )}
THEN
    act1   :   ev ≔ henkou_sokujiKouka
               i
    act2   :   statePre ≔ stat
               e
    act3   :   state ≔ state  [
               1
END
```

grd1～5：「現在の状態」は「1 版～n-1 版：公開-変更有、n 版：公開-なし」であり「n+1 版」は「非公開-削除済」ではない

act1：「操作イベント」に「変更_即時公開」を代入
act2：「一つ前の状態」に「現在の状態」を代入
act3：「現在の状態」の「n 版」に「公開-変更有」を代入し、「n+1 版」に「公開-なし」を代入

```
    henkou_sokujiKoukai04   ≙

    STATUS
     ordinar
     y
ANY
     n
     s1
WHERE
     grd1   :   n ∈ VERSIO
                N
     grd2   :   n < 9
                9
     grd3   :   n ≥
                1
```

grd1～4：「現在の状態」は「1 版～n-1 版：公開-変更有、n 版：公開-なし、n+1 版：非公開-削除済」

```
    grd4   :  partition(state,1‥
              n−1×{koukai↦henkouari},{n↦(koukai↦nasi),n+1↦(hikoukai↦sakujozumi)})
    grd5   :   s1 = {n↦(koukai↦henkouari),n+1↦(koukai↦nasi
               )}
THEN
    act1   :   ev ≔ henkou_sokujiKouka
               i
    act2   :   statePre ≔ stat
               e
    act3   :   state ≔ state  [
               1
END

   shusei01   ≙          //    修正(期限切れ／な
                    し)
   STATUS
    ordinar
    y
ANY
    n
    s1
WHERE
    grd1   :   n ∈ VERSIO
               N
    grd2   :   partition(state,1‥
               n−1×{kigengire↦henkouari},{n↦(kigengire↦nasi)})
    grd3   :   s1 = {x↦(y2↦z)∣∃y1·x↦(y1↦z)∈state∧y1∈(KOUKAI_FLG\{hikoukai})∧
               y2=koukai}
THEN
    act1   :   ev ≔ shuse
               i
    act2   :   statePre ≔ stat
               e
    act3   :   state ≔ state  [
               1
END
```

act1：「操作イベント」に「変更_即時公開」を代入
act2：「一つ前の状態」に「現在の状態」を代入
act3：「現在の状態」の「n 版」に「公開-変更有」を代入し、「n+1 版」に「公開-なし」を代入

grd1～2：「現在の状態」は「1 版～n-1 版：期限切れ-変更有、n 版：期限切れ-なし」

act1：「操作イベント」に「修正」を代入
act2：「一つ前の状態」に「現在の状態」を代入
act3：「現在の状態」で「非公開」でない「版」に「公開」を代入

```
    sakujo01    ≙          //    削除(未公開／な
                       し)
    STATUS
     ordinar
     y
ANY
     n
     s1
WHERE
     grd1    :    n =
                  1
     grd2    :    state = {n↦(mikoukai↦nasi)
                  }
     grd3    :    s1 = {n↦(hikoukai↦sakujozumi)
                  }
THEN
     act1    :    ev ≔ sakuj
                  o
     act2    :    statePre ≔ stat
                  e
     act3    :    state ≔ state  [
                  1
END


    sakujo02    ≙          //    削除(未公開／変更
                       中)
    STATU
     ordinar
     y
ANY
     n
     s1
WHERE
     grd1    :    n ∈ VERSIO
                  N
```

grd1～2：「現在の状態」は「1 版：未公開-なし」

act1：「操作イベント」に「削除」を代入
act2：「一つ前の状態」に「現在の状態」を代入
act3：「現在の状態」の「n 版」に「非公開-削除済」を代入

grd1～4：「現在の状態」は「1 版～n-2 版：公開-変更有、n-1 版：公開-変更中、n 版：未公開-変更中」

```
    grd2   :   n < 9
               9
    grd3   :   n ≥
               2
    grd4   :   partition(state,1‥
               n−2×{koukai↦henkouari},{n−1↦(koukai↦henkouchu),n↦(mikoukai↦henkouchu)}
               )
    grd5   : s1 = (dom({x↦(y↦z)|x↦(y↦z)∈state∧y=koukai∧
               z=henkouchu})×{koukai↦nasi})∪{n↦(hikoukai↦sakujozumi)}
THEN
    act1   :   ev ≔ sakuj
               o
    act2   :   statePre ≔ stat
               e
    act3   :   state ≔ state  [
               1
END
```

act1：「操作イベント」に「削除」を代入
act2：「一つ前の状態」に「現在の状態」を代入
act3：「現在の状態」で「公開-変更中」の「版」に「公開-なし」を代入し、「n 版」に「非公開-削除済」を代入

```
   sakujo03   ≙          //   削除(公開／な
                  し)
   STATUS
    ordinar
    y
ANY
    n
    s1
WHERE
    grd1   :   n ∈ VERSIO
               N
               n ≥
    grd2   :
               1
    grd3   :   partition(state,1‥
               n−1×{koukai↦henkouari},{n↦(koukai↦nasi)})
    grd4   :   s1 = dom({x↦(y↦z)|x↦(y↦z)∈state∧y∈
               (KOUKAI_FLG\{hikoukai})})×{koukai↦sakujozumi}
THEN
```

grd1〜3：「現在の状態」は「1 版〜n-1 版：公開-変更有、n 版：公開-なし」

```
    act1   :   ev ≔ sakuj
               o
    act2   :   statePre ≔ stat
               e
               state ≔ state  [
    act3   :
               1
END

    kyoseiSakujo01   ≙          //   強制削
                                除
    STATUS
     ordinar
     y
ANY
    n
    s1
WHERE
    grd1   :   n ∈ VERSIO
               N
    grd2   :   state={1↦(mikoukai↦nasi)
               }
    grd3   :   s1 = dom({x↦(y↦z)|x↦(y↦z)∈state∧y∈
               (KOUKAI_FLG\{hikoukai})})×{hikoukai↦sakujozumi}
THEN
    act1   :   ev ≔ kyoseiSakuj
               o
    act2   :   statePre ≔ stat
               e
    act3   :   state ≔ state  [
               1
END

    kyoseiSakujo02   ≙          //   強制削除
    STATUS
     ordinar
     y
```

act1：「操作イベント」に「削除」を代入
act2：「一つ前の状態」に「現在の状態」を代入
act3：「現在の状態」で「非公開」でない「版」に「公開-削除済」を代入

grd1～2：「現在の状態」は「1版：未公開-なし」

act1：「操作イベント」に「強制削除」を代入
act2：「一つ前の状態」に「現在の状態」を代入
act3：「現在の状態」で「非公開」でない「版」に「非公開-削除済」を代入

```
ANY
    n
    s1
WHERE
    grd1   :   n ∈ VERSIO
               N
    grd2   :   n ≥
               2
    grd3   :   partition(state,1‥
               n−2×{koukai↦henkouari},{n−1↦(koukai↦henkouchu),n↦(mikoukai↦henkouchu)}
               )
    grd4   :   s1 = dom({x↦(y↦z)|x↦(y↦z)∈state∧y∈
               (KOUKAI_FLG\{hikoukai})})×{hikoukai↦sakujozumi}
THEN
    act1   :   ev ≔ kyoseiSakuj
               o
    act2   :   statePre ≔ stat
               e
    act3   :   state ≔ state□s
               1
END

   kyoseiSakujo03   ≙          //   強制削
                               除
   STATUS
    ordinar
    y
ANY
    n
    s1
WHERE
    grd1   :   n ∈ VERSIO
               N
    grd2   :   n ≥
               1
    grd3   :   partition(state,1‥
```

grd1～3：「現在の状態」は「1 版～n-2 版：公開-変更有、n-1 版：公開-変更中、n 版：未公開-変更中」

act1：「操作イベント」に「強制削除」を代入
act2：「一つ前の状態」に「現在の状態」を代入
act3：「現在の状態」で「非公開」でない「版」に「非公開-削除済」を代入

grd1～3：「現在の状態」は「1 版～n-1 版：公開-変更有、n 版：公開-なし」

```
                n−1×{koukai↦henkouari},{n↦(koukai↦nasi)})
    grd4    :   s1 = dom({x↦(y↦z)|x↦(y↦z)∈state∧y∈
                (KOUKAI_FLG\{hikoukai})})×{hikoukai↦sakujozumi}
THEN
    act1    :   ev ≔ kyoseiSakuj
                o
    act2    :   statePre ≔ stat
                e
    act3    :   state ≔ state  [
                1
END

    kyoseiSakujo04   ≙          //   強制削
                                除
    STATUS
     ordinar
     y
ANY
    n
    s1
WHERE
    grd1    :   n ∈ VERSIO
                N
    grd2    :   n ≥
                1
    grd3    :   state=1‥
                n×{koukai↦sakujozumi}
    grd4    :   s1 = dom({x↦(y↦z)|x↦(y↦z)∈state∧y∈
                (KOUKAI_FLG\{hikoukai})})×{hikoukai↦sakujozumi}
THEN
    act1    :   ev ≔ kyoseiSakuj
                o
    act2    :   statePre ≔ stat
                e
    act3    :   state ≔ state  [
                1
```

act1：「操作イベント」に「強制削除」を代入
act2：「一つ前の状態」に「現在の状態」を代入
act3：「現在の状態」で「非公開」でない「版」に「非公開-削除済」を代入

grd1〜3：「現在の状態」は「1 版〜n 版：公開-削除済」

act1：「操作イベント」に「強制削除」を代入
act2：「一つ前の状態」に「現在の状態」を代入
act3：「現在の状態」で「非公開」でない「版」に「非公開-削除済」を代入

```
END

    kyoseiSakujo05   ≙          //   強制削
                                除
    STATUS
     ordinar
     y
ANY
     n
     s1
WHERE
     grd1   :   n ∈ VERSIO
                N
     grd2   :   n ≥
                1
     grd3   :   partition(state,1‥
                n−1×{kigengire↦henkouari},{n↦(kigengire↦nasi)})
     grd4   :   s1 = dom({x↦(y↦z)∣x↦(y↦z)∈state∧y∈
                (KOUKAI_FLG\{hikoukai})})×{hikoukai↦sakujozumi}
THEN
     act1   :   ev ≔ kyoseiSakuj
                o
     act2   :   statePre ≔ stat
                e
     act3   :   state ≔ state  [
                1
END

END
```

grd1～3：「現在の状態」は「1 版～n-1 版：期限切れ-変更有、n 版：期限切れ-なし」

act1：「操作イベント」に「強制削除」を代入
act2：「一つ前の状態」に「現在の状態」を代入
act3：「現在の状態」で「非公開」でない「版」に「非公開-削除済」を代入

図 3-9　画面アクション明細から抽出したイベントを検証するための Event-B 形式記述

3.3.4　形式記述の検証

検証を行うと MACHINE における下記の「イベント」が証明出来ない。

henkou_teijitenKoukai01

henkou_sokujiKoukai01

henkou_sokujiKoukai02

その理由は以下の通りである。例えば henkou_sokujiKoukai01 は、書類の最新版の公開/削除状態を「未公開/なし」から「公開/なし」に変化させる「イベント」である。この「イベント」は CONTEXT（c1）で定義した状態遷移定義において下記の「操作」に対応する。

axm26 : {1↦(mikoukai↦nasi)}↦{1↦(koukai↦nasi)}∈henkou_sokujiKoukai

これは、書類に 1 版しかない場合にその公開/削除状態を「未公開/なし」から「公開/なし」に変化させる「操作」である。しかしながら、「画面アクション明細」の当該個所では変更元の書類に 1 版しか存在しないというガード条件が記述されていないため、書類に 1 版から n 版まで存在する中の n 版の公開/削除状態を「未公開/なし」から「公開/なし」に変化させるという「イベント」になっている。そのため、「イベントの実行前後における書類の公開/削除状態が CONTEXT（c1）で定義した操作のどれかと一致している」という不変条件を満たさないため証明できない。

他の２つのイベントについても同様の理由である。

3.4　実験結果

3.4.1　工数

本実験において、各作業にかけた工数を表 3-5　に示す。

表 3-5　各作業にかけた工数

作業名	工数
Event-B 要素の抽出	54 人 H
形式記述の作成	16 人 H
形式記述の検証	20 人 H
計	90 人 H

3.4.2 設計書と成果物

3.4.2.1 規模

本実験で用いた、本書の対象範囲に関する設計書、作成した中間成果物や形式記述の量を表 3-6 に記す。

表 3-6 作成した中間成果物や形式記述の量

入出区分	成果物	規模
入力	外部設計書	形式記述の作成対象とした設計書の規模 頁数 49、機能数 5、画面数 5、テーブル数 1
		参照したページも含む設計書の規模 頁数 381、機能数 29、画面数 27、テーブル数 11
中間	Event-B 要素一覧	行数 67
出力	形式記述	行数 425、リファインメントステップ数 0 証明課題数 90 （内、自動証明数 52）

3.4.2.2 作業効率

前節の入力設計書の規模と形式手法の適用にかかった工数との間には以下の関係がある。

- 作業効率(1)(人時/頁)=90(人時)/49(頁)=1.8(人時/頁)
- 作業効率(2)(人時/頁)=90(人時)/381(頁)=0.2(人時/頁)

3.4.3 形式手法による指摘事項

3.4.3.1 指摘事項種別

本実験において、検出した指摘事項の数を表 3-7 に示す。

表 3-7　検出した指摘事項の数

指摘事項種別*	個数
設計内容の衝突	0 個
設計内容の不足	1 個
設計内容の曖昧さ	0 個
誤字脱字など	0 個
合計	1 個

*指摘事項種別

- 設計内容の衝突：（データ構造や判断の条件など）同じ意味を持つべき事柄が記述箇所によって異なる意味をもつように書かれている
- 設計内容の不足：本来あるべき設計がなされていない （複数個所の記述に食い違いがあり、一方の記述の不足により、他方で想定外の状況になることが、想像できる）
- 設計内容の曖昧さ：設計書の書き方が曖昧であるため、人によって異なる解釈が生じる
- 誤字脱字など：誤字・脱字などの表記に関する指摘

3.4.3.2　作業と指摘事項の関係

前節の指摘事項が、形式手法を適用する際のどの作業で見つかったかを表 3-8　に示す。

表 3-8　作業と指摘事項の関係

作業	本実験における指摘事項の数			
	衝突	不足	曖昧さ	合計
要素の抽出	0 個	0 個	0 個	0 個
リファインメント計画の策定	0 個	0 個	0 個	0 個
形式記述の作成	0 個	0 個	0 個	0 個
形式記述の検証	0 個	1 個	0 個	1 個
形式記述の修正	0 個	0 個	0 個	0 個
合計	0 個	1 個	0 個	1 個

3.4.3.3　指摘事項抽出効率

前節の指摘事項の個数と形式手法の適用にかかった工数との間には以下の関係がある。

- 指摘効率（工数ベース）＝ 1[個数]／90[人 H] ＝ 0.01[個数／人 H]
- 指摘効率（指摘事項ベース）＝ 90[人 H]／1[個数] ＝ 90[人 H／個数]

3.4.3.4　抽出した指摘事項の例

「画面アクション明細」の図 3-10　に示す部分で「変更元の書類が“未公開-なし”の場合、初版なので、当該レコード以外の更新を行わない。」と記述されている。この記述は、変更元の書類の公開/削除状態が“未公開-なし”であれば初版であるということを前提にしている。しかしながら、仮に変更元の書類が“未公開-なし”で初版ではない場合、そのような書類の状態は「状態遷移図」には存在し得ないにもかかわらず、チェックを受けずに新たな別の不正な状態として存在し続けてしまう。例えば、「状態遷移図」には存在し得ない状態「1 版：公開-変更有、2 版：公開-変更中、3 版：未公開-なし」であれば、新たな別の存在し得ない状態「1 版：公開-変更有、2 版：公開-変更中、3 版：公開-なし」に変化することを許してしまう。上記の記述は、例えば、「変更元の書類が“未公開-なし”で初版の場合、当該レコード以外の更新を行わない。“未公開-なし”で初版ではない場合にはエラーとする。」という記述に変更すれば不正な状態をエラーとして処理することができる。

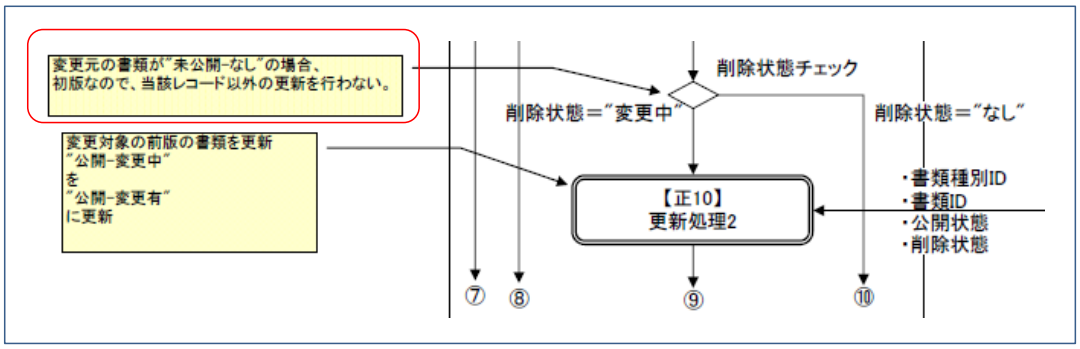


図 3-10　指摘事項の例

3.5　考察

本実験対象システムの場合のように、1 版だけの「状態遷移図」と 1 版から 2 版を生成したときの「状態遷移図」についてだけ設計書に記述されているが 3 版以降の任意の版の状態が明示的に示されていない場合、「画面アクション明細」において、更新対象とする版の二つ以上前の版に対する扱いについて明示的に設計されない可能性がある。このような

場合、Event-B で形式記述を作成していくと 3 版以降の明示されていない版についてもそれが「イベント」の駆動に関係する限り明らかにされるので、「画面アクション明細」において、更新対象とする版の二つ以上前の版に対する扱いについての過不足が明らかになる。例えば、本実験の検証結果からは、「画面アクション明細」では必ずしも全ての版の事前状態を検証しているわけではないことが明らかになった。

このことは、例えば 3.4.3.4 節で示した指摘事項の例について考えると、変更元の書類に 1 版しか存在しないというガード条件が記述されていないため、「状態遷移図」には存在し得ない状態「1 版：公開/変更有、2 版：公開/変更中、3 版：未公開/なし」が、新たな別の存在し得ない状態「1 版：公開/変更有、2 版：公開/変更中、3 版：公開/なし」に変化することを許してしまうことになる。このような不正な事前状態は、仮にシステムの他の全ての部分が書類の状態を正しく保つように設計されていたとしても、データベースのメンテナンス時のミスや仕様変更時の影響範囲の確認ミスなどで発生してしまう可能性がある。このような不正な状態が新たな別の不正な状態に改変されたまま残ってしまうと、システム上の予期しない結果を招く可能性がある。[11]このような予期しない結果は、「画面アクション明細」の当該個所に不足していたガード条件を補うことによって回避することができる。例えば、変更元の書類に 1 版しか存在しないというガード条件を付加することにより不正な状態をエラーとして処理することができる。

このように、予期しない結果を未然に発見するということも形式手法の効果の一つなのではないかと考えられる。

3.6 まとめ

外部設計書で定義された「状態遷移図」に対して、「画面アクション明細」に記載された処理が整合性を保っているかどうかを確認するという目的で実験を行った。その際、設計書には明示的に記載されていない 3 版～n 版までの状態も含めた全ての状態について形式化を行うことにより、「画面アクション明細」における潜在的なガード条件の不足を明確にすることができた。

[11]予期しない結果を許容し得るかどうかはシステムの特性によって異なる。そのため、システムの信頼性要件などを踏まえた上で改善策の導入の可否を検討する必要がある。

4. SPIN を使用した実験(S チーム)

4.1 目的

本実験において、何を確認するために、どのような工程成果物に対して、どの形式手法を用いたかを表 4-1 に記す。

表 4-1 目的

採用した手法	SPIN	
(特徴)	SPIN はシステムの動作を状態遷移として表した形式記述を網羅的に調べる、モデル検査と呼ばれる手法を実行する代表的なツールである。「システムがいつでも特定の性質を満たす」あるいは「システムが動作する中でいつかは特定の性質が成立するようになる」などの時系列的なことがらを調べることができる。 SPIN でモデル検査を実行するためには 2 種類の(形式)記述が必要である。一つは PROMELA と呼ばれる言語でシステムの動作を記述したものである。これだけでもシステムの動作確認(記述した振舞いに従って計算機上でシステムの状態を変化させる)が可能である。 もう一つは LTL 式と呼ばれる記述である。「システムがいつでも特定の性質を満たす」あるいは「システムが動作する中でいつかは特定の性質が成立するようになる」などの時系列的なことがらを表現する。これを上記の PROMELA 記述と一緒に SPIN に入力することにより、時系列的なことがらについて網羅的な調査が可能となる。	
確認すること	複数の外部設計書にある「書類」の状態遷移が矛盾なく記述されていることを確認する。	
(確認方法)	本実験で使用した外部設計書は、「書類」の状態遷移について 2 種類の外部設計書で説明している。一つは『「書類」の状態遷移を状態遷移図で記述した外部設計書』(以後“状態遷移図”)であり、もう一つは『画面イベントによるデータベースの更新処理について書かれた外部設計書』(以後“画面アクション明細”)である。 これら 2 種類の外部設計書は同一の状態遷移について説明しているため、同一の初期状態を持ち、同一の状態遷移イベントに対しては同一の状態遷移を行うはずである。そこで実験チームは 2 種類の外部設計書から独立に「書類」の状態および状態遷移イベントを SPIN の利用に適した形で選出し、それらが同一の初期状態から同一のイベントに対して同一の状態遷移を行うことを確認した。	
確認の対象	成果物	形式記述
	画面遷移図(『「書類」の状態遷移を状態遷移図で書いた外部設計書」』)	PROMELA 記述(mtype, proctype, inline)

	画面アクション明細(『画面イベントによるデータベースの更新処理について書かれた外部設計書』)	PROMELA 記述(proctype, inline)
(備考)	(特になし)	

4.2 体制

本実験を行った作業者の役割やスキルなどを表 4-2 に記す。

表 4-2 体制

作業者	役割*	スキル (経験)			
		業務 AP 開発	類似 AP 開発	形式手法	採用手法
A	形式記述者	0 年	0 年	6 年	6 年
B	形式記述者	2 年	0 年	5 年	0 年

* 役割とその担当する作業は以下のとおり。

- 形式記述者 :『形式記述』の作成と修正を行う
- 形式検証者 :『形式記述』に関する検証を行う

4.3 作業

4.3.1 作業の概要

本実験では、図 4-1 に示す一連の作業を行った。

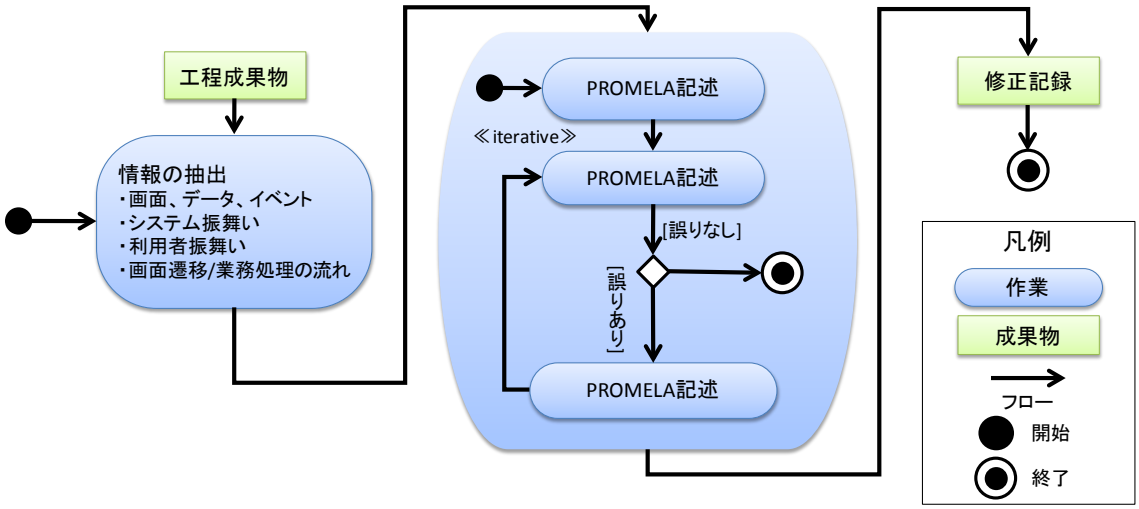


図 4-1

各作業の内容については『形式手法適用手順（SPIN 編）【リリース版】』 [3]の適用法２を参照のこと。なお、本実験では、1 件の実験対象範囲を 1 人の作業者で行ったため、PROMELA 記述作成での情報共有の必要がないことや作業効率の観点から中間記述の作成は省略した。また入力となる外部設計書が適用法２と異なるため、表 4-3 に示すような読み替えを実施した。ここで、状態および状態遷移イベントは図 4-2 に示すように、表 4-3 の状態遷移図に含まれる要素を指す。

表 4-3

番号	適用法 2	本実験
1	画面遷移図	状態遷移図
2	画面	状態
3	画面更新イベント	状態遷移イベント

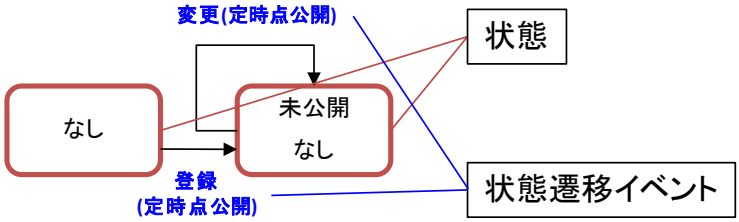


図 4-2

4.3.2 実験対象範囲

SPIN チームは 2 名の作業者が、「提供書類」、「コーポレートアクション情報」における登録、削除、変更、強制削除の処理について検証を行った。

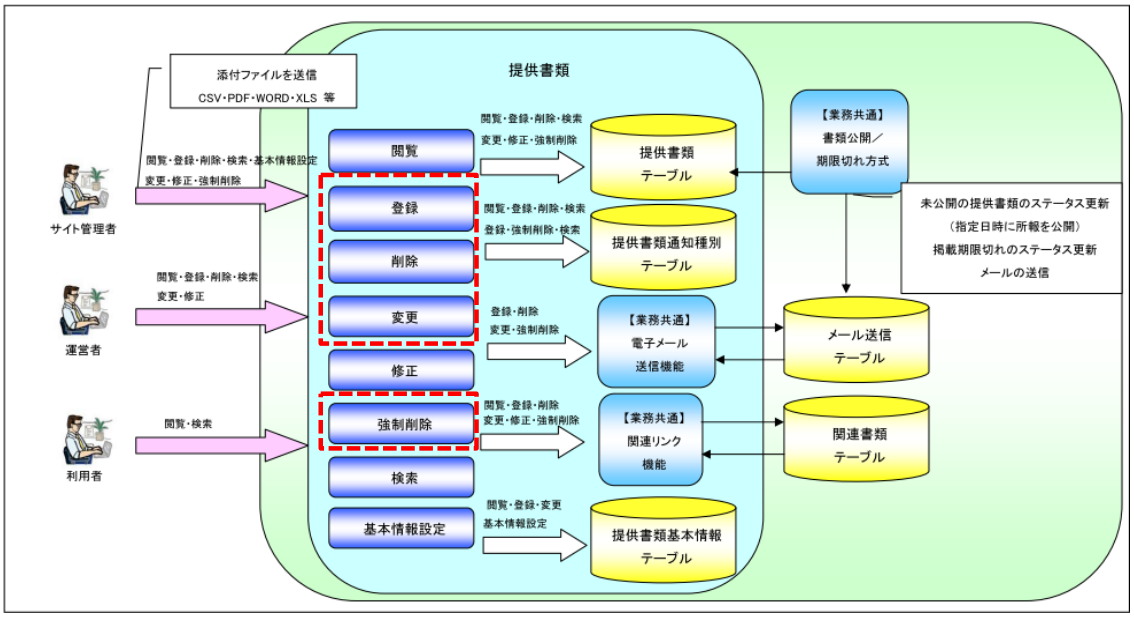


図 4-3

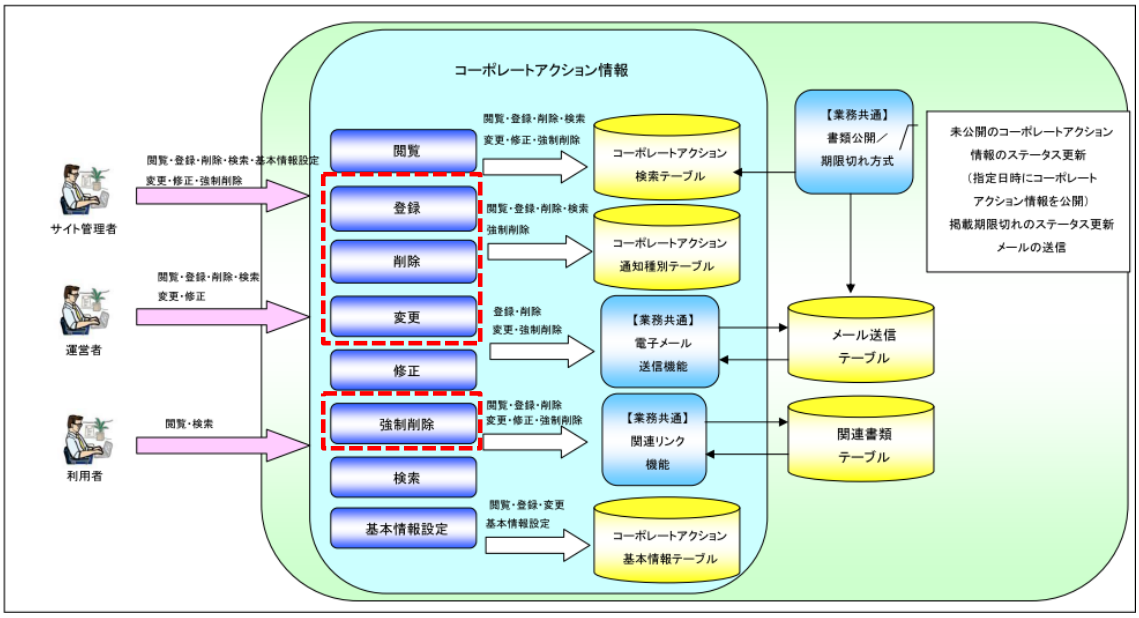


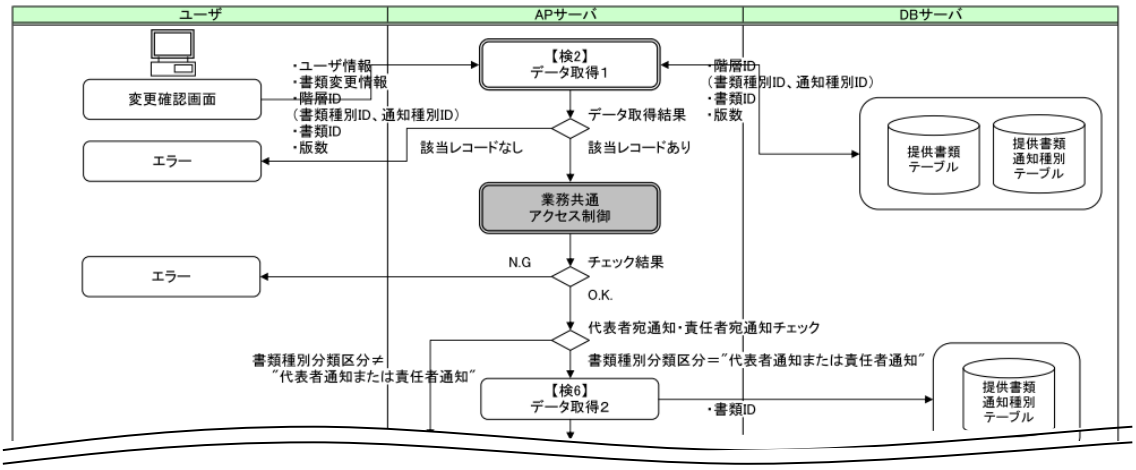
図 4-4

4.3.3 検証目的

登録、削除、変更、強制削除の処理は、「画面アクション明細」（図 4-5）等の外部設計書で定義されており、処理の過程でテーブル上の書類の状態の更新などを実行する。また、処理の結果、満たすべき書類の状態については別途、図 4-6 のように「状態遷移図」で定義されている（詳細は『情報系の実稼働システムを対象とした形式手法適用実験報告書』3.3.2 節を参照）。

本実験では、全ての「画面アクション明細」において、「状態遷移図」で定義された遷移

に違反することなく書類操作が行われているか検証することを目的とする。



パターン	No	アクション詳細ID	枝番	処理
検証	1			
	2	(DTRBCPD0108)	2-1	入力値(階層ID、書類ID、版数)を条件に、提供書類テーブルより書類情報を取得
	3	-	3-1	検証2で取得したレコードが存在しない場合
			3-2	エラーメッセージ(MSGTRBCBC40940:"指定された[書類]が見つかりません。")をセット
			3-3	エラー画面へ遷移し、終了
	4	共通(FTRBCBC1901)	4-1	検証2で取得した書類の公開状態/削除状態から、変更可否を判定(業務共通)
	5	-	5-1	検証3の結果、N.Gの場合
			5-2	エラーメッセージ(MSGTRBCBC40951:"アクセス権がない為、[変更]できません。")をセット
			5-3	エラー画面に遷移し、終了
	6		6-1	検証2の結果、代表者宛通知または責任者宛通知の書類の場合
		(DTRBCPD0109)	6-2	入力値(書類ID)を条件に、提供書類通知種別テーブルより通知種別情報を取得
	7	-	7-1	検証6の結果
			7-2	該当レコードなしの場合、
			7-3	エラーメッセージ(MSGTRBCBC40940:指定された[書類]が見つかりません。)をセット
			7-4	エラー画面へ遷移し、終了
	8	共通(ITRUMQF0902)	8-1	検証6で取得したすべての通知種別IDに対し、登録権があるか確認

図 4-5

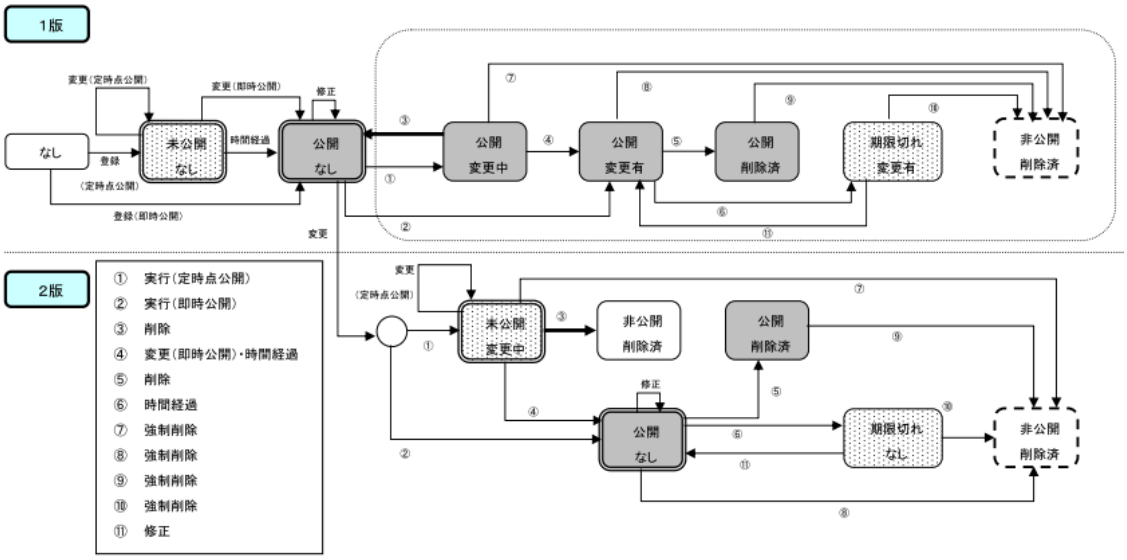


図 4-6

4.3.4　外部設計書と形式記述（PROMELA 記述）の関係

✧ **課題**

作業「情報の抽出」では、「画面遷移図」と「システム化業務説明」を入力としている。本実験で用いた外部設計書では、「画面遷移図」と「システム化業務説明」にあたる設計が存在しない。

✧ **対策**

「画面遷移図」については書類の「状態遷移図」が「画面遷移図」と同等の設計情報を有しているため、代用することで課題を解決した。「システム化業務説明」については「画面アクション明細」から相当する情報を抽出し、対応関係を整理したうえで形式記述（PROMELA 記述）を作成した。

外部設計書と形式記述（PROMELA 記述）の関係を図 4-7 に示す。『形式手法イディオム集（SPIN 編）【リリース版】』 [4] のイディオム S-S-2「全体制御によるシステム間の動作の表現」に従い、外部設計書の「状態遷移図」から、それぞれ「状態とデータと状態遷移イベント」、「利用者振舞いの PROMELA マクロ」、「状態遷移図の振舞いを表す PROMELA マクロ」の PROMELA 記述を作成し、「画面アクション明細」から、「画面アクション明細の振舞いを表す PROMELA マクロ」の PROMELA 記述を作成した。また検証に必要な「全体制御用 PROMELA プロセス」を同じくイディオム S-S-2 に従い追記した。

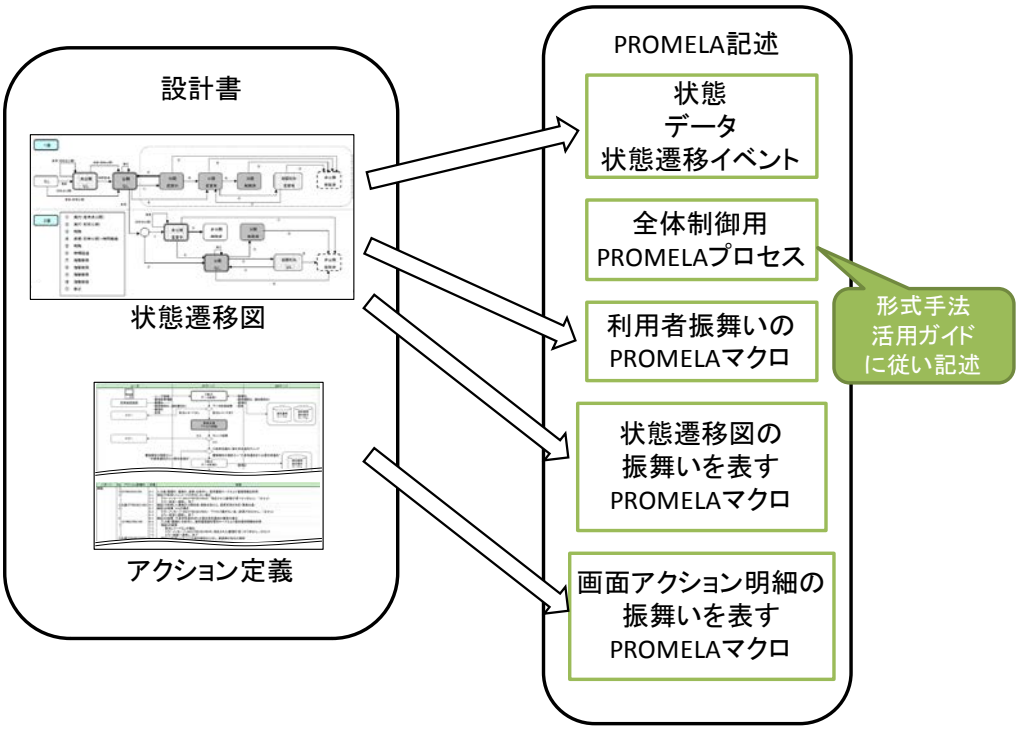


図 4-7

✧ **結果**

外部設計書から形式記述（PROMELA 記述）に必要な情報を抽出し、対応関係を整理することで、形式活用ガイドのイディオム S-S-2 を用いて PROMELA 記述の作成を行うことができた。

4.3.4.2　状態、データ、状態遷移イベントの抽出

「状態遷移図」から図 4-8 のように状態とデータおよび状態遷移イベントを抽出した。状態と状態遷移イベントは状態遷移図上から抽出し、データについては図中下部に「注)未公開/変更中の…」といった補足説明文中に遷移時のアクション内容が記載されていたため、その説明文から抽出した。

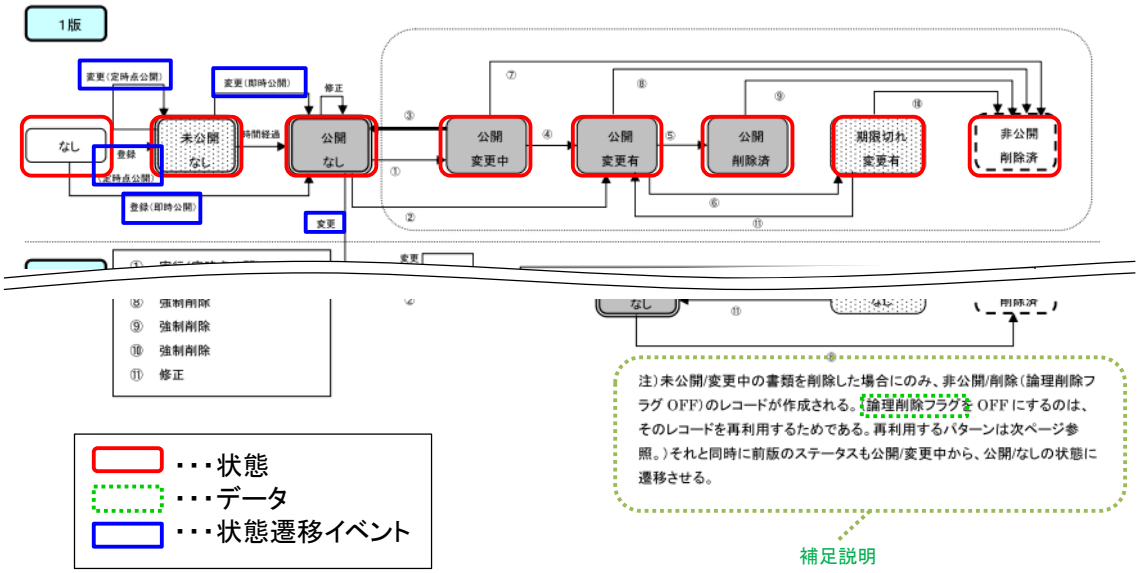


図 4-8

4.3.4.3　利用者振舞いの PROMELA マクロ

「状態遷移図」から各状態で選択可能な処理（状態遷移イベント）を抽出し、『形式手法適用手順（SPIN 編）【リリース版】』 [3]の適用法２に従い、利用者振舞いの PROMELA マクロを記述した。

「状態遷移図」と PROMELA 記述の対応を図 4-9 に示す。例えば、状態が「なし」の場合、利用者振舞いの PROMELA マクロでは状態遷移イベントとして登録（即時公開）または登録（定時点公開）が選択される。

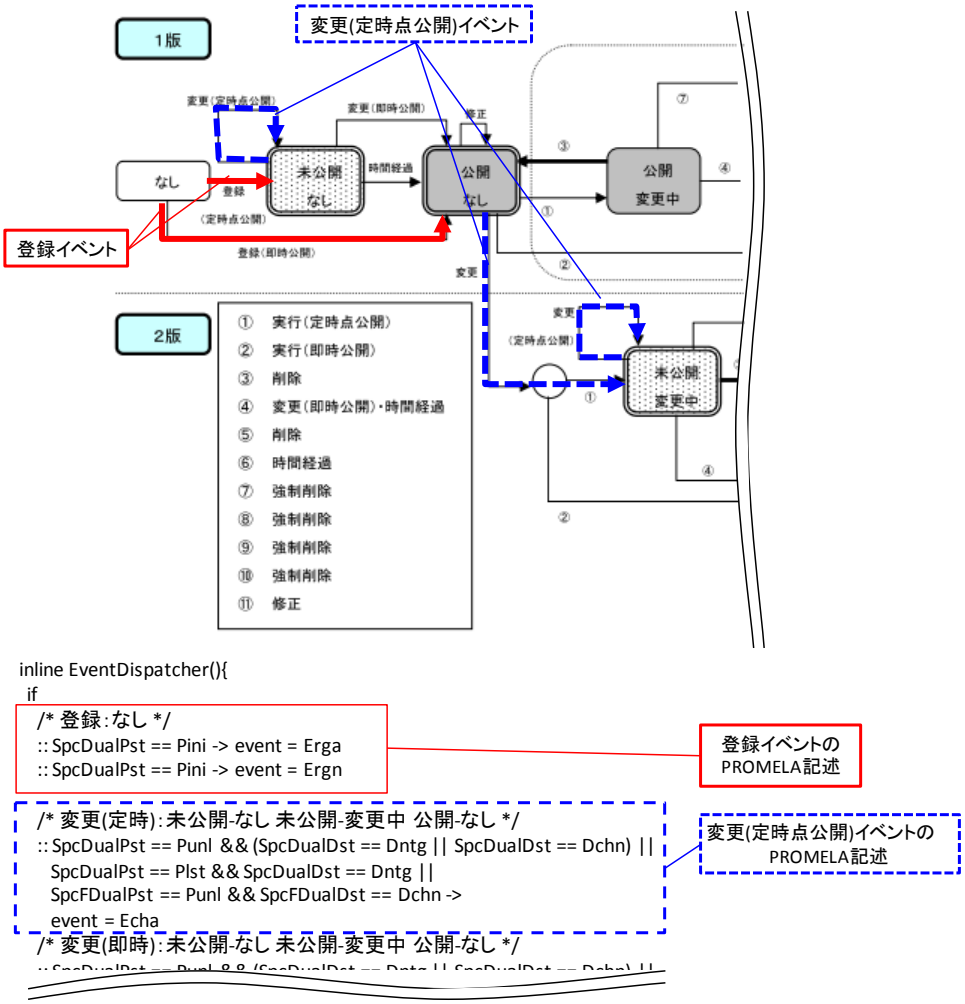


図 4-9

4.3.4.4 状態遷移図の振舞いを表す PROMELA マクロ

「状態遷移図」からの PROMELA 記述作成については『形式手法適用手順（SPIN 編）【リリース版】』 [3]の適用法２の「画面遷移図」から PROMELA 記述を作成する手順を用いて作成することができる。「状態遷移図」を元に作成した PROMELA 記述（一部）は図 4-10 のようになる。

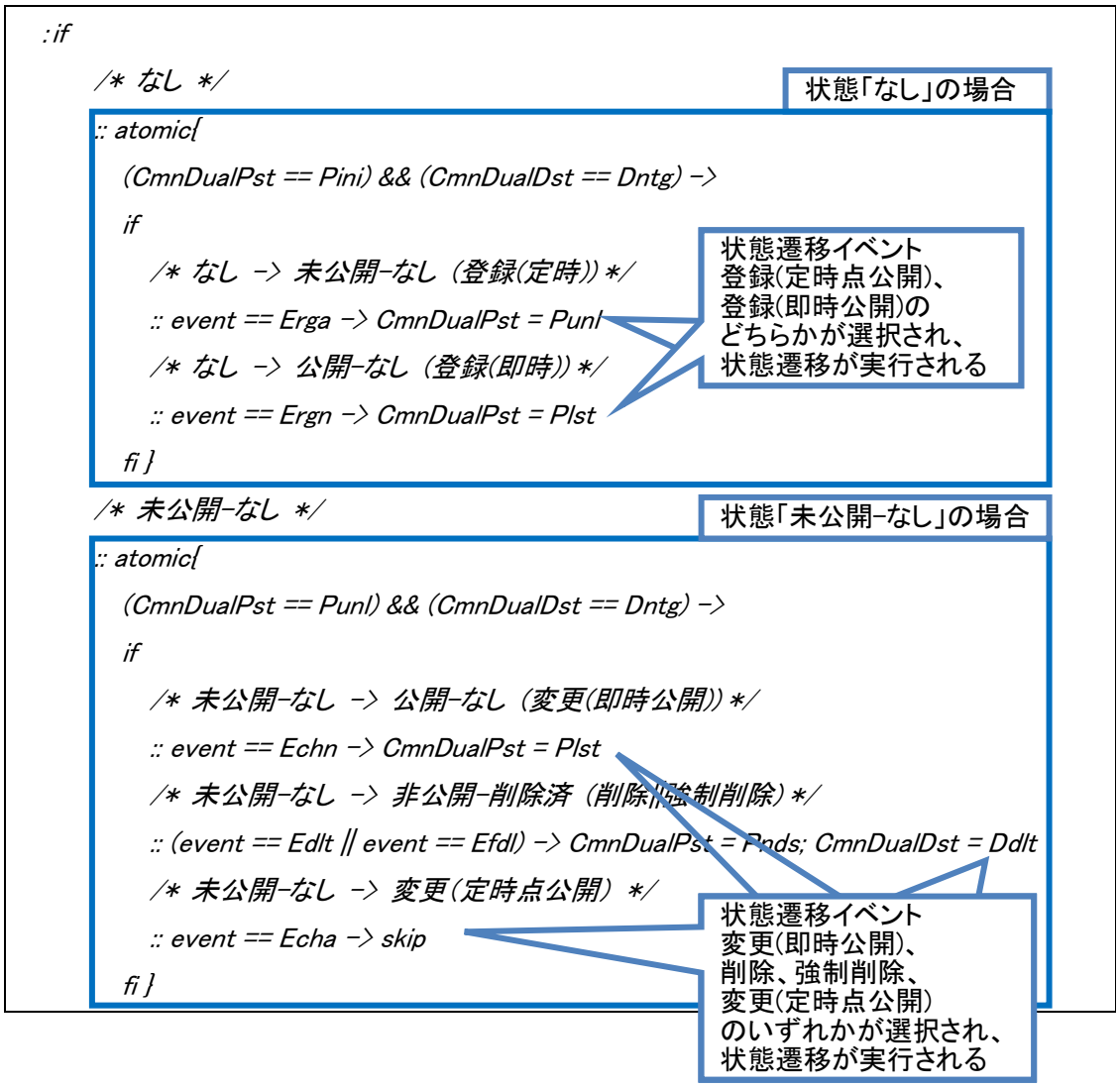


図 4-10

4.3.4.5　画面アクション明細の振舞いを表す PROMELA マクロ

各処理の PROMELA 記述を「画面アクション明細」を元に作成する。「画面アクション明細」のうち、書類の状態遷移 に関連するガード条件やアクション、処理の振舞いをピックアップし、PROMELA 記述に反映した。

提供書類の変更処理を例に示す。書類の公開状態が公開のときの処理の流れは図 4-11 のようになっている。書類の状態遷移に関連するガード条件、アクション、処理の振舞いは図中、赤色の太い矢印および枠のようになる。

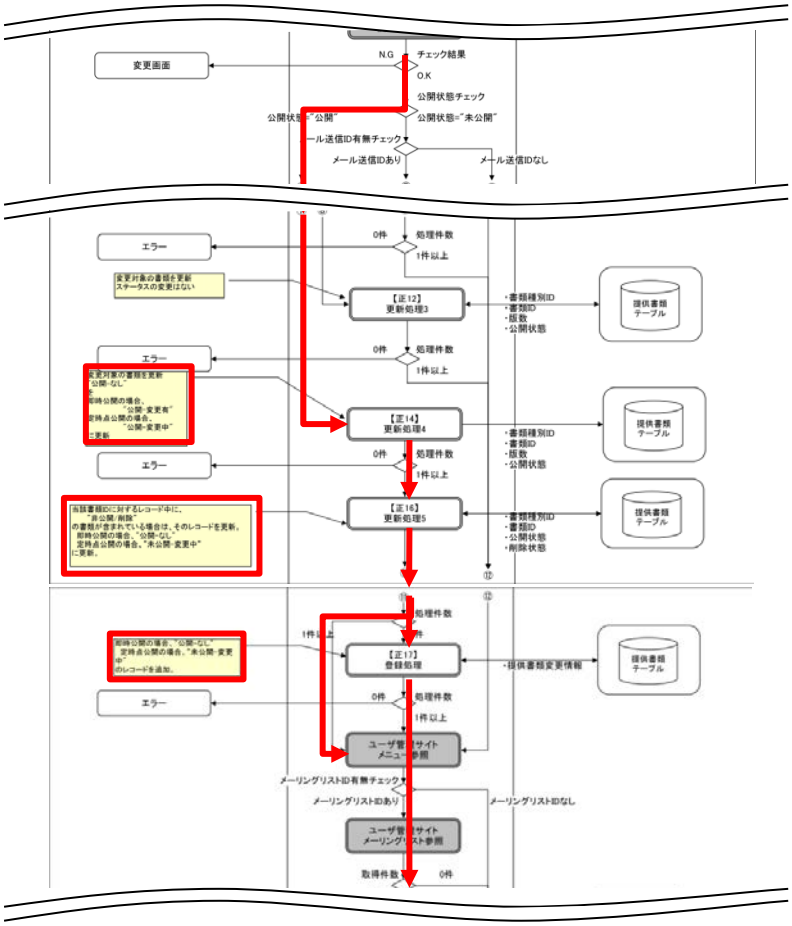


図 4-11

図 4-11 を例に PROMELA を記述すると図 4-12 のようになる。「【正 XX】の処理」という記述は画面アクション明細中の特定のアクションを指す記号を表す。

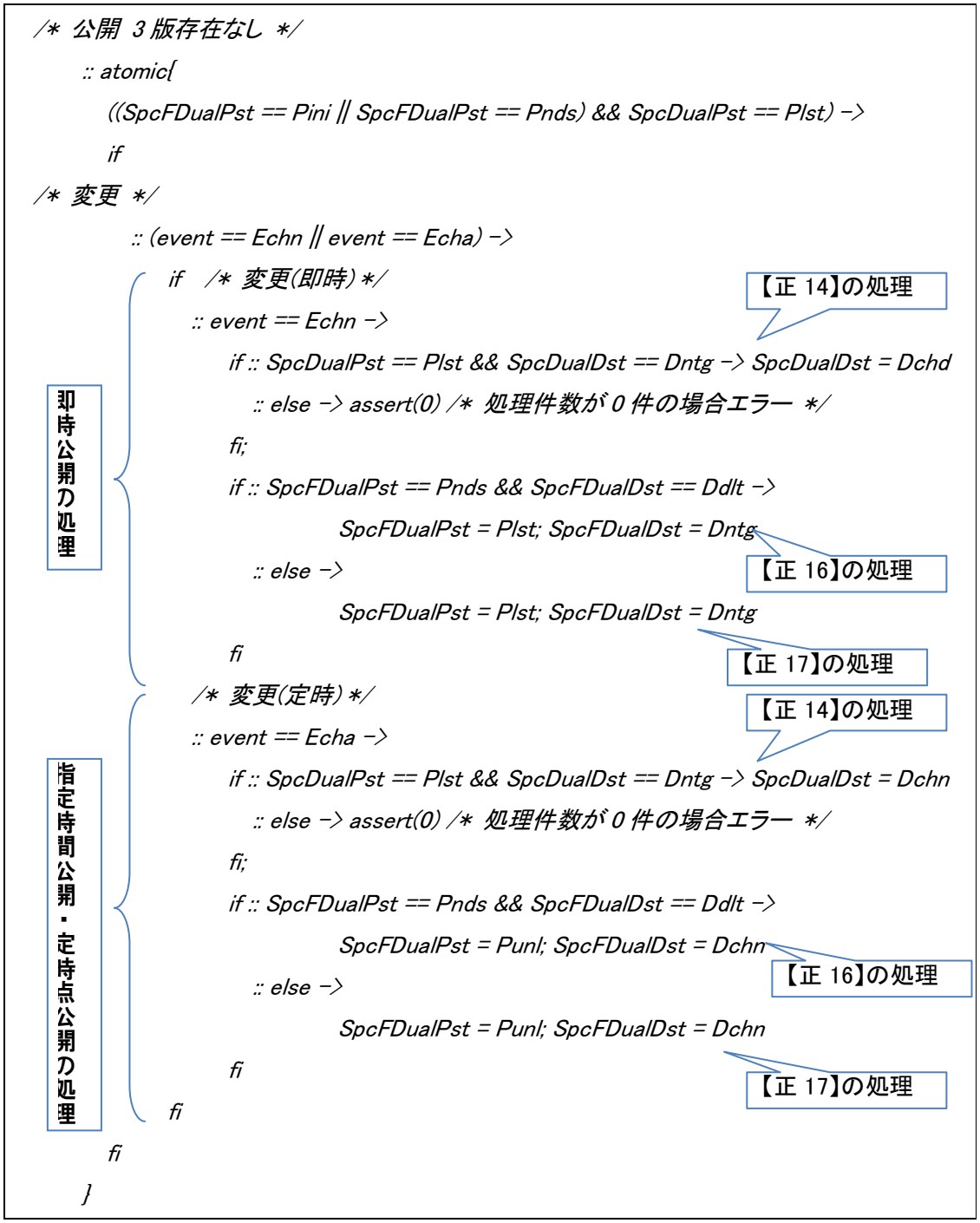

/* 公開 3 版存在なし */
:: atomic{
((SpcFDualPst == Pini || SpcFDualPst == Pnds) && SpcDualPst == Plst) ->
if
/* 変更 */
:: (event == Echn || event == Echa) ->
if /* 変更(即時) */
:: event == Echn ->
if :: SpcDualPst == Plst && SpcDualDst == Dntg -> SpcDualDst = Dchd
:: else -> assert(0) /* 処理件数が 0 件の場合エラー */
fi;
if :: SpcFDualPst == Pnds && SpcFDualDst == Ddlt ->
SpcFDualPst = Plst; SpcFDualDst = Dntg
:: else ->
SpcFDualPst = Plst; SpcFDualDst = Dntg
fi
/* 変更(定時) */
:: event == Echa ->
if :: SpcDualPst == Plst && SpcDualDst == Dntg -> SpcDualDst = Dchn
:: else -> assert(0) /* 処理件数が 0 件の場合エラー */
fi;
if :: SpcFDualPst == Pnds && SpcFDualDst == Ddlt ->
SpcFDualPst = Punl; SpcFDualDst = Dchn
:: else ->
SpcFDualPst = Punl; SpcFDualDst = Dchn
fi
fi
fi
}
即時公開の処理
指定時間公開・定時点公開の処理
【正 14】の処理
【正 16】の処理
【正 17】の処理
【正 14】の処理
【正 16】の処理
【正 17】の処理

図 4-12

4.4 実験結果

4.4.1 工数

本実験において、各作業にかけた工数(作業者 2 名の合計)を表 4-4 に示す。

表 4-4

作業名	工数(作業者 2 名の合計)
情報の抽出と中間記述の作成	16.5 人 H
PROMELA 記述(LTL 式)の作成	19.5 人 H
PROMELA 記述(LTL 式)の検証	8.5 人 H
(合計)	44.5 人 H

4.4.2 外部設計書と成果物

4.4.2.1 規模

本実験で用いた、本書の対象範囲に関する外部設計書、作成した中間成果物や形式記述の量を表 4-5 に記す。

表 4-5

入出区分	成果物	規模
入力	画面遷移図	(形式記述と直接関係する部分のみ)A4 用紙 3 ページ(作業者 2 名による重複を合わせると 5 ページ) (上記および参照部分)A4 用紙 25 ページ(作業者 2 名による重複を合わせると 50 ページ)
	画面アクション明細	(形式記述と直接関係する部分のみ)A4 用紙 106 ページ(作業者 2 名による重複を合わせると 146 ページ) (上記および参照部分)A4 用紙 344 ページ(作業者 2 名による重複を合わせると 562 ページ)
出力	形式記述	行数 724

4.4.2.2　作業効率

前節の入力外部設計書の規模と形式手法の適用にかかった工数との間には以下の関係がある。

- 作業効率(工数ベース、形式記述と直接関係する部分のみ)＝ 44.5[人 H]／151(5 + 146)[ページ] ＝ 0.29[人 H／ページ数]
- 作業効率(工数ベース、形式記述と直接関係する部分および参照部分)＝ 44.5[人 H]／612(50 + 562)[ページ] ＝ 0.07[人 H／ページ数]

4.4.3　指摘事項

4.4.3.1　指摘事項種別

本実験において、検出した指摘事項の数を表 4-6 に示す。

表 4-6

指摘事項種別*	個数
設計内容の衝突	2 個
設計内容の不足	7 個
設計内容の曖昧さ	4 個
誤字脱字など	1 個

＊ 指摘事項種別

- 設計内容の衝突:(データ構造や判断の条件など)同じ意味を持つべき事柄が記述箇所によって異なる意味をもつように書かれている
- 設計内容の不足:本来あるべき設計がなされていない（複数個所の記述に食い違いがあり、一方の記述の不足により、他方で想定外の状況になることが、想像できる）
- 設計内容の曖昧さ:外部設計書の書き方が曖昧であるため、人によって異なる解釈が生じる
- 誤字脱字など：誤字・脱字などの表記に関する指摘

4.4.3.2　作業と指摘事項の関係

前節の指摘事項が、形式手法を適用する際のどの作業で見つかったかを表 4-7 に示す。

表 4-7

作業	指摘事項の個数				
	衝突	不足	曖昧さ	誤字脱字など	合計
情報の抽出	2 個	3 個	3 個	0 個	8 個
PROMELA 記述(LTL 式)の作成	0 個	1 個	0 個	0 個	1 個
PROMELA 記述(LTL 式)の検証	0 個	3 個	1 個	1 個	5 個

4.4.3.3 指摘事項の抽出効率

前節の指摘事項の個数と形式手法の適用にかかった工数との間には以下の関係がある。

- 抽出効率(工数ベース)= 14[個数]/44.5[人 H] = 0.31[個数/人 H]
- 抽出効率(指摘事項ベース)= 44.5[人 H]/14[個数] = 3.17[人 H/個数]

4.4.3.4 抽出した指摘事項の例

- 設計内容の曖昧さ
  ある「書類」がある状態をとるとき二つの状態遷移があり、画面遷移図ではこれを状態遷移図で二つの矢印として表している。この矢印が一部交差しているため両者の判別がつかない。

- 設計内容の衝突
  ある「書類」について第 3 版まで存在する場合を考える。このとき、第 1 版の(公開状態/削除状態)が(公開/変更有)、第 2 版が(公開/変更中)、第 3 版が(未公開/変更中)のとき、「削除」の状態遷移イベントについて遷移後の状態が状態遷移図と画面アクション明細で異なる。具体的には、状態遷移図では第 1 版が(公開/削除済)となるが、画面アクション明細では(公開/変更有)である。

- 設計内容の不足
  画面アクション明細において「詳細は【正 15】【正 17】【正 18】を参照」となっているが、その参照先がない。

4.5 考察

状態爆発について

SPIN を用いた検証でよく問題になる事柄として、検証対象の性質により SPIN が検証すべき状態が SPIN の限界を超えるいわゆる「状態爆発」問題がある。

本実験ではこの「状態爆発」問題は発生しなかった。その理由として以下2点が考えられる。

1. 抽象化について

   本実験では検証に直接関係する状態および状態遷移イベントのみを形式記述対象とした（抽象化した）ため、状態数を必要最低限に抑えることができた。

2. プロセス数について

   本実験で作成したPROMELA記述の方針としたイディオムS-S-2「全体制御によるシステム間の動作の表現」は、同時に処理が進行するプロセスを一つに絞っている。このため複数プロセスの並行動作による状態数の増加を抑えることができた。

4.6 まとめ

本実験では、状態遷移図と画面アクション明細から「書類」の状態および状態遷移イベントを SPIN の利用に適した形で選出し、それらが同一の初期状態から同一のイベントに対して同一の状態遷移を行うことを確認した。結果をまとめると以下のとおりとなる。

- 4.4.3.1 節の結果から、SPIN による外部設計書の検証が指摘事項抽出が可能であることを確認できた。また 4.4.3.2 節の結果から、抽出した指摘事項の多くが形式記述を作成する前の段階（「情報の抽出」）に集中していることがわかった。
- 4.4.3.4 節から、SPIN による外部設計書の検証で抽出できる指摘事項は、従来型のレビュー（インスペクションやウォークスルー等）で抽出できる指摘事項に類似していることがわかった。

5. VDM++を使用した実験（V チーム）

5.1 目的

本実験において、何を確認するために、どのような工程成果物に対して、どの形式手法を用いたかを表 5-1　目的に記す。なお、当チームでは作業者ごとに確認の観点に差異があるため、より具体的な確認事項と確認方法は 5.3.4 節を参照のこと。

表 5-1　目的

採用した手法	VDM++	
（特徴）	VDM++は、事後条件を記述することで「何をしたいか」を明確化することができ、不変条件と事前条件で記述対象の制約条件を記述できる。VDMTools では、静的チェックと証明課題生成、及び回帰テストと組み合わせテスト（Combinatorial Test）による動的チェックで、要求される機能と共に、その制約条件を検証することができる。 VDM++記述自体を要求辞書（リアルタイム構造化分析･設計手法で導入された考え方で、名詞と述語の両方を含んだ用語辞書）とすることができるので、曖昧な用語を早期に排除できる。 また、システムないし対象領域（ドメイン）全体の要件を記述でき、日本語識別子が使用できるため､擬似コードを使用した日本語に近い形の形式記述を作成することで、多くのエラーを発見できる。 結果として、対象システムの品質を高く保ちながら、コミュニケーションロスを防いで高い生産性を実現することができる。	
確認すること	要求辞書の整合性、対象システムの機能と制約条件、仕様の漏れを検証する。	
（確認方法）	データと陰仕様を定義し、静的チェックを行うことで、要求辞書としてのVDM++記述を検証する。 陽仕様を追加して、動的チェックを行うことで、対象システムの機能と制約条件を検証する。 生成された証明課題をレビューすることで、仕様の漏れ、特に制約条件の漏れを抽出する。	
確認の対象	**成果物**	**形式記述**
	外部設計書・画面	操作と関数、事前／事後条件
	外部設計書・エンティティ定義	型と変数、およびそれらの不変条件
（備考）	上記は「機能要件の合意形成ガイド」(IPA SEC)における下記の技術領域・工程成果物に相当する。	

	・システム化振舞い・システム化業務説明 ・データモデル・エンティティ定義

5.2 体制

本実験を行った作業者の役割やスキルなどを表 5-2 に記す。

表 5-2 作業者のスキル

作業者	役割*	スキル （経験）			
		業務 AP 開発	類似 AP 開発	形式手法	採用手法
A	形式記述者／形式検証者	2 年	0 年	5 年	5 年
B	形式記述者／形式検証者	2 年	0 年	5 年	1 年
C	形式記述者／形式検証者	6 年	6 年	20 年	2 年
D	形式記述者／形式検証者	0 年	0 年	3 年	2 年
E	形式記述者／形式検証者	37 年	25 年	17 年	12 年

* 役割とその担当する作業は以下のとおり。

- 形式記述者 ： 『形式記述』の作成と修正を行う
- 形式検証者 ： 『形式記述』に関する検証を行う

5.3 作業

5.3.1 作業の概要

本実験では、図 5-1 のような一連の作業を行った。

ただし、抽出した指摘事項を受けた外部設計書の修正は実施しないため、図のうち「工程成果物への反映」は今回の実験の対象外となる。

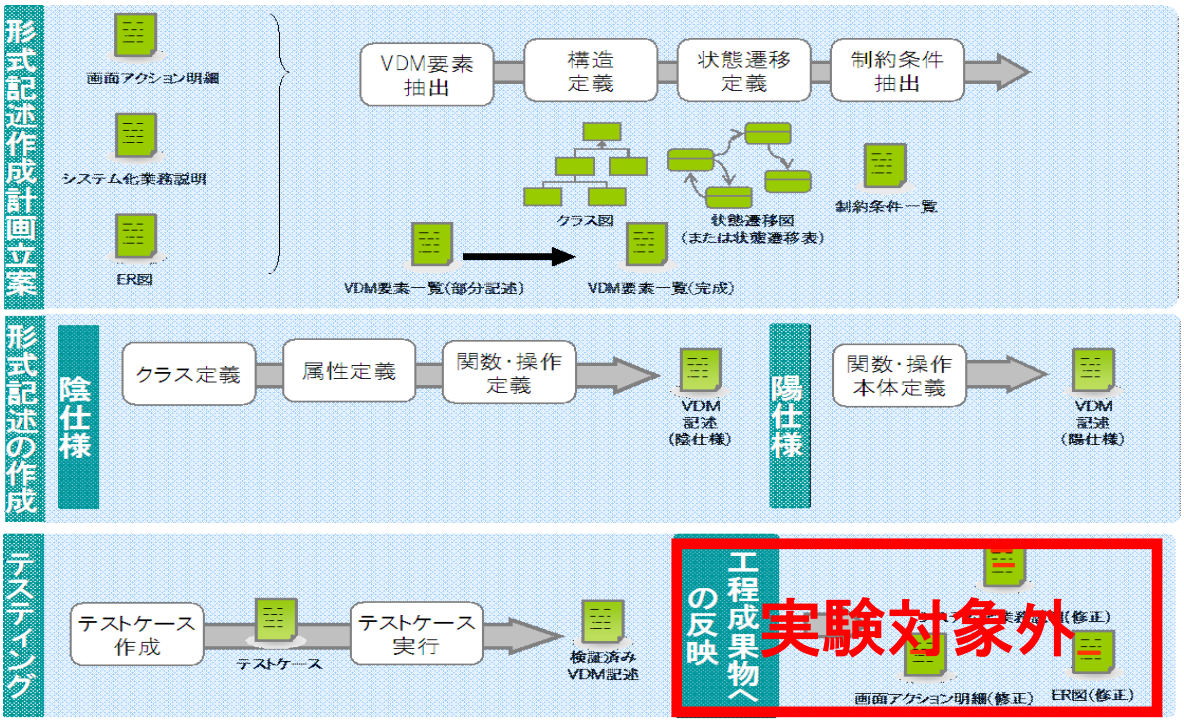


図 5-1 作業の概要

各作業の内容については『形式手法適用手順（VDM++編）【リリース版】』 [5]を参照のこと。

5.3.2 実験対象範囲と分担

当チームでは、5 名の作業者が図 5-2 に示す作業分担により実験を行った。外部設計書のうち、コーポレートアクション情報に対する登録・削除・変更・検索・強制削除の各機能とこれらに関わる共通業務、および参照するテーブルを本実験の対象としている。

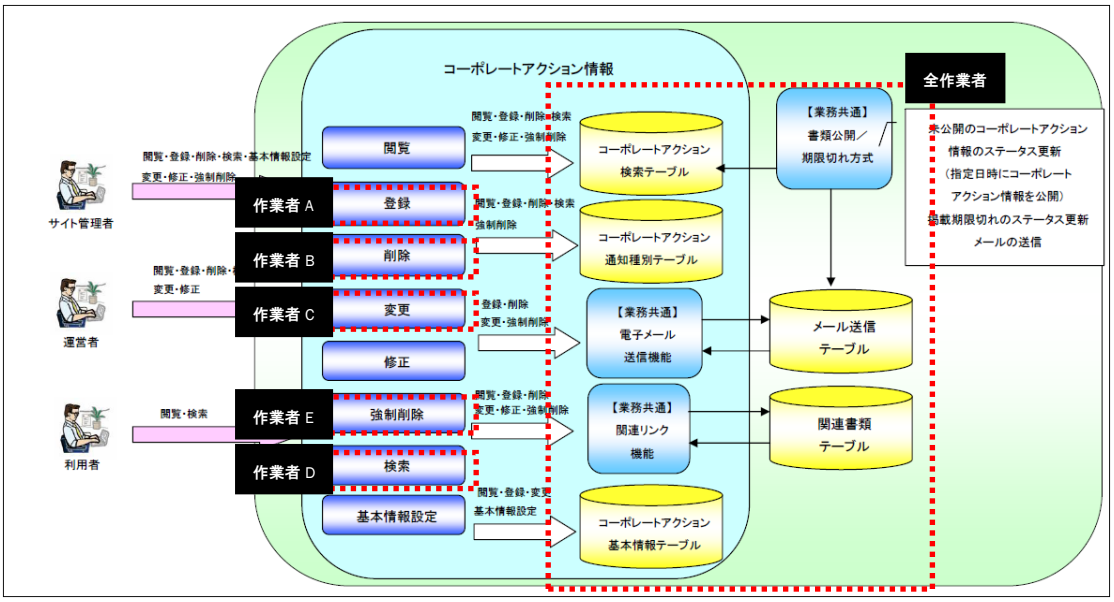


図 5-2 対象範囲と分担

全作業者が実験対象とする各種テーブルについては、形式記述作成の効率化を図るためにデータベース操作を擬似的に実現するフレームワークを事前に作成した。詳細は 5.3.3 節で述べる。

5.3.3 形式記述のフレームワーク

✧ **課題**

- DB への CRUD 処理を VDM でシミュレートする方法
- DBFramework 関連の VDM を記述する作業を省力化

✧ **対策**

I. DB への CRUD 処理を VDM でシミュレート

今回の実験対象は、Web システムであり、何らかユーザーが入力したデータを受け取り、編集を施した上で DB に保存することが主な機能と言える。
そのため、VDM において DB への CRUD 処理（Create, Read, Update, Delete）をどのように取り扱うかが課題となっていた。
今回は、DB の動作を記述・検証することが目的ではないため、DBFramework という DB の動作を簡単にシミュレートするフレームワークを適用した。

II. DBFramework 関連の VDM 記述を省力化

採用したフレームワークである DBFramework は、各テーブルの定義項目を VDM で記述しなければならないため、記述量が多くなる。今回のように作業工数が限られている場合に、

これらを手作業で対応するのは効率的ではないため、今回は DBFramework の各テーブルにあたる VDM 記述を、Excel 表から自動生成するツールを作成した。

✧ **結果**

対策で実施した 2 つの事により、DB をシミュレートした VDM 形式記述を簡単に作成することと、そのフレームワークに合わせた記述を自動生成することが可能となった。

5.3.4 作業の詳細

5.3.4.1 作業者 A

✧ **目的**

対象システムで扱われる電子書類「コーポレートアクション情報」における、コーポレートアクション情報の登録処理に関する、指摘事項、問題点の抽出を目的とした。

✧ **記述対象**

コーポレートアクション情報登録機能における画面アクション明細に記述されたビジネスロジックを対象とした。

✧ **形式記述の作成**

I. **記述方針**

(a) **課題**

設計書にある指摘事項を限られた工数の中で VDM を用いて抽出すること

(b) **対策**

以下のような記述方針を策定。

- 画面におけるアクションをできるだけ忠実に記述する
  設計書に存在する指摘事項を抽出することを目的としているため、過度に抽象化してしまうと設計書の内容と乖離してしまう可能性がある。
  そのため、設計書の画面アクションとして記述されている内容を忠実に VDM で記述することとした。
- 整理された形式記述を目指すのではなく、冗長であっても設計書と対応付けやすい記述を心がける
  これも目的から来る方針である。整理された形式記述は、形式記述としての分かりやすさがある反面、検証対象である設計書との対応付けが取りづらいと判断した。
  そのため、たとえ冗長な記述であっても、設計書との対応付けを重視した形式記述を心がけた。

- **陰仕様は【クラス図】からの自動生成とする**
  これは、今回の実験に割くことができる工数が限られていることから取った方針である。
  今回は陰仕様を明示的に記述するのではなく、クラス図から自動生成したものを陰仕様と位置付け、手作業での形式記述は陽仕様としている。

(c) 結果

上記対策の結果、設計書に忠実かつ対応を取りやすい形式記述をスケジュール通りに作成でき、3 件の指摘事項を抽出した。

II. 記述の流れ

本実験において、作業者 A は次のような一連の作業を行った。

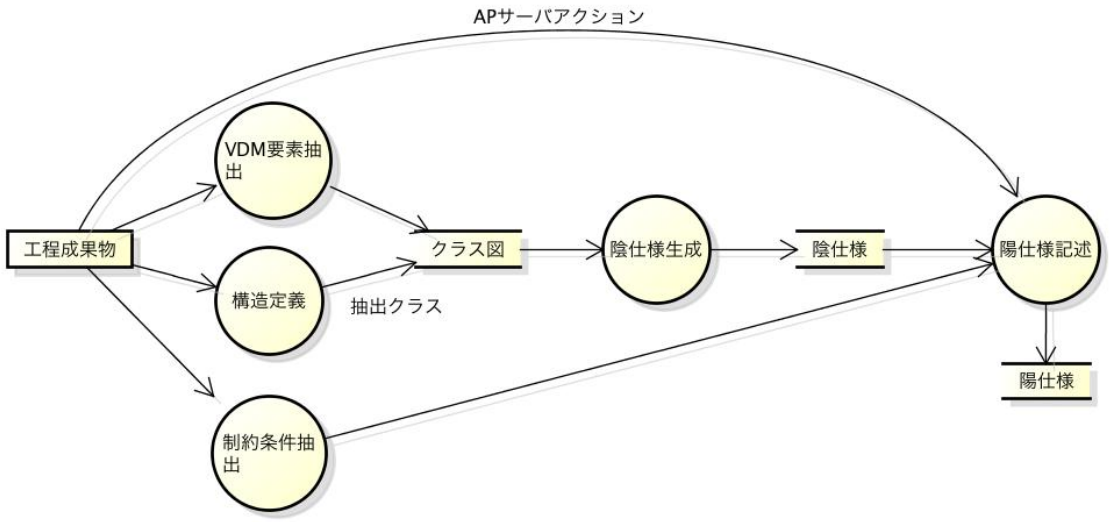


図 5-3 作業者 A の手順

上記の手順は、『形式手法適用手順（VDM++編）【リリース版】』 [5]**をベースにしている。その中で、本実験の題材や提供された設計書に応じて、以下のようなテーラリングを実施した。**

【VDM 要素抽出】を作業として実施するが、『VDM 要素一覧』として一覧化はせず、【構造定義】の成果物である『クラス図』に記述することと、【クラス定義】・【属性定義】・【関数・操作本体定義】時に直接記述することを実施した。
これは、実験工数の縮小を狙ったもので、記述対象に現れた用語を、都度記述する方式を取った。これにより、『VDM 要素一覧』と『形式記述』の間でトレーサビリティを取ることを省力化することが可能となった。

次に、【制約条件抽出】を作業として実施するが、『制約条件一覧』として一覧化はせず、【クラス定義】・【属性定義】・【関数・操作本体定義】時に直接記述することを実施した。
これも、実験工数の縮小を狙ったもので、『VDM 要素一覧』と同じく、『制約条件一覧』と『形式記述』の間でトレーサビリティを取ることの省力化に寄与した。

なお、これらの中間成果物省略による工数の縮小は、本作業者が VDM に慣れており、上記中間成果物を作成しなくとも、VDM を記述できることから実施している。あまり VDM に慣れていない作業者が中間成果物を省略すると、入力設計書からの情報をまとめきれず、記述された VDM が整理されず混乱を招く恐れがある。

また、前述したように、今回は明示的に陰仕様を記述せず、クラス図から生成できる範囲については、VDMTools が持つ UML リンク機能を用いて陰仕様を自動生成した。それ以外の部分については、全てにおいて陰仕様記述→陽仕様記述のステップを踏むのではなく、始めから陽仕様を記述しつつ、設計書で理解できない部分や記述が足りない部分、不明確な部分を is not yet specified とした。
これにより、形式記述化できていない部分を明確にし、かつ記述抜けを防ぐことが可能となった。
陽仕様は、【工程成果物】をベースに記述を進めた。これは、設計書の検証を目的としているため、できるだけ抽象化せず、設計書に忠実に記述をすることを目的としたためである。
そのため、陰仕様生成後の陽仕様記述時、クラス分割やクラス統合が必要になった場合、【クラス図】に立ち戻るのではなく、直接【形式記述】上で反映させている。
これも、実験工数の縮小を狙ったため取った施策である。

III. 中間成果物（パッケージ図、クラス図）

(a) 課題

他の作業者が形式記述を作成する際にも記述しやすい構造を作ること。

(b) 対策

記述の流れにあるように、形式記述前の中間成果物として、VDM の構造を表すクラス図を作成している。本節では、クラス図とそのアウトラインを表すパッケージ図を掲載する。

- パッケージ図

まずは、作成したパッケージ図を図 5-4 に示す。

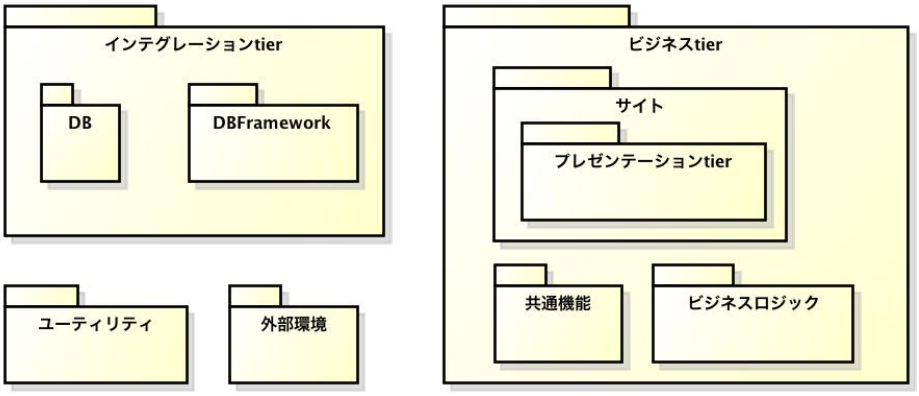


図 5-4 パッケージレベルの構造

上記のようにパッケージを分割したのは、対象システムのソフトウェアアーキテクチャについて説明している「オンライン共通基盤」にある本システムのアーキテクチャを以下のように解釈したためである。

対象システムは、J2EE の 4 層モデルを基にしたアーキテクチャを採用しており、その中でサーバーサイドの構造はインテグレーション tier・プレゼンテーション tier・ビジネス tier と名付けられている。

対象システムのアーキテクチャを踏まえた上で、今回の実験におけるパッケージ構成は、3 つの tier とそれらが用いるシステム共通機能、補助ユーティリティ等で構成されている。また、特徴として、インテグレーション tier に前述した DBFramework というフレームワークを採用している点が挙げられる。

- クラス図

続いて、パッケージ図を詳細化したクラス図を図 5-5 に示す。

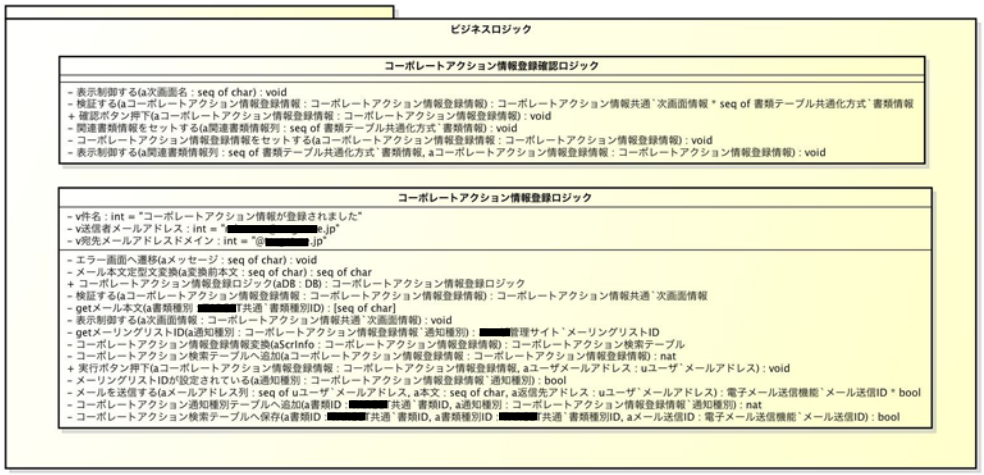


図 5-5 ビジネスロジックのクラス構造

上記は、今回の記述対象の中心である、ビジネスロジック部分を抜き出したものである。
設計書の流れに従い、以下の流れを中心として記述をしている。

- 検証
- 操作
- 表示制御

(c) 結果

これらの構造を用いた結果、設計書に忠実な形式記述を作成することができ、
また作業者 5 名中 3 名も本構造を用いて記述することができた。

IV. 特徴的な記述

(a) 課題

設計書に忠実な VDM による形式記述の作成

(b) 対策

今回記述した形式記述の内、特徴的な記述を以下に掲載する。

```
-- 4 DTRSSCA0202
-- 4-1 コーポレートアクション情報登録画面で入力された値を
-- コーポレートアクション検索テーブルに追加
def
  ret = コーポレートアクション検索テーブルへ追加(a コーポレートアクション情報登録情報);
in
  -- 5
  -- エラー画面へ遷移，メッセージは「{登録}に失敗しました。」
  if ret = 0 then エラー画面へ遷移("MSGTRBCBC40802") else skip;

-- 6 DTRSSCA0203
-- 6-1 コーポレートアクション登録画面で選択した通知種別を
--       コーポレートアクション通知種別テーブルに追加
def
  ret = コーポレートアクション通知種別テーブルへ追加(w 書類 ID,
                                    a コーポレートアクション情報登録情報.i 通知種別);
in
  -- 7
  -- エラー画面へ遷移，メッセージは「{登録}に失敗しました。」
  if ret = 0 then エラー画面へ遷移("MSGTRBCBC40802") else skip;
```

画面アクション明細の項番に合わせて、コメントに記述箇所を明記

設計書記述内容をコメントに記述し、合わせて呼び出す操作名もできるだけ設計書表記に合わせる

図 5-6 特徴的な VDM 記述

対象機能は、コーポレートアクション検索テーブルとコーポレートアクション通知種別テーブルという 2 テーブルへ入力データを保存し､保存結果を制御することを担っており、上記はそのテーブルへ保存し、表示を制御している部分である。

(c) 結果

設計書の流れに従って記述を進めつつ、DB への保存処理等は別操作に定義することで、設計書との対応付けと読みやすさを実現することができた。

✧ 静的検査

今回は、VDMTools を用いた構文チェックと型チェックを静的検査として実施した。静的検査によって指摘事項は抽出されなかったが、自らの記述ミスがないことを確認することができた。

✧ 動的検査

VDM++を用いてテスト実行することで、動的な振る舞いでの欠陥を検出しようとしたが、以下の理由により今回は実施しなかった。

I. 【動的検査を実施しない理由】

対象は、個別業務を記述した設計書をベースに、システム全体で共通の機能を記述した設計書を用いながら画面アクションを記述している。

そのため、システム全体で共通の機能を記述した設計書の機能が重要な役割を担うのだが、システム全体で共通の機能を記述した設計書は機能概要とインターフェースのみ記述されていることが多く、動作可能な形式記述を作成するには情報が足りない。

また、個別業務を記述した設計書のアクションは、かなり詳細レベルの記述がされており、これを忠実に記述するという方針の下では、実装するのと変わらなくなってしまう。この場合、実験にかかる工数が増大し、所定の期間内に実験を終了させることができないため、陽仕様を動作させるレベルまでは記述しなかった。

✧ 抽出した問題点

実験を進める中で、指摘事項ではないが、対象の設計書にいくつか問題点が抽出された。ここでは、その問題点について記す。

● 用語のブレ

自然言語で記述された設計書に良く見られるが、今回の設計書も用語のブレが多く見られた。似たような用語の別定義や用語の字面から想定する意味と違う用語等が見られた。

- 似たような用語の別定義
  - メール送信 ID と送信メール ID
- 字面から想定する意味と異なる用語
  - 通知種別、書類種別 ID が機能/階層 ID と同等の意味を持つ

これらのブレは、複数の設計書間で顕著に見られ、その中でも特に個別業務を記述した設計書とシステム全体で共通の機能を記述した設計書の間に多く見られた。

- **設計書の粒度**

設計書の位置づけを明確にし、各設計書の粒度を揃えることで、その設計書は受け取った人に伝わりやすくなる。しかし、今回の設計書は粒度にバラつきが見られた。

この特徴を端的に表すと、「設計時に判明していることをできるだけ詳細に記述している」ようであった。そのため、設計時点で詳細まで判明していると思われる部分については、そのまま実装ができる程の詳細度であったが、反面あまり詳細まで判明していなかったと思われる部分については、「詳細は詳細設計以降で検討する」のような記述や、機能のインターフェースのみ記述したような部分が見られた。

- **検証容易性の欠如**

前述した通り、今回の実験では動的検査を実施しなかった。この理由についても前述したが、この動的検査を実施できないことと用語のブレ等の設計書が持つ問題点は、検証容易性が極めて低いという特性として現れている。

また形式記述を進める中で、指摘事項と思われる事象が 3 点抽出された。ここでは、その 3 点を説明する。

I. コード設計に存在しないコードの定義

(a) 抽出した内容

コード設計に存在しないコードを個別業務を記述した設計書にて参照している指摘事項

(b) システムへの影響

コード設計には、個別業務を記述した設計書に記載されている「権利種類コード」の定義が存在しないことから、コーポレートアクション情報登録画面において、権利種類リストを選択するドロップダウンリストに必要な項目が表示されず、適正な登録がされない可能性がある。

(c) 抽出方法

個別業務を記述した設計書の流れに沿って設計書を理解しつつ VDM を記述している過程において抽出した

II. 画面設計書とコード設計書の齟齬

(a) 抽出した内容

個別業務を記述した設計書に記載されているドロップダウンリストの項目定義と画面イメージ、コード設計の項目定義に齟齬がある。

まず、個別業務を記述した設計書の画面イメージには、図 5-7 のようにドロップダウンリストの項目として、「外国株券・参加者通知」が表示されている。

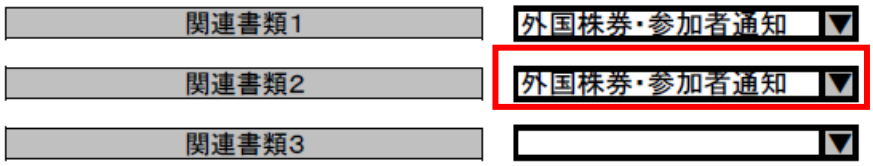


図 5-7 個別業務を記述した設計書における定義

一方、同設計書の図 5-8 にある項目定義には、ドロップダウンリストの選択肢が全て定義されており、そこには「ブランク」「参加者通知」のみとなっており、画面イメージにある「外国株券・参加者通知」という選択肢がない。さらに、項目定義には「※2 汎用コードマスタから取得。」とある。

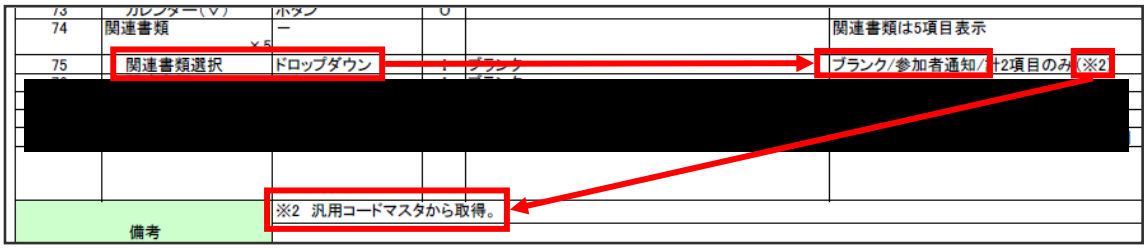


図 5-8 個別業務を記述した設計書における項目定義

そこで、汎用コードが定義されている設計書であるコード設計を見ると、図 5-9 のように、個別業務を記述した設計書の「関連書類選択」にあたる項目がない。似たような項目として「関連書類種別」があるが、これは個別業務を記述した設計書の項目定義とは全く異なる定義である上、「関連書類種別」が 2 箇所に定義されている。

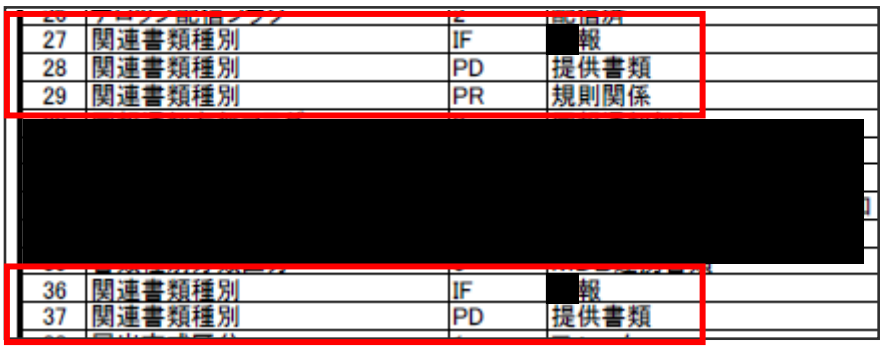


図 5-9 コード設計における定義

(b) システムへの影響

関連書類を選択するドロップダウンリストに必要な項目が表示されず、適正な登録がされない可能性がある。

(c) 抽出方法

個別業務を記述した設計書の流れに沿って設計書を理解しつつ VDM を記述している過程において抽出した

III. 個別業務を記述した設計書とシステム全体で共通の機能を記述した設計書の齟齬

(a) 抽出した内容

共通機能であるメール送信とそれを用いるコーポレートアクション情報登録の間にあるコード定義に齟齬がある。

本指摘事項は、3 つの設計書で同じ項目が異なる定義をされていることである。

1 つ目は、個別業務を記述した設計書であり、あるアクションを実行する過程において、システム全体で共通な電子メール送信機能を用いる部分である。ここでは、図 5-10 のように、送信フラグのコード体系は「1:即時公開」「2:定時点公開・指定時間公開」となっている。

枝番	処理
12-1	共通機能 メール送信
12-2	取得したメールアドレスに対し、正常11で取得したメール内容を登録する(業務共通) (メール送信機能インターフェース入力内容)
	送信者アドレス: [redacted]
	返信先アドレス: ユーザ管理サイトより取得(当該書類登録者ユーザID[redacted]) ※アクションID:ATRSSCA0203の正常1で取得したメールアドレス
	宛先×n: 正常8の結果。(宛先ユーザID[redacted])
	件名: [コーポレートアクション情報]が登録されました。
	本文: 正常11の結果
	送信フラグ: 即時公開:1、定時点公開・指定時間公開:2 をセット。

図 5-10 個別業務を記述した設計書における送信フラグの定義

次に、システム全体で共通の電子メール送信機能の設計書には、図 5-11 のように送信フラグは「1:未送信」「2:送信待機」とされている。この時点で、両者の用語は全く異なっている。

インターフェース	インターフェースID	入力	出力
メールDB更新	FTRBCBC1601	送信メール情報 ・宛先メールアドレス×n ・宛先名×n ・差出人メールアドレス ・差出人名 ・返信先メールアドレス ・返信先名 ・件名 ・本文 ・送信フラグ (1:未送信、2:送信待機)	メール送信ID

図 5-11 システム全体で共通の機能を記述した設計書における定義

最後に、データベースの設計書にも同じ送信フラグに関する定義がされており、図 5-12 のように「1:未送信」「2:送信済み」「3:送信失敗」「4:削除」「5:送信待機」となっており、システム全体で共通の機能を記述した設計書の方式設計とは、「送信待機」の値が異なっている。

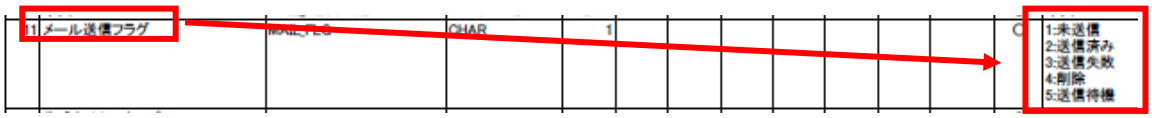
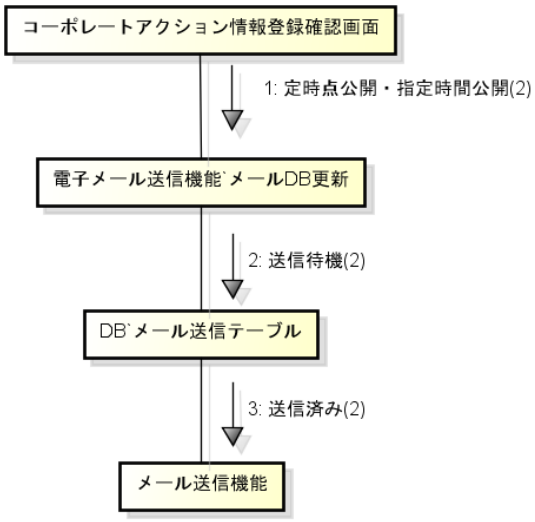


図 5-12 DB 設計における定義

(b) システムへの影響：

図 5-13 のように、メール送信機能プロセスがメール送信テーブルを参照する際、メール送信フラグが「2（送信済み）」で登録されているため、メールが送信されない。



- コーポレートアクション情報登録確認画面より、「定時点公開･指定時間公開」を想定し、「2（定時点公開･指定時間公開）」をシステム全体で共通の機能を記述した設計書の電子メール送信機能 / メールDB更新へ送る
- 電子メール送信機能 / メールDB更新は、メール送信フラグとして受け取った引数「2（送信待機）」を、DBのメール送信テーブルへ保存
- メール送信機能プロセスがメール送信テーブルを参照するが、メール送信フラグが「2（送信済み）」で登録されているため、メールが送信されない

図 5-13 指摘事項を引き起しうるシーケンス

(c) 抽出方法

個別業務を記述した設計書の流れに沿って設計書を理解しつつ VDM を記述している過程において抽出した。

5.3.4.2　作業者 B

✧ **目的**

コーポレート情報削除アクションの業務ロジックとその事前条件／事後条件を記述し、設計書に示された設計情報の抜け漏れや曖昧さ、矛盾点などを抽出する。

✧ **記述対象**

コーポレートアクション検索テーブルおよびコーポレートアクション通知種別テーブル、およびそれらに対する更新や削除等の操作を含むコーポレート情報削除アクションを記述対象とする。また関連する共通業務（電子メール送信機能、関連リンク機能）についても可能な範囲で記述する。

コーポレートアクションにおける書類の状態に関しては、公開状態、削除状態、論理削除フラグを記述対象とする。削除対象の書類の状態によって操作に複数のバリエーションが存在し、結果も異なることから、本実験ではこれらの状態遷移の適切な事前条件と事後条件に着目する。

✧ **形式記述の作成**

形式記述の作成にあたり、以下の記述方針を定めた。

I. **アクション定義に示された業務ロジックを主な記述対象とする**

個別業務の基盤となっているユーザ権限や、画面遷移などは記述対象としない。外部設計書においても適切な権限が設定されていることを前提としていることや、他チームの記述範囲であることから、作業の優先順位が低いと判断したためである。

II. **外部設計書の内容を忠実に記述する**

仕様が明確でない部分を推測や頻繁な Q&A によって補うことはせず、明確となっている部分のみを記述する。明確でない、あるいは実験目的に照らして重要でない仕様は陰仕様として記述する。

ただし、外部設計書において実装を意識した内部用コードに近い命名がされているアクションや述語に関しては、可読性を考慮して要求仕様レベルの文言に置き換える。

III. **正常系のみ記述する**

異常とならない条件を事前条件として表現し、エラー処理は明に記述しない。コーポレート情報削除アクションにおける異常系処理はすべてエラーメッセージの出力とアクション中止であり、記述の優先度は低いと判断したためである。

また、『形式手法適用手順（VDM++編）【リリース版】』 [5]が定める作業手順に対し、以下のようなテーラリングを行った。いずれも、入力となる外部設計書の理解を進めながら形式記述を作成するという本実験固有の制約に基づくものである。

- 構造定義および状態遷移定義を省略
- 陰仕様から陽仕様へという記述の流れではなく、情報量の多寡によって最初から両者を使い分ける
- クラス定義や属性定義をトップダウンに行うのではなく、必要な情報を適宜追加していくボトムアップ型の手順を取る

テーラリング後の作業の流れを以下に示す。

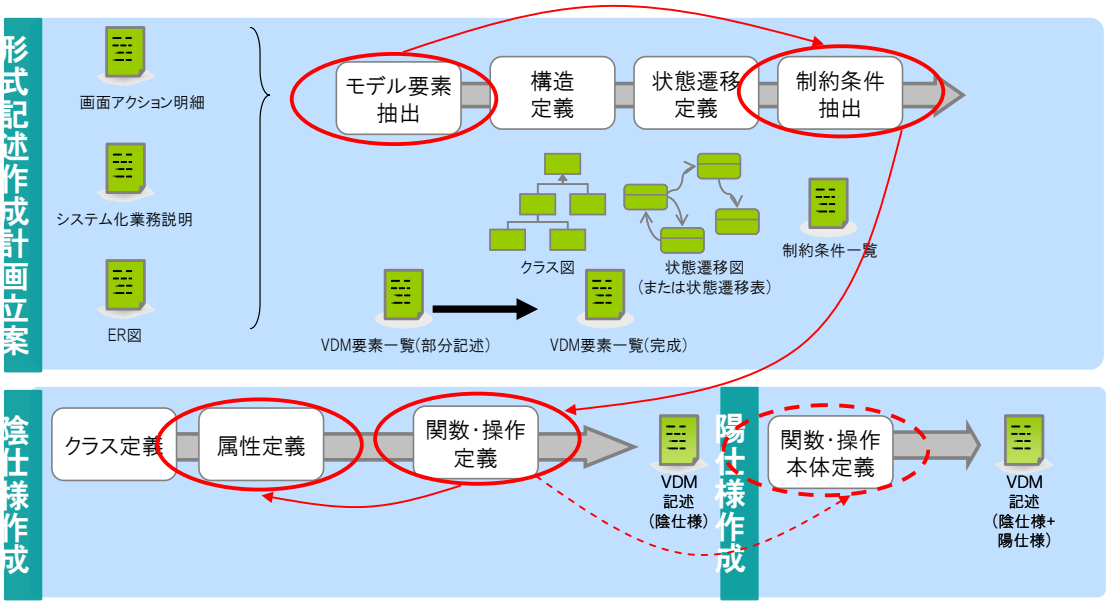


図 5-14 テーラリング後の作業の流れ

作成した形式記述は、以降に示すようにアクションのメインフローと、および外部設計書において“【検】”、“【正】”のラベルで示されるパターン（アクションの詳細）の大きく２つに分かれる。図 5-15 に、両者の典型例と対応する形式記述を示す。

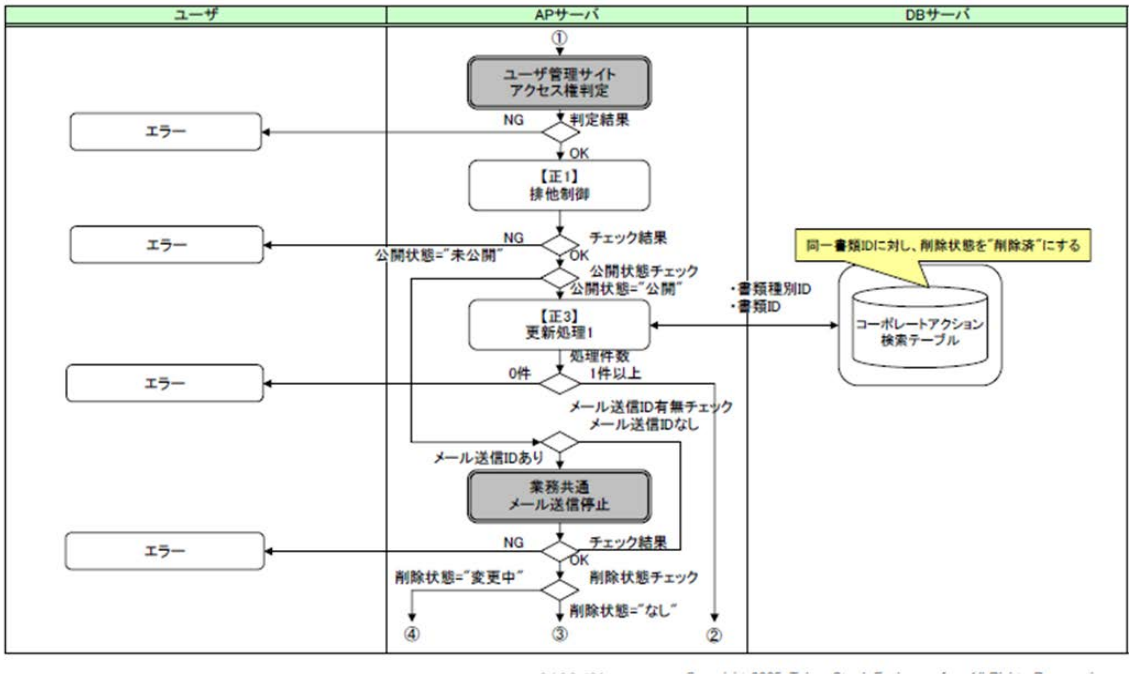


図 5-15 コーポレートアクション情報削除メインフロー（一部）

図 5-15 に対応する形式記述は図 5-16 のとおり。

```
operations
public 《ATRSSCA0301》コーポレートアクション情報削除アクション : ユーザ * 書類種別ID * 書類ID * 版数 * set of コーポレートアクション検索テーブル * set of 関連書類テーブル ==> set of コーポレートアクション検索テーブル * set of 関連書類テーブル
《ATRSSCA0301》コーポレートアクション情報削除アクション(aユーザ, a書類種別ID, a書類ID, a版数, a検索DB, a関連DB) ==
(
  dcl o検2 : コーポレートアクション検索テーブル;
  dcl o検4 : bool;
  dcl o検6 : set of 通知種別ID;
  dcl o検8 : bool;

  dcl o正1  : bool;
  dcl o正3  : set of コーポレートアクション検索テーブル;
  dcl o正5  : bool;
  dcl o正7  : set of コーポレートアクション検索テーブル;
  dcl o正9  : set of コーポレートアクション検索テーブル;
```

```
  dcl o正10 : set of コーポレートアクション検索テーブル;
  dcl o正12 : 関連書類テーブル;
  dcl o正14 : set of コーポレートアクション検索テーブル;
  dcl o正16 : set of 関連書類テーブル;
  dcl o正17 : set of 関連書類テーブル;

  o検2 := 【検2】《DTRSSCA0104》コーポレートアクション情報を取得する(a書類ID, a版数, a書類種別ID, a検索DB);
  o検4 := 【検4】《FTRBCBC1901》削除可能であるか判定する(o検2, aユーザ);
  o検6 := 【検6】《DTRSSCA0105》通知種別IDを取得する(a書類ID, a検索DB);
  o検8 := 【検8】《ITRUMOF0902》登録権の有無を判定する(o検6, aユーザ);
  o正1 := 【正1】排他チェックする();
  if o検2.get公開状態() = <公開>
  then (
    o正3 := 【正3】《DTRSSCA0301》公開書類を削除する(o検2, a検索DB);
    o正17 := 【正17】《FTRBCBC1104》関連書類情報を削除する(o検2, a関連DB);
    return mk_(o正3, o正17);
  )
  else if o検2.get公開状態() = <未公開>
  then (
    o正5 := 【正5】《FTRBCBC1602》メール送信を停止する(o検2);
    if o検2.get削除状態() = <なし>
    then (
      o正7 := 【正7】《DTRSSCA0302》未公開書類を完全に削除する(o検2, a検索DB);
      o正9 := 【正9】《DTRSSCA0603》当該書類の論理削除フラグを立てる(o検2, o正7);
      o正17 := 【正17】《FTRBCBC1104》関連書類テーブルを削除する(o検2, a関連DB);
      return mk_(o正9, o正17);
    )
    else if o検2.get削除状態() = <変更中>
    then (
      o正10 := 【正10】《DTRSSCA0303》未公開書類を削除する(o検2, a検索DB);
      o正12 := 【正12】《DTRSSCA0304》関連書類情報を取得する(a書類種別ID, a書類ID, a公開状態, a削除状態, a検索DB);
      o正14 := 【正14】《DTRSSCA0305》変更中の状態を解除する(o検2, o正10);
      o正16 := 【正16】《FTRBCBC1105》関連書類テーブルを更新する(o正12);
      return mk_(o正14, o正16);
```

```
    )
    else
      skip;
  )
  else
    skip;
)
```

図 5-16 コーポレートアクション情報削除メインフローの形式記述

アクション詳細の例として、検証パターンを図 5-17 に示す。

パターン	No	アクション詳細ID	枝番	説明
検証	1			
	2	(DTRSSCA0104)	2-1	入力値(書類ID、版数、書類種別ID)を条件に、コーポレートアクション検索テーブルより コーポレートアクション情報を取得
	3	-	3-1	検証2の結果、
			3-2	該当レコードなしの場合、
			3-3	エラーメッセージ(MSGTRBCBC40940:"指定された[書類]が見つかりません。")をセット
			3-4	エラー画面へ遷移し、終了
	4	共通(FTRBCBC1901)	4-1	検証2で取得したデータのステータスをもとに、削除可能なコーポレートアクション情報かを判定(業務共通)
	5	-	5-1	検証4の結果、NGの場合
			5-2	エラーメッセージ(MSGTRBCBC40951:アクセス権がない為、[削除]できません。)をセット
			5-3	エラー画面へ遷移し、終了
	6	(DTRSSCA0105)	6-1	入力値(書類ID)を条件に、コーポレートアクション通知種別テーブルより通知種別IDを取得
	7	-	7-1	検証6の結果
			7-2	該当レコードなしの場合、
			7-3	エラーメッセージ(MSGTRBCBC40940:指定された[書類]が見つかりません。)をセット
			7-4	エラー画面へ遷移し、終了
	8	共通(ITRUMOF0902)	8-1	検証6で取得したすべての通知種別IDに対し、登録権があるか確認
			8-2	・すべての通知種別IDに対し、登録権がある場合、判定結果を"OK"とする
			8-3	・すべての通知種別IDに対し、一つでも登録権がない場合、判定結果を"NG"とする
	9	-	9-1	検証8の結果、NGの場合
			9-2	エラーメッセージ(MSGTRBCBC40951:アクセス権がない為、[削除]できません。)をセット
			9-3	エラー画面へ遷移し、終了

図 5-17 アクション詳細例（検証パターン）

図 5-17 に対応する形式記述は図 5-18 のとおり。

```
functions
public 【検2】《DTRSSCA0104》コーポレートアクション情報を取得する : 書類ID * 版数 * 書類種別ID * set of コーポレートアクション検索テーブル -> コーポレートアクション検索テーブル
【検2】《DTRSSCA0104》コーポレートアクション情報を取得する(a書類ID, a版数, a書類種別ID, aコーポレートアクション検索テーブル集合) == is not yet specified
pre
  exists1 t in set aコーポレートアクション検索テーブル集合 & t.get書類ID() = a書類ID and t.get版数() = a版数 and t.get書類種別ID() = a書類種別ID
post
  RESULT.get書類ID() = a書類ID and RESULT.get版数() = a版数 and RESULT.get書類種別ID() = a書類種別ID;
```

```
functions
public 【検4】《FTRBCBC1901》削除可能であるか判定する : コーポレートアクション検索テーブル * ユーザ -> bool
【検4】《FTRBCBC1901》削除可能であるか判定する(aコーポレートアクション検索テーブル, aユーザ) ==
is not yet specified
post
  RESULT = true;

functions
public 【検6】《DTRSSCA0105》通知種別IDを取得する : 書類ID * set of コーポレートアクション検索テーブル -> set of 通知種別ID
【検6】《DTRSSCA0105》通知種別IDを取得する(a書類ID, aコーポレートアクション検索テーブル集合) ==
is not yet specified
pre
  exists t in set aコーポレートアクション検索テーブル集合 & t.get書類ID() = a書類ID
post
  forall id in set RESULT & exists t in set aコーポレートアクション検索テーブル集合 & t.get書類ID() = a書類ID and t.get通知種別ID() = id;

functions
public 【検8】《ITRUMOF0902》登録権の有無を判定する : set of 通知種別ID * ユーザ -> bool
【検8】《ITRUMOF0902》登録権の有無を判定する(a通知種別ID集合, aユーザ) ==
  forall id in set a通知種別ID集合 & 登録権を持っている(aユーザ, id);
```

図 5-18 検証パターンの形式記述

✧ 静的検査

形式記述（陰仕様・陽仕様）の作成時に、VDMTools による構文チェック、型チェックを実施した。

✧ 動的検査

一部の仕様について事後条件を抽出するための情報が足りず、陽仕様の記述が不可能であることが判明したため、動的検査は実施しないこととした。

✧ 抽出した問題点

形式記述の作成段階において、以下の問題点を抽出した。

I. アクセス制御仕様の漏れ（設計の不足）

(a) 抽出した内容

コーポレートアクション情報削除時のアクセス制御仕様が不足している。後の Q&A により画面表示制御のマトリクスがアクセス制御仕様を兼ねていることが判明したが、そのことが明記されていない。（発注者による評価の結果、不具合ではないことが判明）

(b) システムへの影響

アクセス制御仕様が設計情報の不足により適切に実装されず、利用者に意図しないアクセスを許してしまう可能性がある。

(c) 抽出方法

形式記述の陰仕様作成時に、検証パターンの事前条件と事後条件を洗い出す際に抽出した。

II. 書類状態の組み合わせバリエーションの漏れ（設計の不足）

(a) 抽出した内容

公開状態と削除状態の組合せについて、以下のケースを除く組合せにおける処理の仕様が不足している。

・公開状態が「公開」
・公開状態が「未公開」かつ削除状態が「なし」
・公開状態が「未公開」かつ削除状態が「変更中」

（発注者による評価の結果、不具合ではないことが判明）

(b) システムへの影響

未公開書類を削除する際、要求と異なる動作を引き起こす可能性がある。

(c) 抽出方法

形式記述の陰仕様作成時に、メインフローと各詳細アクションの事前／事後条件の対応を確認する際に抽出した。

III. 用語の曖昧な定義（設計の曖昧さ）

(a) 抽出した内容

「当該書類と同一の書類 ID を持つ書類」という記述は、当該書類自身も含んでいるか否かが曖昧である。外部設計書全般で用いられている文言であるため、単なる文章表現上の問題ではなく曖昧な設計による指摘事項であると判断した。

(b) システムへの影響

削除対象の書類と、削除対象の書類と同じ書類 ID を持つそれ以外の書類の整合性が失

われ、意図した削除動作が行われない可能性がある。

(c) 抽出方法

形式記述の陽仕様作成時に、事後条件を処理の記述に落としこむ段階で抽出した。

5.3.4.3 作業者 C

✧ 目的

対象システムで扱われる電子書類「コーポレートアクション情報」の、改版を伴う変更処理(以下「変更処理」)に関する、指摘事項、問題点の抽出を目的とした。

✧ 記述対象

コーポレートアクション情報についての変更に関する処理は、対象システムにおいては以下の画面遷移で行われる：

(1) 一覧画面→(2)詳細画面→(3)変更画面→(4)変更確認画面→(5)変更完了画面

このうち、実際の変更処理が行われるのは、「(4)変更確認画面」から「(5)変更完了画面」への画面遷移の間である。

そこで、変更確認画面から変更完了画面への画面遷移処理を中心に、そこから呼ばれる下位処理、およびその関連処理を記述対象とした。さらに、それらの処理と関係する、エンティティ(DB テーブル)および画面も記述対象とした。またその処理の中で扱われる値(各種区分等)についても抽象化した形で記述した。

✧ 形式記述の作成

I. 記述方針

対象とした設計書群は膨大であり、また、適用手順で想定していた「機能要件の合意形成ガイド」に、完全に従った形で記述されているわけではなかった。そのような性質を持った設計書群の理解と、それにもとづく VDM++記述の作成、さらに可能ならばそのテスティングまでを、限られた時間内で行う必要があった。このため VDM++による記述を設計書の読解と平行して行うことで「モデリングすることで理解する」という方針をとることにした。

また、実験作業は他の業務の合間を縫って行うことになるため、作業期間内にしばしば大きな中断期間が挟まることが当初から予想された。このため、中断期間中の忘却の影響を最小限にすることが必要であった。この対処として、設計書と VDM++記述の対応がなるべく明確になるように VDM++記述を行うという基本方針をとった。これは、もし、設計書と VDM++記述の対応関係が明確でない場合、対応関係を何か別の成果物に記述しておくか、記憶しておく必要が生じる。前者の場合、作業時間をその分だけ増大させることにつなが

るがそれは上述の通り時間制約がある中では困難であり、後者の場合、中断期間中の忘却の可能性があるからである。なお、この方針を採用することで、生成された VDM++記述は、必ずしも VDM++らしい記述法でもなく、十分に抽象化されている形にもならないことが想定されたが、それを犠牲にしても、設計書との対応関係を明確にすることを優先した。これは今回の実験の目的が設計書の指摘事項や問題点の抽出であり、それを達成するためには VDM++記述自体を良くする必然性は高くないと判断したからである。

II. 適用手順のテーラリング

上述の記述方針から、「適用手順」に書かれた手順を、基本的考え方を大きく変えない範囲で、幾つかの作業を平行して行ったり、一部を省略したりする形にテーラリングした。具体的には以下の通りである。

まず作業の平行化について説明する。「モデリングすることで理解する」という基本方針のもと「モデリングする」に対応する作業「形式記述の作成」(図 5-19 の C)を冒頭から開始することとなった。一方「理解する」に対応する作業は適用手順には記述されていなかったが、上述の通り対象設計書は膨大であり理解しやすいものでもなかったため、実際には、この作業に大きな時間をとった。この作業をここでは「既存ドキュメントの調査」(図 5-19 の A)と呼ぶ。「モデリングすることで理解する」という方針であるから、「既存ドキュメントの調査」は「形式記述の作成」と平行して実施した。結局、この 2 つに加えて、本来、手順の冒頭に行われる作業「形式記述作成計画立案」(図 5-19 の B)も合わせた 3 つの作業を平行に実施した。

次に、手順のより詳細な部分における、作業時間の短縮を目的とした中間成果物の省略と、VDM++記述と設計書の対応関係の明確化を目的としたテーラリングについて説明する。まず、「形式記述作成計画立案」の結果の中間成果物は省略し、それにあたる情報を直接 VDM++記述に反映する形をとった。また、識別子の命名は、適用手順では抽出した「主語」「述語」を使うこととなっているが、必ずしもそれには従わず、なるべく設計書内で使われている名称や ID を利用した。VDM++記述の各部には、コメントとして設計書との対応関係や自身が行った判断、推論などを記述した。各処理の記述にあたっては、条件分岐や、エラー発生による処理からの脱出等が、設計書内のアルゴリズム的記述と形式記述とでほぼ 1 対 1 に対応するように記述した。

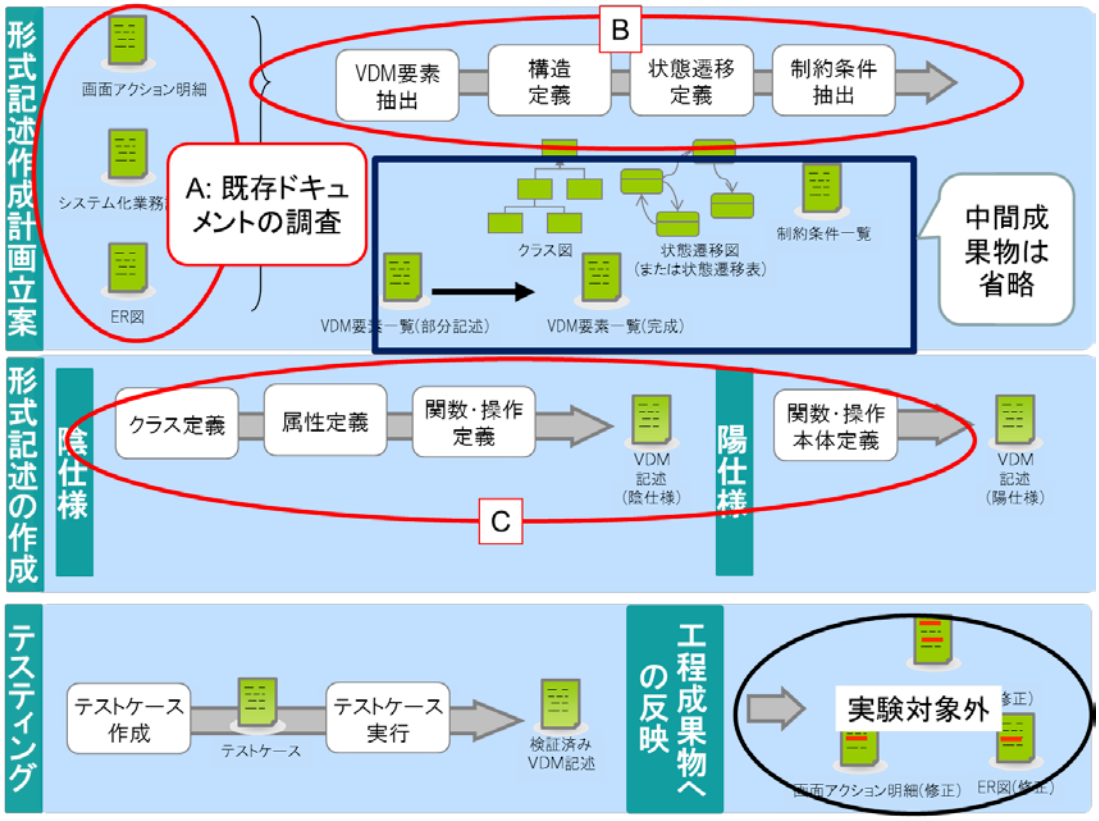


図 5-19 適用手順のテーラリング

III. 中間成果物

上記テーラリングの項で説明したように、適用手順に書かれた中間成果物の多くは省略した。ただし、適用手順に書かれた中間成果物ではないが、「既存ドキュメントの調査」の中間成果物として、全部で 350 個程度ある各設計書ファイルについての情報をまとめた「設計書ファイル一覧表」を作成した。その表の項目はおおよそ図 5-20 の通りである。このうち、わかりにくい項目について、以下で簡単に説明する。

設計書群にはフォルダ構成があったが、「フォルダパス」欄には、各ファイルが存在するフォルダの相対パスを記述した。設計書は、記述方法に何種類かのパターンがあり(これについて主要なものは後述の「作成した形式記述」の節で説明する)、また、直接の設計内容ではなく修正履歴だけが書かれたファイルなども混ざっていた。「内容種別」欄には、これらを区別する情報を記述した。『「機能要件の合意形成ガイド」との対応』欄には、そのファイルが「機能要件の合意形成ガイド」のどの成果物と対応するかを記述したが、多くの場合 1 対 1 に対応するわけではないので、1 つのファイルに複数の対応する「機能要件の合意形成ガイド」成果物があることが通常であった。

この中間成果物を作成したことで、設計書の全体像の把握がより深まり、また作業の長期中断があった後でも、作業再開が容易になった。

項番	フォルダパス	ファイル名	ページ数	内容種別	概要	「機能要件の合意形成ガイド」との対応	備考
17	XXX/YYY	変更履歴.pdf	1	修正履歴			
18	XXX/YYY	AAA.pdf	17	メニュー処理概要	「AAA」メニュー		
19	XXX/YYY	BBB.pdf	2	ユースケース一覧		システム化業務一覧	
20	XXX/YYY	CCC.pdf	336	画面セット		画面一覧、画面遷移、画面レイアウト、画面入出力項目一覧、画面アクション明細、システム化業務説明	
21	XXX/YYY	変更履歴.pdf	2	修正履歴			
22	XXX/YYY	DDD.pdf	5	メニュー処理概要	「DDD」メニュー		
23	XXX/YYY	EEE.pdf	1	ユースケース一覧 + 補足情報		システム化業務一覧	
24	XXX/YYY	FFF.pdf	383	画面セット		画面一覧、画面遷移、画面レイアウト、画面入出力項目一覧、画面アクション明細、システム化業務説明	
25	XXX/YYY	GGG.pdf	1	ユースケース一覧	「GGG」メニューのユースケース一覧	システム化業務一覧	
26	XXX/YYY	HHH.pdf	624	画面セット		画面一覧、画面遷移、画面レイアウト、画面入出力項目一覧、画面アクション明細、システム化業務説明	
27	XXX/YYY	HHH_別紙1.pdf	1	入出力ファイルセット	入出力ファイルのフォーマット定義		「データファイルセット」と若干異なる形式で記述されている。

図 5-20 「設計書ファイル一覧表」イメージ図

IV. 作成した形式記述

以下で、作成した形式記述について代表的な部分を抜粋して説明する。

(a) エンティティ

各エンティティは、5.3.3 節のフレームワークを使用し、テーブル設計書に記述されていた各テーブルをほぼそのまま VDM++で記述した。

図 5-21 は対象システムにおける、「コーポレートアクション検索テーブル」の設計書内の記述のおおよそのイメージである。このテーブルに対応する VDM++による記述が図 5-22 である。

このように、1 つのテーブルに対して 1 つのクラスを用意し、テーブルの各カラム(属性)はそのクラスのインスタンス変数として記述した。

なお、実際にはほとんどの部分を、5.3.3 節のフレームワーク用のジェネレータ(当作業者が若干カスタマイズしたもの)を使って自動生成した。

エンティティ名	コーポレートアクションテーブル						
テーブル名	…						
…							
No	属性名	カラム名	カラムデータタイプ	…			
1	書類 ID	…	…				
2	版数	…	…				
3	書類種別 ID	…	…				
4	書類種別分類区分	…	…				
5	公開日時	…	…				
6	掲載期限有無	…	…				
7	公開状態	…	…				
8	削除状態	…	…				
9	….						

図 5-21 テーブル設計書における「コーポレートアクション検索テーブル」の記述イメージ

```
class コーポレートアクション検索テーブル is subclass of LinkableDocumentEntity

instance variables
  public i 書類 ID:書類 ID;
  public i 版数:版数;
  public i 書類種別 ID:書類種別 ID;
  public i 書類種別分類区分:書類種別分類区分;
  public i 公開日時:公開日時;
  public i 掲載期限有無:[掲載期限有無];
  public i 掲載期限:[掲載期限];
  public i 公開状態:公開状態;
  public i 削除状態:削除状態;
  public i 登録者部署名:登録者部署名;
  public i 登録者メールアドレス:登録者メールアドレス;
  public i 公開種別:公開種別;
  public i 送信メール ID:[送信メール ID];
    …
```

クラス名は、テーブル名をそのまま利用

テーブルのカラムはそれぞれインスタンス変数に。それらの型は別ファイルにまとめて定義し(後述)、それを継承。

```
  …
  public i関連書類ID01:[関連書類ID01];
  public i関連書類ID02:[関連書類ID02];
  public i関連書類ID03:[関連書類ID03];
  public i関連書類ID04:[関連書類ID04];
  public i関連書類ID05:[関連書類ID05];
  public i備考:[備考];
  -- File 111 の p.35 の入力チェック等から，削除状態と公開状態について
  -- 以下の関係があると推定される．
  inv i公開状態 = <未公開> => i削除状態 in set {<なし>, <変更中>};
  inv i公開状態 = <期限切れ> => i削除状態 in set {<なし>, <変更あり>};
  inv i公開状態 = <非公開> => i削除状態 = <削除済>;

operations
  /*
   * constructor
   */
  public コーポレートアクション検索テーブル : 書類ID
    *版数
    *書類種別ID
    * …
    *[備考]
    *エンティティ登録情報
    ==> コーポレートアクション検索テーブル
  コーポレートアクション検索テーブル(a_書類ID,
    a_版数,
    a_書類種別ID,
    …
    a_備考,
    a_エンティティ登録情報
    ) == (
    atomic(
    i書類ID := a_書類ID;
    i版数 := a_版数;
    i書類種別ID := a_書類種別ID;
    …
```

コーポレートアクション検索テーブルは関連書類情報を持つので、それも設計書の記述方法に合わせてVDM++でも記述。

設計書との対応関係を随時コメントとして記述。

設計書から確認または推定される、カラムに関する不変条件がある場合は、それを記述。

各カラムの情報をすべて与えて新しいレコード(インスタンス)を作る構成子を各エンティティで定義。

```
    i備考 := a_備考;
    );
    initialize(a_エンティティ登録情報);
);

/* for copy */
public コーポレートアクション検索テーブル :
  コーポレートアクション検索テーブル
  ==> コーポレートアクション検索テーブル
コーポレートアクション検索テーブル(original) == (
  atomic(
  i書類ID := original.i書類ID;
  i版数 := original.i版数;
  …
  i備考 := original.i備考;
  ); );

public copy: () ==> コーポレートアクション検索テーブル
copy() == return new コーポレートアクション検索テーブル(self);

protected keyEquals: Record ==> bool
keyEquals(r) ==
  let
    rr : コーポレートアクション検索テーブル = VDMUtil`cast[Record, コーポレートアクション検索テーブル](r)
  in
    return
        i書類ID = rr.i書類ID
        and i版数 = rr.i版数
        and i書類種別ID = rr.i書類種別ID
;

public get関連書類情報セット: () ==> set of (関連書類ID * 関連書類種別)
get関連書類情報セット() == (
  dcl 結果 : set of (関連書類ID * 関連書類種別) := {};
```

便宜上、レコード(インスタンス)のコピーを作るための構成子も各エンティティで定義。

レコードの等しさを、テーブル設計書に定義されたキー情報をもとに定義し、それを使ってテーブルに登録することで、テーブル内のキーの一意性を保っている。

処理の中で、レコード情報を別の形でまとめて取り出すようなことがあるときに、場合によってはエンティティに対応する操作/関数を定義している。

```
    if (i 関連書類 ID01 <> nil)
    then 結果 := 結果 union {mk_(i 関連書類 ID01,i 関連書類種別 01)};
    …
    if (i 関連書類 ID05 <> nil)
    then 結果 := 結果 union {mk_(i 関連書類 ID05,i 関連書類種別 05)};
    return 結果;
  );

  -- stub
  public get タイトル : () ==> 書類タイトル
  get タイトル() == return mk_(<書類タイトル>, mk_token("コーポレートアクション"));

  public get 版数 : () ==> 版数
  get 版数() == return i 版数;

end コーポレートアクション検索テーブル
```

今回対象とした範囲では決まっている値などは、スタブ的な形で定義している場合もある。

図 5-22 「コーポレートアクション検索テーブル」の VDM++による記述(抜粋)

(b) 画面

画面も基本的には 1 画面に対して 1 クラスの形で記述した。画面に対応するクラスのクラス名は、設計書で定義されている画面 ID を使った。

対象設計書内の画面設計部分は、基本的には表 5-3 の要素から成っていた。

表 5-3 設計書内の画面設計の記述項目

項番	項目名	概要
1	画面イメージ	表示される画面のイメージ図
2	アクション概要	ボタンを押下した場合等にどのアクションが実行されるかをまとめたもの
3	アクセス制御	画面でボタン押下等ができる権限の制御の説明
4	項目定義	画面に表示される各項目の定義
5	入力可能項目定義	項目定義されたもののうち、ユーザが画面入力できる項目
6	チェック仕様	入力可能項目に入力された内容について、表面的に(DB アクセス等をすることなく)行うチェックの仕様(全角カナであること、など)

このうち、基本的には「項目定義」の情報をもとに、画面に対応するクラスを作成した。基本的には画面の入力/出力項目が、VDM++クラスのインスタンス変数と対応するように記述している。なお、各項目は、実際には画面では文字列として入出力されるが、文字列とそれに対応する実際の値(例えば数値、日付、何らかの区分など)との間の変換は捨象して、画面の段階で、実際の値が入出力される形で記述している。例えば、「金額」を入力項目として持つ画面がある場合、実際のシステムでは、金額を表す 10 進数の数字列を画面に入力し、それがどこかの段階で数値に変換されて処理されるが、VDM++の記述においては、画面の入力項目「金額」の型を最初から数値型としている。

「コーポレートアクション情報」の「変更確認画面」の設計書内における「項目定義」はおおよそ図 5-23 のように記述されている。それに対応する VDM++のクラス記述が図 5-24 である。なお、この画面の場合、たまたま他の画面と共通部分が多かったため、共通部分を親クラスに定義し、それを継承する形をとっている。この画面の親クラスの VDM++記述が図 5-25 である。

画面 ID	GTRSSCA0402	画面名	コーポレートアクション情報変更確認画面		
画面項目定義					
No	項目名	属性	I/O	初期値(デフォルト値)	説明
1	…	…	O		
2	…	…	O		
3	銘柄名	…	O		
4	銘柄コード	…	O		
5	国名	…	O		
6	….				

図 5-23 設計書における「コーポレートアクション情報変更確認画面」の「項目定義」記述イメージ

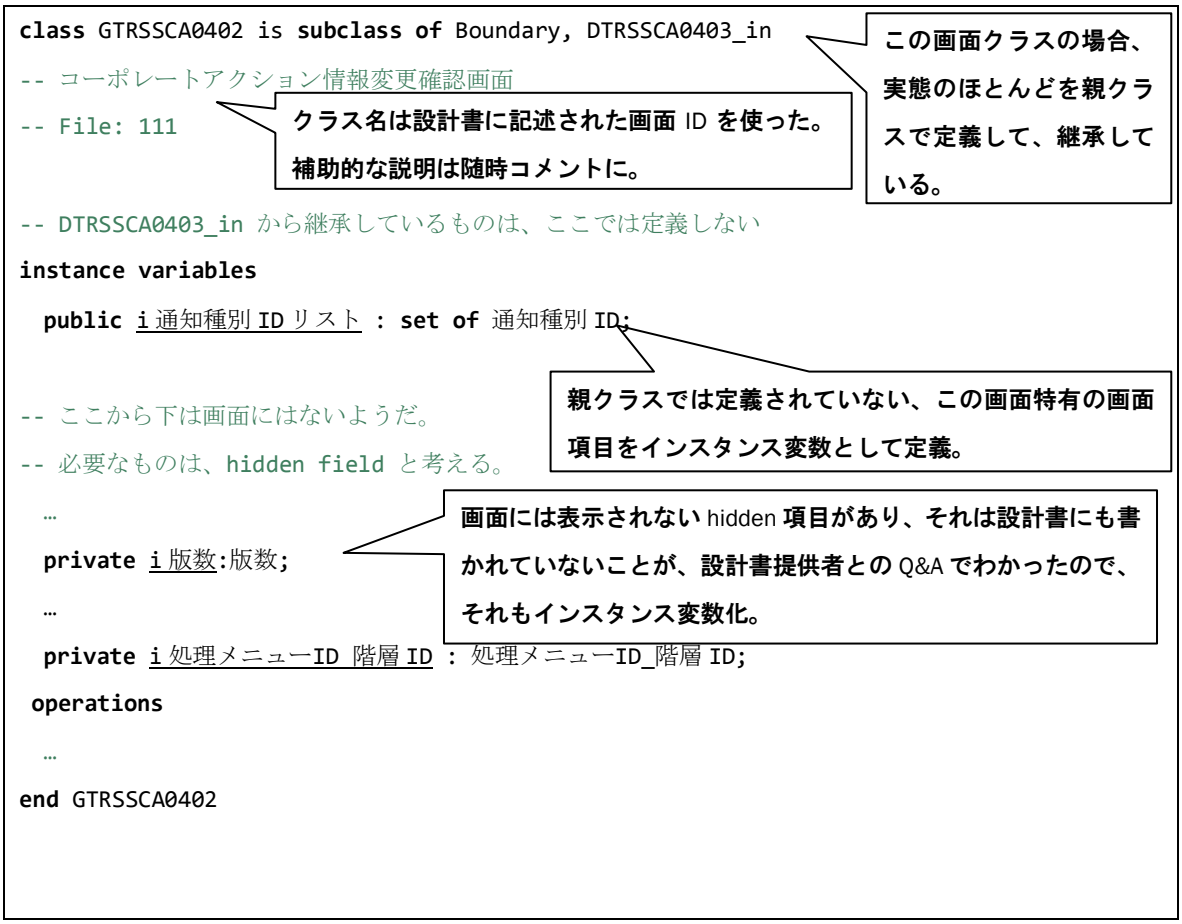


図 5-24 「コーポレートアクション情報」変更確認画面の VDM++記述(抜粋)

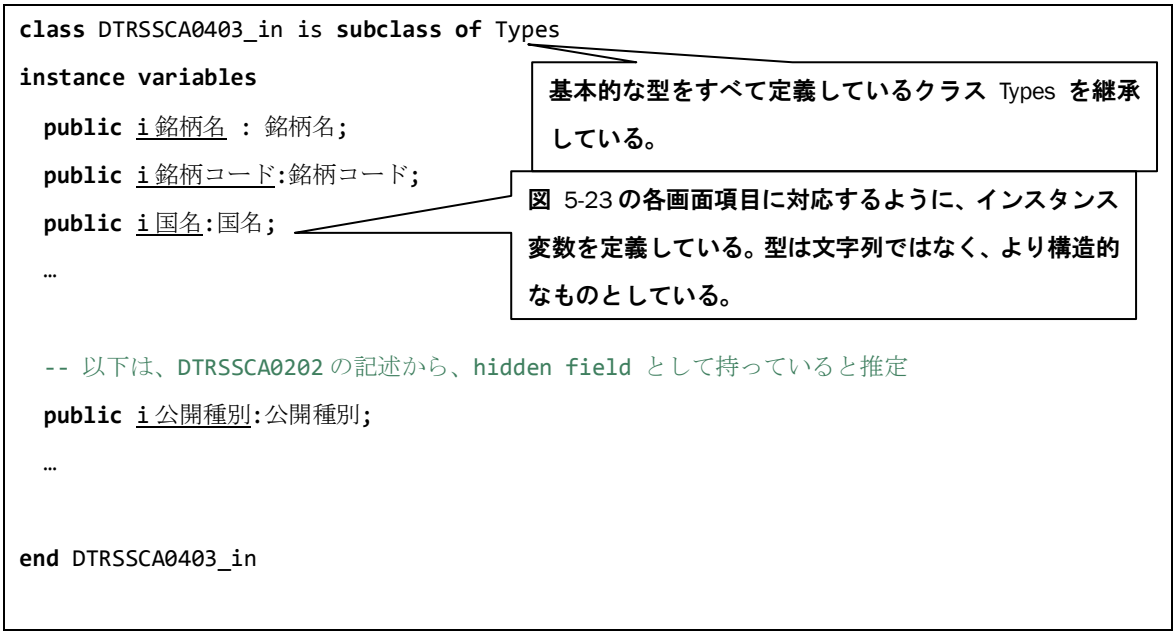


図 5-25 「コーポレートアクション情報」変更確認画面の親クラス(抜粋)

(c) 処理

対象設計書内では、処理には大きく表 5-4 の種類があった。

表 5-4 設計書に含まれる処理の種類

項番	処理の種類
1	画面のボタン押下などで始まる画面遷移処理
2	画面遷移処理の中で呼ばれる、主として DB アクセスに関する下位処理
3	いろいろな場所から呼ばれる共通処理
4	共通処理の下位処理
5	その他の処理(今回の記述対象外。バッチ処理など)

表 5-4 の処理の種類によって、設計書内での処理記述方法が異なっていた。ここでは、まず、表 5-4 項番 1 の画面遷移処理の記述例を説明する。

対象設計書においては、画面遷移処理は、出発点となる画面設計に付随する形で記述され、表 5-5 の項目より成っていた。

表 5-5 設計書における画面遷移処理の記述項目

項番	記述内容
1	アクション ID
2	アクション名称
3	入力 ： 開始画面の情報のうち処理内で使うもの
4	出力 ： 正常終了時の最終画面に出す情報のうち主要なもの
5	備考：補助的な説明
6	(変則)フローチャート
7	アルゴリズムの自然言語記述
8	例外： 処理中で起きうるエラーとその場合の最終画面(エラー画面など)の説明

このうち、主として、表 5-5 項番 7 のアルゴリズムの自然言語記述をもとに VDM++記述を行い、必要に応じて項番 6 の(変則)フローチャートで情報を補う、という方法をとった。

「コーポレートアクション情報」の「変更処理」(「変更確認画面」から「変更完了画面」への画面遷移処理)の、アルゴリズムの自然言語記述部分が、図 5-26、図 5-27、および図 5-28 である。

「アルゴリズムの自然言語記述」は基本的に「検証」部と「正常」部から成る。「検証」部は、入力値が、データベースに格納された情報などと照らし合わせたときに妥当な値であることを検証する部分である。「正常」部は、検証済みの入力をもとに、実際の処理をす

る部分である。この「変更処理」の場合、図 5-26 が「検証」部の記述であり、図 5-27、図 5-28 が「正常」部の記述である(図 5-26 内のふきだしについては、後の「抽出した問題点」で説明する)。図をみてもらえばわかるように、「正常」部の処理の中でもエラーが起こるときもあり、その時にどうするかもそこに自然言語で記述されている。

なお、これらの図において、「アクション詳細 ID」欄に記述されているのは、その処理をするために呼ばれる下位処理や共通処理である。記号列(例えば DTRSSCA0104)はその下位処理(あるいは共通処理)の処理 ID である。

パターン	No	アクション詳細ID	枝番	処理
検証	1			
	2	(DTRSSCA0104)	2-1	入力値(書類種別ID、書類ID、版数)を条件に、コーポレートアクション検索テーブルより書類情報を取得
	3	-	3-1	検証2で取得したレコードが存在しない場合
			3-2	エラーメッセージ(MSGTRBCBC40940:"指定された[書類]が見つかりません。")をセット
			3-3	エラー画面へ遷移し、終了
	4	共通(FTRBCBC1901)	4-1	検証2で取得した書類の公開状態/削除状態から、変更可否を判定(業務共通)
				入力値(公開状態、削除状態、権限、マトリックス名称)
	5	-	5-1	検証4の結果、NGの場合
			5-2	エラーメッセージ(MSGTRBCBC40951:"アクセス権がない為、[変更]できません。")をセット
			5-3	エラー画面へ遷移し、終了
	6	(DTRSSCA0105)	6-1	入力値(書類ID)を条件に、コーポレートアクション通知種別テーブルより通知種別IDを取得
	7	-	7-1	検証6の結果
			7-2	該当レコードなしの場合、
			7-3	エラーメッセージ(MSGTRBCBC40940:指定された[書類]が見つかりません。)をセット
			7-4	エラー画面へ遷移し、終了
	8	共通(ITRUMOF0902)	8-1	検証6で取得したすべての通知種別IDに対し、登録権があるか確認
			8-2	・すべての通知種別IDに対し、登録権がある場合、判定結果を"OK"とする
			8-3	・すべての通知種別IDに対し、一つでも登録権がない場合、判定結果を"NG"とする
	9	-	9-1	検証8の結果、NGの場合
			9-2	エラーメッセージ(MSGTRBCBC40951:アクセス権がない為、[変更]できません。)をセット
			9-3	エラー画面へ遷移し、終了
	10	-	10-1	公開状態チェック
			10-2	取得した書類の公開状態を判定
	11	-	11-1	検証10の結果、公開状態が"公開"の場合
			11-2	版数判定を実施
			11-3	当該書類の版数を判定。
			11-4	版数が99版未満のとき、判定結果OK
			11-5	版数が99版以上のとき、判定結果NG
	12	-	12-1	検証11の結果、NGの場合
			12-2	エラーメッセージ(MSGTRBCBC40931:"指定された書類は変更できません。")をセット
			12-3	エラー画面へ遷移し、終了
	13	-	13-1	入力チェック
	14	-	14-1	検証13の結果がNGの場合
			14-2	エラーメッセージ("～を入力してください"等)をセット
			14-3	変更画面へ遷移し、終了

ここの記述が「NG の場合」とされている問題については後述

図 5-26 設計書における「コーポレートアクション情報」変更の処理記述(入力値検証部分)

パターン	No	アクション詳細ID	枝番	処理
正常	1	-	1-1	排他チェック
	2	-	2-1	正常1の結果、NGとなった場合
			2-2	エラーメッセージ(MSGTRBCBC40801:"他ユーザが更新した為、[変更]ができません。")をセット
			2-3	エラー画面へ遷移し、終了
	3	共通(FTRBCBC1102)	3-1	関連書類情報を更新(業務共通)
	4	-	4-1	正常3の結果がNGの場合
			4-2	関連書類がすでに10件の書類に関連付けられている場合
			4-3	エラーメッセージ(MSGTRBCBC40901:"指定した書類({0})は、すでに10件の書類より関連付けられています。")をセット
			4-4	関連書類が見つからない場合
			4-5	エラーメッセージ(MSGTRBCBC40902:"関連書類({0})が見つかりません。")をセット　※{0}には書類IDをセット
			4-6	コーポレートアクション情報変更画面へ遷移し、終了
	5		5-1	公開状態チェック
	6		6-1	当該書類の公開状態が"未公開"の場合
			6-2	当該書類のメール送信IDがある場合、
		共通(FTRBCBC1602)	6-3	業務共通　メール送信停止 入力値(メール送信ID)
			6-4	当該メール送信IDに対する送信待機中メールの削除を実施(業務共通)
	7	-	7-1	正常6の結果がNGの場合
			7-2	エラーメッセージ(MSGTRBCBC40802:"[変更]に失敗しました。")をセット
			7-3	エラー画面へ遷移し、終了
	8		8-1	(入力値)公開日時が"即時"の場合
		DTRSSCA0401	8-2	入力値(書類種別ID、書類ID、版数)を条件に、コーポレートアクション検索テーブルを更新
			8-3	※ステータスの更新:　公開/削除状態が"未公開/なし"、または、"未公開/変更中"を"公開/なし"に更新
	9	-	9-1	正常8の結果、処理件数が0件の場合、
			9-2	エラーメッセージ(MSGTRBCBC40802:"[変更]に失敗しました。")をセット
			9-3	エラー画面へ遷移し、終了
	10	-	10-1	削除状態チェック
			10-2	削除状態が"変更中"ではない場合
			10-3	正常20の処理を継続する
	11		11-1	削除状態が"変更中"の場合
		DTRSSCA0402	11-2	入力値(書類種別ID、書類ID)を条件に、コーポレートアクション検索テーブルを更新
			11-3	※ステータスの更新:　公開/削除状態が"公開/変更中"を"公開/変更有"に更新
	12	-	12-1	正常11の結果、処理件数が0件の場合
			12-2	エラーメッセージ(MSGTRBCBC40802:"[変更]に失敗しました。")をセット
			12-3	エラー画面へ遷移し、終了

図 5-27 設計書における「コーポレートアクション情報」変更の処理記述(正常系処理冒頭)

	26	-	26-1	正常25の結果がNGの場合
			26-2	エラーメッセージ(MSGTRBCBC40802:"[変更]に失敗しました。")をセット
			26-3	エラー画面へ遷移し、終了
	27	(DTRSSCA0205)	27-1	入力値(書類種別ID、書類ID)を条件に、コーポレートアクション検索テーブルを更新
			27-2	メール送信IDを当該書類IDに対し、更新する(全ての版数に対して、実施)
	28	-	28-1	正常27の結果がNGの場合
			28-2	エラーメッセージ(MSGTRBCBC40802:"[変更]に失敗しました。")をセット
			28-3	エラー画面へ遷移し、終了
	29		29-1	表示制御
		共通(FTRBCBC0501)	29-2	入力値(メニューID、画面ID)を条件に画面タイトルバーに画面タイトルをセット(業務共通)
			29-3	完了メッセージをセット
パターン		現象		処理
例外		DB接続エラー リモート接続エラー		エラー画面へ遷移 エラー画面へ遷移
備考				

図 5-28 設計書における「コーポレートアクション情報」変更の処理記述(正常系最終部)

これを VDM++で記述するにあたって、処理内容が非常に長いので、扱いやすくするため「検証」部分と「正常」部分に分割して記述することにした。図 5-30 が「検証」部分の記述、図 5-31 が「正常」部分の記述、その 2 つを組み合わせた全体が図 5-29 である。

```
class PageTransition is subclass of DetailedAction

instance variables
  public errorMessageList : seq of (seq of char) := [];

operations
…
  /*
   * コーポレートアクション情報変更アクション
   * もとの画面に戻るのは入力チェックエラーのときだけ。
   * File: 111
   */
  public ATRSSCA0402: GTRSSCA0402 ==> GTRSSCA0403 | GTRBCBC9999 | GTRSSCA0402
  ATRSSCA0402(入力画面) == (
    -- 機能 ID は呼ばれた操作と結びついていると仮定する。
    i機能 ID := <外国株券・コーポレートアクション>;
    def db1 = 外国株サイト DAO`getSiteDB();
        -- dbu = ユーザ管理サイト DAO`getSiteDB();
        -- dbc = SiteDAO`getSiteDB();
    in (
      db1.setTransaction();
      -- dbu.setTransaction();
      -- dbc.setTransaction();
      def 検証結果 = ATRSSCA0402_検(入力画面)
      in
        if (isofclass(GTRBCBC9999,検証結果) or
            isofclass(GTRSSCA0402,検証結果))
        then (
          def - = db1.rollback();
              -- - = dbu.rollback();
              -- - = dbu.rollback();
          in skip;
          return 検証結果;
        )
        else (
          def res = ATRSSCA0402_正(入力画面, 検証結果);
```

画面遷移処理は、遷移前画面を入力とし、遷移後画面を出力とする操作として記述。
この処理の場合、
入力：「変更確認画面」、出力：「変更完了画面」「エラー画面」「変更確認画面」(入力次第では元の変更確認画面に戻る場合がある)の 3 種。
それぞれが画面 ID で記述されている。

設計書の記述方法に合わせて、入力値検証処理と、検証後の処理に分けて記述した。

正常系以外の画面の場合…

ロールバック(DB フレームワーク利用)。

正常系画面の場合…

```
            - = db1.commit();
            -- - = dbu.commit();
            -- - = dbc.commit();
          in return res;
        );
    );
  )
  -- pre -- 変更できるのは、最新版のみという Q&A 回答から。
  --   let 版_set =
  --         { r.i 版数 | r in set i コーポレートアクション検索テーブル DAO.getAll() &
  --                   r.i 書類 ID = 入力画面.i 書類 ID }
  --   in max(版_set) = 入力画面.get 版数()
  post
    isofclass(GTRSSCA0403,RESULT) =>
    let - = new IO().echo(VDMUtil`val2seq_of_char[set of コーポレートアクション検索テーブル](i コーポレートアクション検索テーブル DAO.getAll()))
    in
    (exists1 r in set i コーポレートアクション検索テーブル DAO.getAll() &
      r.i 書類 ID = 入力画面.i 書類 ID and
      r.i 版数 = 入力画面.get 版数()) and
    (card({ r | r in set i コーポレートアクション検索テーブル DAO.getAll() &
      r.i 書類 ID = 入力画面.i 書類 ID and
      r.i 版数 = 入力画面.get 版数()+1 }) <= 1);
```

コミット(DB フレームワーク利用)。

事後条件は、記述可能な範囲で記述(ここで記述されているのは本来必要なものより少し弱い条件)。

図 5-29 「コーポレートアクション情報」の「変更処理」の VDM++記述(抜粋)

```
/*
 * ATRSSCA0402 の 「検」部分のみ
 */
public ATRSSCA0402_検: GTRSSCA0402
==> コーポレートアクション検索テーブル | GTRBCBC9999 | GTRSSCA0402
ATRSSCA0402_検(入力画面) == (
  -- 検 2
  def 書類情報_set
        = DTRSSCA0104(入力画面.i 書類 ID,
                      get 書類種別 ID(入力画面.i 書類 ID),
                      nil, 入力画面.get 版数());
  in (
    if (書類情報_set = {})
    then -- 検 3
       return ATRBCBC9901(入力画面, "指定された書類が見つかりません。");
                                  -- MSGTRBCBC40940
    -- 検 4
    def mk_(書類情報, -) = iota - in set 書類情報_set & true;
        変更可否判定結果 = FTRBCBC1901(書類情報.i 公開状態,書類情報.i 削除状態,
                                       <登録>, <変更マトリックス>);
                                       -- 権限とマトリックス名称は
                                       -- よくわからないので仮に置いたもの。
                                       -- ***stub***
    in (
      if (変更可否判定結果 = <パラメータエラー>) then
        return ATRBCBC9901(入力画面,
                           "パラメータエラー。"); -- このように内部処理でエラーが起った場合の
全体としての処理は
                                                 -- 設計書には明言されていないが、仮設定。

      if (isNG(変更可否判定結果))
      then -- 検 5-2
        return ATRBCBC9901(入力画面,
                           "アクセス権がない為、変更できません。"^"(1)");
                           -- MSGTRBCBC40951
      -- 検 6
```

対応する設計書(図 5-26)に合うように条件分岐等を順に記述。

このように設計書で明確でない部分を推測で補っている部分もある。

```
      …
            -- 検 13
            if (not ATRSSCA0402_入力チェック(入力画面))
            then
              return 入力画面;
          );
        ); -- 検 11 then 終わり
        return 書類情報;
      ); -- 検 8 in 終わり
    ); -- 検 6 in 終わり
  ); -- 検 4 in 終わり
); -- 検 2 in 終わり
) -- ATRSSCA0402_検 定義終わり
pre i機能 ID <> nil;
```

推定される事前条件。

図 5-30 「コーポレートアクション情報」の「変更処理」の「検証」部分の VDM++記述（抜粋）

```
/*
   * ATRSSCA0402 の「正」部分のみ
   */
  public ATRSSCA0402_正: GTRSSCA0402 * コーポレートアクション検索テーブル
  ==> GTRSSCA0403 | GTRBCBC9999 | GTRSSCA0402
  ATRSSCA0402_正(入力画面, 書類情報) == (
    -- 正 1
    if (not ATRSSCA0402_排他チェック())
    then
      return ATRBCBC9901(入力画面,
                         "他ユーザが更新した為、変更ができません。");
                         -- MSGTRBCBC40801
    -- 正 3
    def 関連リンク登録結果
        = FTRBCBC1102(入力画面.i 書類 ID, 入力画面.get 関連書類リスト());
    in (
      if (関連リンク登録結果 <> <OK>)  -- 4-1
      then (
        if (関連リンク登録結果 = <登録数上限エラー>)
        then
          return ATRBCBC9901(入力画面,
                 "指定した書類は、すでに 10 件の書類より関連付けられています。")
                 -- MSGTRBCBC40901
        …
        -- 正 29
        FTRBCBC0501(nil, "GTRSSCA0403");
        return new GTRSSCA0403();
      ); -- def w 処理メニュー階層 終わり
    ); -- 正 3 def 関連リンク登録結果終わり
  ); -- 検 11-3 終わり
```

やはり対応する設計書(図 5-27)に合うように条件分岐等を順に記述。

図 5-31 「コーポレートアクション情報」の「変更処理」の「正常」部分の VDM++記述(抜粋)

以上、表 5-4 項番 1 の画面遷移処理の VDM++記述を例で説明した。

次に、表 5-4 項番 2 の DB アクセスに関係する下位処理の記述を説明する。DB アクセスに関する下位処理は、設計書においては、おおよそ図 5-32 のように記述されている。

アクション詳細定義		
ID	DTRSSCA0402	
テーブル名	コーポレートアクション検索テーブル	
カラム名	セット項目	編集仕様
公開状態	“2”(公開)	
削除状態	“3”(変更有)	
条件	1. 書類種別 ID = (入力値).書類種別 ID And 2. 書類 ID = (入力値).書類 ID And 3. 公開状態 = “公開” And 4. 削除状態 = “変更中”	
備考		

図 5-32 設計書におけるコーポレートアクション検索テーブルの「公開状態」と「削除状態」の更新処理の記述イメージ

図 5-33 は、CRUD でいう U(update)の処理である。「条件」部分に記述された条件を満たすレコードを対象に、その「カラム名」に記述されたカラムに「セット項目」に記述された値をセットする。

これに対応する VDM++記述が図 5-33 である。

```
/*
 * コーポレートアクション検索テーブルの「即時」更新。
 * 指定された書類種別 ID, 書類 ID を持ち、状態が 公開/変更中 である書類の
 * 状態を 公開/変更有 に更新する。
 * File: 111
 */
public DTRSSCA0402: 書類種別 ID * 書類 ID ==> ()
DTRSSCA0402(書類種別 ID, 書類 ID) == (
  def コーポレートアクション検索テーブル_set =
    { T1 | T1 in set i コーポレートアクション検索テーブル DAO.getAllRecord() &
           T1.i 書類種別 ID=書類種別 ID and
           T1.i 書類 ID=書類 ID and
           T1.i 公開状態 = <公開> and
           T1.i 削除状態 = <変更中>}
  in (
    for all w コーポレートアクション検索テーブル in
            set コーポレートアクション検索テーブル_set do (
      dcl T1: コーポレートアクション検索テーブル
            := w コーポレートアクション検索テーブル.copy();
      atomic(
        T1.i 公開状態 := <公開>; -- 2 -- この処理は冗長
        T1.i 削除状態 := <変更あり>; -- 3
      );
      i コーポレートアクション検索テーブル DAO.update(T1);
    );
  );
);
```

設計書の「条件」をほぼそのまま内包の条件に記述。

不変条件を崩さないため atomic 内で値を同時設定。

更新処理なので update (DB フレームワーク利用)。

図 5-33 コーポレートアクションの状態更新の VDM++記述

上記は DB の更新の例であったが、CRUD の他の項目、すなわち C(生成)、R(検索、抽出)、D(削除)についてもほぼ同様の記法で設計書内に記述されており、VDM++記述もそれに対応した記述をしている。

共通処理については、設計書内でいろいろな記述方法がとられている。このため VDM++は、画面遷移処理と類似のアルゴリズム記述がある場合はそれをもとに記述し、DB アクセス記述がある場合にはそれをもとに記述し、いずれとも違う場合には、設計書内から抽出

可能な情報を組み合わせて分かる範囲で処理を記述した。

(d) コード類、種別等(型)

いろいろな場所で使われる型を Types という一つのクラスの中で定義し、それを他のすべてのクラスが継承するようにすることで、抽象度の高い形で VDM++記述を行い、またその抽象度を途中で自由に変更することを容易にした。

その中で、コード類や、各種の種別は、VDM++の引用型とその合併型の組み合わせとして記述することで、高い抽象度と可読性を保つことを行った。

図 5-34 が Types の抜粋である。

```
/*
 * 全クラスで使用する型を定義するクラス
 */

class Types

types


  public 文字列 = seq of char;
  public 日時 = <日時> * token;
  public 日付 = <日付> * token;

-- ユーザ情報はセッションが持っている(File:244)
-- アクション ATRBCHM0101 の入力として登場.
  public ユーザ情報 ::
    i ユーザ ID : ユーザ ID
    …
    i ユーザグループ ID : ユーザグループ ID
    i ユーザグループ名_略称_ : ユーザグループ名_略称_;
  public 最終ログイン日時 = 日時;
  …
  public 公開状態 =
    <未公開>
  | <公開>
  | <期限切れ>
  | <非公開>
  ;
```

特に詳細化する必要がないものは token 型にするという VDM++のイディオムの考え方は守りながら、別種のものを混同して使った場合には型チェックでエラーになるようにするため、引用型 *token という型を使っている。

設計書内で出てくる構造をもったデータの型は、レコード型で定義。名称はなるべく設計書内のものを使っている。

別種である可能性があるものは、とりあえず別の型名をつけておき、あとで同一だとわかった場合は型の同値宣言をしている。

公開状態は、「未公開」、「公開」、「期限切れ」、「非公開」の 4 種の値を持つことを、引用型の合併型として定義。

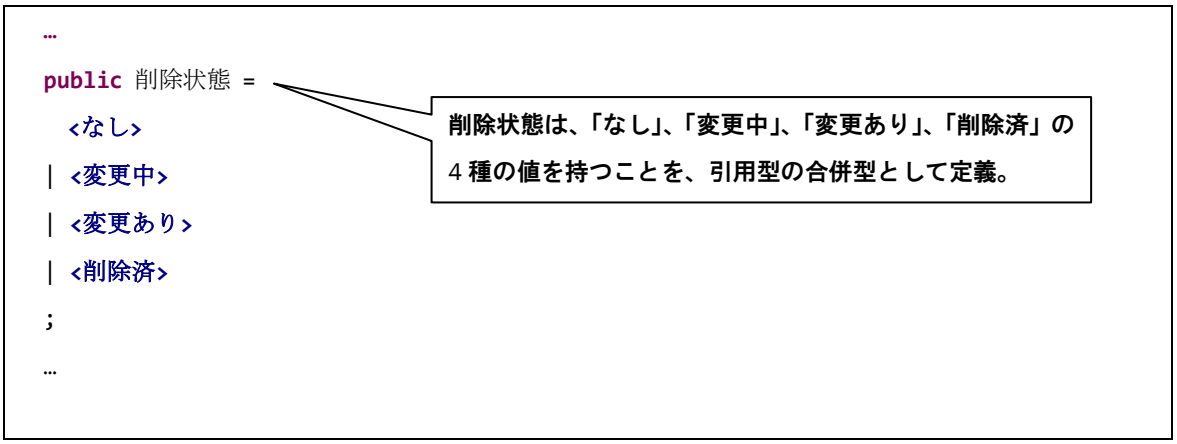


図 5-34 型を定義したクラス(抜粋)

✧ 静的検査

静的検査としては、明示的には型チェックをかける処理を行った。これによって抽出された問題点もあるが、詳細は、後述の「抽出した問題点」の各「抽出方法」参照のこと。

✧ 動的検査

動的検査としては、「変更確認画面」から「変更完了画面」への画面遷移のうち、一番の主要部分のテストをした。結果としては、テストした範囲では設計書記述内容との不整合は抽出できなかった。

なお、不整合ではないが、動的検査の作業中に実感させられた設計書群のわかりにくさの問題について次節「抽出した問題点」の最後に簡単に触れる。

✧ 抽出した問題点

本節では、抽出した設計書の問題点を以下の項目に分けて記述する。

(a) 実際に抽出した内容

(b) その問題点が最終的に完成したシステムにひき起こし得る影響

(c) 抽出方法

✧ アクセス制御処理の設計不足

(a) 抽出した内容：

「コーポレートアクション情報」の「変更処理」の中で呼ばれる書類に対するアクセス制御を行う下位処理(以下「アクセス制御処理」と呼ぶ)の設計内容が、設計されていない。（発注者による評価の結果、不具合ではないことが判明）

(b) システムへの影響：

「コーポレートアクション情報」の「変更処理」において、その書類の「公開状態」お

よび「削除状態」からは本来変更する権利がない利用者が、変更処理を進めてしまう(ことができるような実装がされてしまう)可能性がある。

(c) **抽出方法：**

「アクセス制御処理」を VDM++で記述しようとしたときに、設計書内にその設計がないことに気付いた(その後、設計書提供者との Q&A で、この部分の設計が本来記述されているべきところに書かれていないことが確認された)。

II. 処理記述内の分岐条件の記述の曖昧さ

(a) **抽出した内容：**

「変更処理」内の分岐条件として、下位処理(「アクセス制御処理」)の結果が「NG の場合」に分岐すると記述されていた(図 5-26 のふき出し参照)が、「アクセス制御処理」自体の設計書では「アクセス制御処理」が返す値は「○」「×」「—」の三種類とされており、どれが「NG」でどれが「NG」ではないのかがはっきりしない。

(b) **システムへの影響：**

「アクセス制御処理」が使うアクセス制御用 XML ファイルの作成者がアクセス不可能の意味で「—」を使ってしまった場合、「コーポレートアクション情報」の「変更処理」において、アクセス制御用 XML ファイルの作成者の意図としては変更する権利がないことになるはずの利用者が、「コーポレートアクション情報」の変更処理を進めてしまう(ことができるような実装がされてしまう)可能性がある。

(c) **抽出方法：**

「コーポレートアクション情報変更処理」の処理内容を VDM++で記述していた最中に、「NG の場合」という設計書の表現に遭遇し、具体的に VDM++でどう記述するのが正しいのか判断できなかったことによる。

III. 共通処理「関連書類リンク登録処理」の設計の曖昧さ

(a) **抽出した内容：**

共通処理である「関連書類リンク登録処理」(2 種類あるうちの 1 つ)について、設計書には「他書類からリンク先書類へのリンク数が 10 を越える場合でも、登録数の上限エラーを返さない」としか記述されておらず、10 を越えた場合にリンクが生成されるのか生成されないのかが曖昧 (その後、設計書提供者との Q&A により、10 を越えてもリンクを生成するのが正しい動作と確認された)。

(b) **システムへの影響：**

この共通処理を使って関連書類情報を登録することを含むすべての処理(例えばコーポレ

ートアクション情報の削除処理など)において、他書類からリンクされる数が上限値 10 を越える書類があった場合、その書類についてのリンクが生成されない(ように実装されてしまう)可能性がある。

(c) **抽出方法：**

対象の「関連書類リンク登録処理」の処理内容を VDM++で記述しようとして、どう記述するのが正しいのか判断できなかったことによる。

IV. 共通処理が呼び出す下位処理の指定間違い

(a) **抽出した内容：**

「コーポレートアクション情報」の扱いにも関係する共通処理「ユーザメールアドレス取得処理」等において、 呼び出す下位処理が、本来「ユーザ存在確認処理」(与えられたユーザ ID が既に登録されたユーザのものかを確認する処理)のはずのところ、設計書には「ユーザグループ存在確認処理」(与えられたユーザグループ ID が既に登録されたユーザグループのものかを確認する処理)を呼ぶように記述されている。

(b) **システムへの影響：**

内部で、「ユーザメールアドレス取得処理」等の共通処理を呼ぶすべての処理(例えば「コーポレートアクション情報」の登録処理、変更処理など)において、正当なユーザのユーザ ID を使ったのに、そのユーザ ID と同じユーザーグループ ID を持つユーザグループが存在しなかった場合、処理が進まずエラーとなる。あるいは逆に、指定したユーザＩＤが存在しないユーザのものだったのに、そのユーザ ID とたまたま同じユーザグループ ID を持つユーザグループが存在した場合、エラーにならずに処理が進んでしまう。

(c) **抽出方法：**

VDM++記述で、ユーザ ID とユーザグループ ID を異なる型として記述していたため、VDMTools の「型チェック」の実行により、「操作呼び出しの型が合わない」というエラーが発生し抽出された。

V. DB アクセス処理における指定テーブルカラム名の間違い

(a) **抽出した内容：**

「コーポレートアクション検索テーブル」というテーブルにレコードを「新規登録」をする下位処理において、値を設定するカラムの一つについて、本来「登録者メールアドレス」であるべきところが設計書においては「登録時メール」と記述されている。

(b) **システムへの影響：**

たまたま名称が似ている別のカラムはないため、実装時までに間違いに気付き修正される可能性は高い(実際に当該プロジェクトでは修正された)が、もし修正されない場合には、

このレコードが更新された時など、本来「登録者メールアドレス」にメールが送信されるはずの時に、そのメールが送られない可能性がある。

(c) **抽出方法：**

「コーポレートアクション検索テーブル」と対象処理をそれぞれ VDM++で記述した後、VDMTools で「型チェック」を行わせた時に、「代入しようとしているインスタンス変数が存在しない」という種類のエラーが発生したことから抽出された(テーブルのカラムは VDM++ではインスタンス変数として記述している)。

VI. DB 検索処理における検索条件式の記述間違い

(a) **抽出した内容：**

「コーポレートアクション検索テーブル」および「コーポレートアクション通知種別テーブル」という 2 つのテーブルから条件にマッチするレコードを抽出しその情報を取得する下位処理において、設計書にはレコードを指定する条件式の一部に「削除状態≠"非公開"」と記述されているが、「削除状態」が「非公開」という値をとることはありえない。(その後の設計書提供者との Q&A で、「公開状態≠"非公開"」の間違いであることが確認された。)

(b) **システムへの影響：**

コーポレートアクション情報の強制削除処理の結果として、「公開状態」が「非公開」の書類が削除されてしまう(ように実装されてしまう)可能性がある。

(c) **抽出方法：**

「コーポレートアクション検索テーブル」と対象処理(検索条件を含む)をそれぞれ VDM++で記述した後、VDMTools で「型チェック」を行わせた時に、「型が違うものを≠で比較している」という種類のエラーが発生したことから抽出された。

VII. DB 検索処理における検索条件の遺漏

(a) **抽出した内容：**

「コーポレートアクション検索テーブル」および「コーポレートアクション通知種別テーブル」という 2 つのテーブルから、与えられた「書類 ID」の書類の情報を取得する処理において、検索条件に「書類 ID」が含まれていない。

(b) **システムへの影響：**

コーポレートアクション情報の一覧画面で、表示された書類の中に「変更有」のものがあり、その「変更有」リンクを押下した場合、その書類とは無関係の書類の詳細情報が表示される場合がある。

(c) **抽出方法：**

当該処理を VDM++によって記述している最中に、当該処理の本来の目的と処理内容の一

致を確認している過程で抽出した。

VIII. DB の更新処理における、カラム設定の遺漏

(a) 抽出した内容：

特定条件の時に「コーポレートアクション情報」の新しい版を作成し、DB の「コーポレートアクション検索テーブル」に登録する下位処理において、新しく生成されるレコードの「関連書類」、「掲載期限」、「公開日時」、「公開種別」等に値を設定する処理が抜けている。

(b) システムへの影響：

ある条件のときに「コーポレートアクション情報」の新しい版を作成すると、新しい版の「関連書類」、「掲載期限」、「公開日時」、「公開種別」等が、登録しようとしたものとは違う値になってしまうことが発生する。

(c) 抽出方法：

当該処理を VDM++で記述している際に、処理設計書に記述された新しいレコードに値を設定するカラム(総数 36)が DB 定義書に記述されたそのレコードのすべてのカラム(実質的に意味のある数 58)に比べてかなり少ないことに気付いたことで抽出された。

IX. 取得される関連書類情報の不整合

(a) 抽出した内容：

「コーポレートアクション情報」の関連書類を取得するために使われる処理が、対象の「コーポレートアクション情報」の版にかかわらず、常に最新版に対する関連書類の情報を返却するように設計されている。

(b) システムへの影響：

コーポレートアクション情報に複数の版(1 版と 2 版とする)があり、1 版と 2 版の関連書類が異なっていたとする。その書類がある状態のとき古い版(1 版)の詳細画面を表示することができるが、そのとき、関連書類の欄に最新版(2 版)の関連書類が表示されてしまう。

(c) 抽出方法：

「書類 ID」と「版数」を指定しなければ個々の書類を特定できない、という対象システムについての知識を持った状態で、当該処理を記述しようとした際に、「書類 ID」のみが入力になっており、「版数」が指定されていないことに気付いたことから発見された。

なお、上記以外にも、いくつもの設計上の曖昧さや誤字・脱字を抽出したが、些細なも

のであると考えられたため、指摘事項としては報告していない。

ただ最後に、指摘事項ではないが、動的検査の際に作業者が体感した、対象設計書のファイル間の関係のわかりにくさの問題について簡単に説明する。

「変更確認画面」から「変更完了画面」への画面遷移のテストのための作業をしている中で、設計内容(図 5-26、図 5-27、図 5-28 等)が記述されたファイルとは別に、処理の概要を記述したファイルが存在しており、そこに「テスト結果がどうなるべきか」に関して記述されていることが発見された。両者の関係は少なくとも当該作業者にはわかりにくく、またこのファイルの存在に気付かない段階で推定していた「テスト結果がどうなるべきか」は、そのファイルに記述されたものとは異なっていた。言い換えると、設計書ファイル間の関係が明示されていないことにより、正しいものを間違いと判定してしまうこともあり得るということである。これからその逆、すなわち間違っているものを正しいと判定してしまうこともあり得るということは簡単に類推できる。まとめると、対象設計書は設計書ファイル間の関係についての明示化が不十分であり、このことは問題点の温床になりかねないといえる。

5.3.4.4 作業者 D

✧ **目的**

外部設計書のアクション定義に明示されていないフロー（明確な定義が無く、また明確に禁止もされていないフロー）でコーポレートアクション情報検索を実行した場合について、制約条件違反（事前条件違反、事後条件違反、不変条件違反）を抽出する。

✧ **記述対象**

コーポレートアクション情報検索（以下、CA 情報検索）を記述対象とする。

外部設計書に記述された CA 情報検索のアクション定義は、図 5-35、図 5-36 の通り。

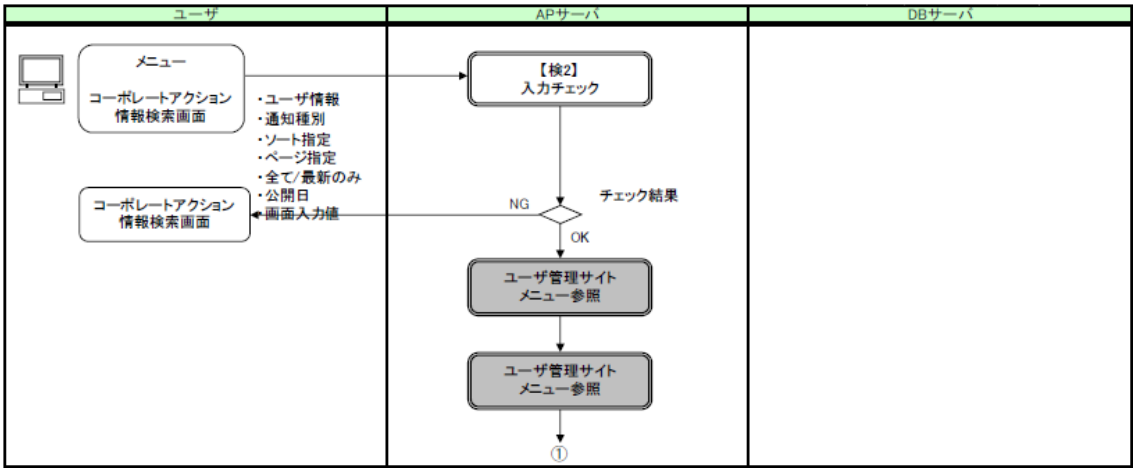


図 5-35　CA 情報検索のアクション定義（前半）

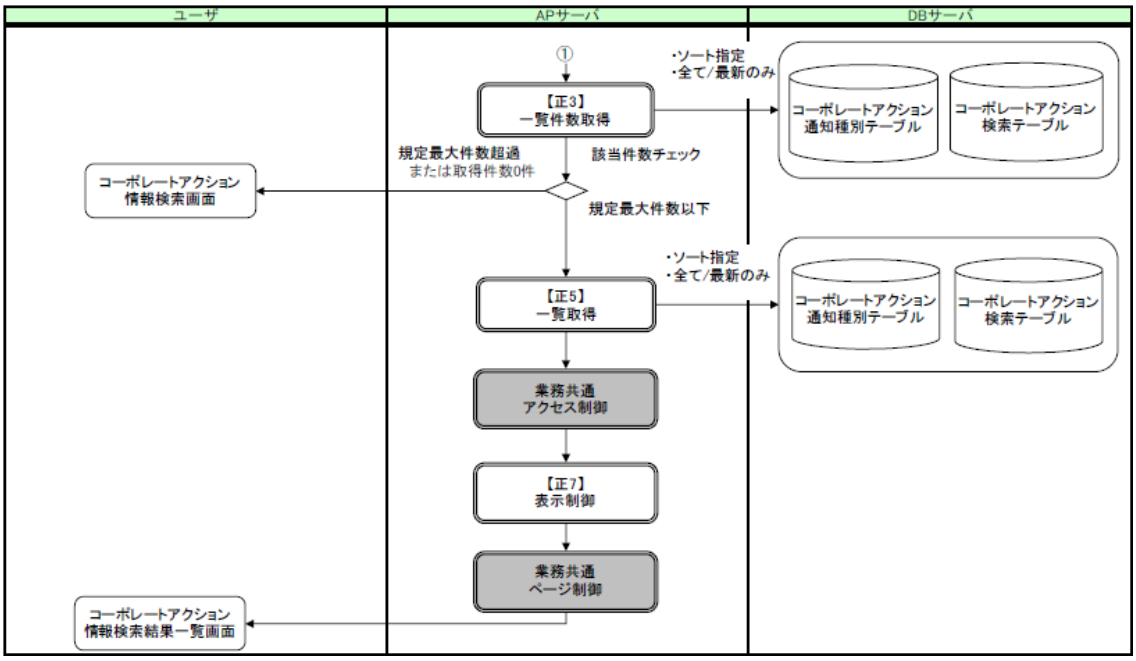


図 5-36　CA 情報検索のアクション定義（後半）

CA 情報検索の制約条件違反の抽出を目的とする為、CA 情報検索のアクション定義とその事前条件、事後条件を VDM++で記述する。

✧ 形式記述の作成

I. 記述方針

記述方針として、以下の 2 つを設定した。

- CA 情報検索のアクション定義で記述されたノード（図 5-35 の【検 2】、図 5-36 の【正 3】など）ごとに、VDM++の操作を記述する
- 全てのノードについて、詳細な陽仕様は記述しない

1 つ目の記述方針の背景には、CA 情報検索が複数のユーザ（サイト管理者、運営者、利用者）によって同時に使用される可能性がある。また、CA 情報検索の実行中に、別のアクション（CA 情報の登録や削除、変更など）が実行される可能性もある。こうした場合を VDM++で記述する為に、図 5-35、図 5-36 に記述されているノードごとに 1 つの VDM++の操作とし、各ノードのアクション詳細定義を操作の陽仕様として記述する方針を立てた。こうする事で、【正 3】を実行した後に、（別のユーザによって）【検 2】が実行されるといった場合も VDM++で表現が可能となり、外部設計書に明示されていないフローで CA 情報検索を実行、検査することが可能になる。それぞれの操作の事前条件、事後条件は、CA 情報検索全体に対する条件を、全ての操作が事前条件、事後条件として持つ事とした。例え

ば、CA 情報検索のアクション定義では、CA 情報検索を実行するユーザは、登録権と閲覧権の少なくとも 1 つのユーザ権限を持っている事が前提となっている。その為、全ての操作について、登録権と閲覧権の少なくとも 1 つを持つ事を事前条件とした。また、実行順序に沿って、前に実行される操作の結果が後から実行される操作に与える影響を、事後条件として追加した。例えば、図 5-36 の「該当件数チェック」で規定最大件数以下と判定された後に実行される操作には、検索結果の件数（該当件数）が規定最大件数を越えない事を事後条件として追加した。

もう 1 つの記述方針は、実証実験期間が検査対象の規模に対して十分でない事から、全ての操作について詳細な陽仕様の記述を行う事が難しいと判断した。そこで、CA 情報検索に固有のアクション詳細定義を含む操作についてのみ陽仕様を記述し、それ以外は陰仕様のみ記述した。

II. 適用手順のテーラリング

CA 情報検索の形式記述を作成するにあたり、『形式手法適用手順（VDM++編）【リリース版】』 [5]で示された手順を次の様にテーラリングした（図 5-37）。

- 前述の記述方針に従った場合、CA 情報検索は状態遷移が含まれない為、状態遷移定義は手順から削除した
- 陽仕様の記述を行った後に構造定義を見直す等、形式記述作成計画立案と形式記述の作成を繰り返し行なう事で形式記述を作成した
- クラス図や制約条件一覧などの中間成果物は作成せず、VDM++の記述に直接反映した
-

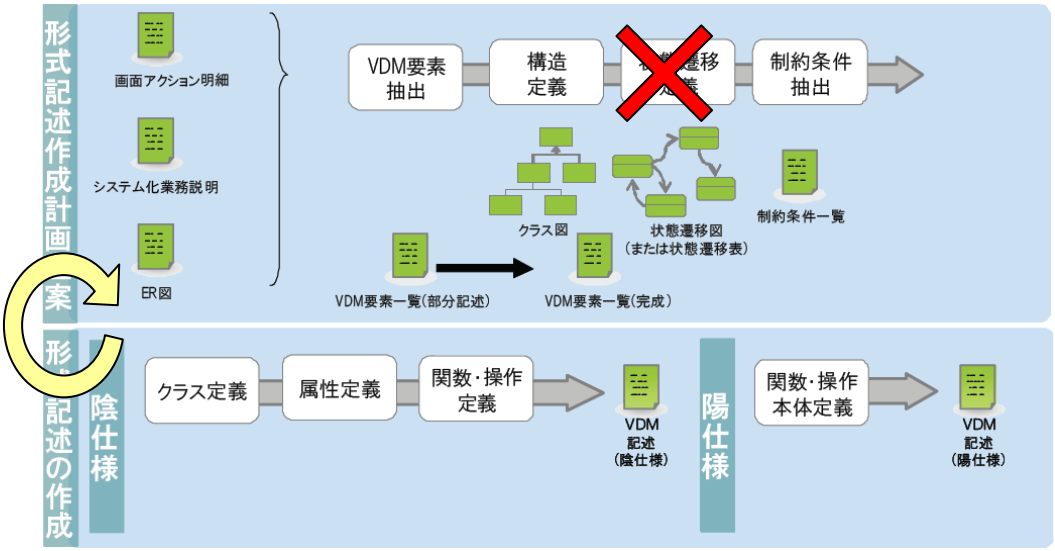


図 5-37　VDM++適用手順のテーラリング

III. 作成した形式記述

```
class コーポレートアクション情報検索ロジック

values
  検索件数上限 = 1;

types
  private 検索方法 = <DTRSSCA0801> | <DTRSSCA0802> | <DTRSSCA0803>

instance variables
  private i ログインユーザ登録権 : bool;
  private i ログインユーザ閲覧権 : bool;

operations
  -- 検索を行い、検索結果の件数を取得する操作
  public 一覧件数取得 : コーポレートアクション検索テーブル DAO * コーポレートアクション通知種別テーブル DAO * seq of char * 検索条件
  ==> set of コーポレートアクション検索テーブル
  一覧件数取得(a 検索テーブル DAO, a 通知種別テーブル DAO, a 通知種別 ID, a 検索条件) == (
    cases true:
      (検索方法選択(a 検索条件) = <DTRSSCA0801>) -> return 検索 DTRSSCA0801(a 検索テーブル DAO, a 通知種別テーブル DAO, a 通知種別 ID, a 検索条件),
      (検索方法選択(a 検索条件) = <DTRSSCA0802>) -> return 検索 DTRSSCA0802(),
      (検索方法選択(a 検索条件) = <DTRSSCA0803>) -> return 検索 DTRSSCA0803(),
      others -> return {}
    end
  )
  pre コーポレートアクション情報検索画面`get 登録権() <> false or コーポレートアクション情報検索画面`get 閲覧権() <> false;

  -- 検索の主操作(この操作が、具体的な検索操作、検索関数を使用する)
  public 一覧取得 : コーポレートアクション検索テーブル DAO * コーポレートアクション通知種別テーブル DAO * seq of char * 検索条件
  ==> set of コーポレートアクション検索テーブル
  一覧取得(a 検索テーブル DAO, a 通知種別テーブル DAO, a 通知種別 ID, a 検索条件) == (
```

CA 情報検索のアクション定義を記述したクラス

実際の上限は 200 件だが、テスティングの都合上、1 としている

【正 3】に対応する操作

【正 5】に対応する操作

```
    cases true:
      (検索方法選択(a 検索条件) = <DTRSSCA0801>) -> return 検索 DTRSSCA0801(a 検索テーブル
DAO, a 通知種別テーブル DAO, a 通知種別 ID, a 検索条件),
      (検索方法選択(a 検索条件) = <DTRSSCA0802>) -> return 検索 DTRSSCA0802(),
      (検索方法選択(a 検索条件) = <DTRSSCA0803>) -> return 検索 DTRSSCA0803(),
      others -> return {}
    end
  )
  pre コーポレートアクション情報検索画面`get 登録権() <> false or コーポレートアクション情報
検索画面`get 閲覧権() <> false
  post card(RESULT) <= 検索件数上限 and card(RESULT) > 0;

  -- 検索条件に書類 ID が指定された場合
  private 検索 DTRSSCA0801 : コーポレートアクション検索テーブル DAO * コーポレートアクション
通知種別テーブル DAO * seq of char * 検索条件
  ==> set of コーポレートアクション検索テーブル
  検索 DTRSSCA0801(a 検索テーブル DAO, a 通知種別テーブル DAO, a 通知種別 ID, a 検索条件) ==
    -- ユーザの登録権、閲覧権の有無を確認する
    -- 条件 1：登録権を持つ場合
    if i ログインユーザ登録権 = true then (
      let
        w 権利確定日検索結果 = {b | b in set 権利確定日検索(登録権用公開状態検索(a 検索テーブ
ル DAO.getAllRecord())) & b.get 書類 ID() = a 検索条件.get 書類 ID()},
        w 議決権代理行使指図書締切日検索結果 = {c | c in set 議決権代理行使指図書締切日検索(登
録権用公開状態検索(a 検索テーブル DAO.getAllRecord())) & c.get 書類 ID() = a 検索条件.get 書類
ID()}
      in
        return w 権利確定日検索結果 union w 議決権代理行使指図書締切日検索結果
    )
    -- 条件 2：閲覧権を持つ場合
    else if i ログインユーザ閲覧権 = true then (
      let
        w 権利確定日検索結果 = {b | b in set 権利確定日検索(閲覧権用公開状態検索(a 検索テーブ
ル DAO.getAllRecord())) & b.get 書類 ID() = a 検索条件.get 書類 ID()},
        w 議決権代理行使指図書締切日検索結果 = {c | c in set 議決権代理行使指図書締切日検索(閲
覧権用公開状態検索(a 検索テーブル DAO.getAllRecord())) & c.get 書類 ID() = a 検索条件.get 書類
```

【正 3】、【正 5】の具体的な処理内容

```
ID()}
      in
        return w権利確定日検索結果 union w議決権代理行使指図書締切日検索結果
    )
    -- 条件3：登録権と閲覧権の両方を持つ場合
    -- 条件4：登録権も閲覧権も持たない場合
    else return {}
  pre a検索条件.get書類ID() <> "";

  -- 検索結果の件数チェック
  public 取得件数チェック : set of コーポレートアクション検索テーブル ==> bool
  取得件数チェック(aテーブル) == (
    let
      w検索結果件数 = card(aテーブル)
    in
      if w検索結果件数 <= 検索件数上限 and w検索結果件数 > 0 then return true
      else return false
  );

end コーポレートアクション情報検索ロジック
```

該当件数チェックに相当する操作
条件分岐もノードの１つなので、操作とした

図 5-38 作成した形式記述

✧ 静的検査

形式記述を記述する過程で目視による検査を行い、外部設計書の記述不足などを抽出した。また、VDMTools による構文チェック、型チェックを実行した。

✧ 動的検査

陽仕様を記述した操作を用いて、CA 情報検索のテスティングを実施した。
テストケースとして、CA 情報検索のフローを表す実行シナリオを作成した。実行シナリオでは形式記述で作成した操作を組み合わせ、外部設計書のアクション定義に明示されていない実行シナリオを作成した。
具体的なテストケースとして以下の実行シナリオを作成し、CA 情報検索のテスティングを行なった。

- **CA 情報検索の処理中に、（サイト管理者によって）アクセス権限が変更された場合**
- **CA 情報検索の処理中に、（別のユーザによって）CA 情報の登録が行なわれた場合**

記述したテストケースは以下の通り。

```
class コーポレートアクション情報検索テスト is subclass of TestCase

operations
  -- ログインユーザ権限が無い状態で検索を実行するテストシナリオ
  public test1 : () ==> ()
  test1() ==
    def
      db1 = new DB();
      db2 = new DB();
      w コーポレートアクション検索テーブル DAO : コーポレートアクション検索テーブル DAO = new コーポレートアクション検索テーブル DAO(db1);
      w コーポレートアクション通知種別テーブル DAO : コーポレートアクション通知種別テーブル DAO = new コーポレートアクション通知種別テーブル DAO(db2);
      rec1 : コーポレートアクション検索テーブル = new コーポレートアクション検索テーブル();
    in (
      db1.createTable(w コーポレートアクション検索テーブル DAO.getTableName());
      db2.createTable(w コーポレートアクション通知種別テーブル DAO.getTableName());

      rec1.set 書類 ID("1111");
      rec1.set 版数(1);
      rec1.set 公開状態(<公開>);
      rec1.set 削除状態(<なし>);
      rec1.set 銘柄コード("");
      rec1.set 銘柄名("");
      rec1.set 国名("");
      rec1.set 権利種類("");
      rec1.set 権利確定日(mk_token("20111201"));
      rec1.set 権利落ち日(mk_token("20111201"));
      rec1.set 現地配当金支払日(mk_token("20111201"));
      rec1.set 配当金円転日(mk_token("20111201"));
      rec1.set 国内配当金支払開始（予定）日(mk_token("20111201"));
      rec1.set 現地株式割当日(mk_token("20111201"));
      rec1.set 国内株式割当日(mk_token("20111201"));
      rec1.set 総会開催日(mk_token("20111201"));
      rec1.set 議決権代理行使指図書締切日(mk_token("20111201"));
```

CA 情報検索の処理中に、アクセス権限が変更された場合の実行シナリオ

DB に登録するテストデータを作成

```
        rec1.set株式等売却代金円転日１(mk_token("20111201"));
        rec1.set株式等売却代金国内支払開始（予定）日１(mk_token("20111201"));
        rec1.set株式等売却代金円転日２(mk_token("20111201"));
        rec1.set株式等売却代金国内支払開始（予定）日２(mk_token("20111201"));
        rec1.set締切日１(mk_token("20111201"));
        rec1.set締切日２(mk_token("20111201"));
        rec1.set締切日３(mk_token("20111201"));
        rec1.set締切日４(mk_token("20111201"));
        rec1.set締切日５(mk_token("20111201"));

        wコーポレートアクション検索テーブルDAO.insert(rec1);
```

作成したテストデータをDBに登録

```
        iコーポレートアクション情報検索画面.検索条件入力("", "", "", <指定無し>, <全て>, <日付指定無し>, mk_token(""), mk_token(""), "1111");

        -- 検索開始前のログインユーザ権限の取得
        -- 検索処理は、ここで取得したログインユーザ権限で実行される
        iコーポレートアクション情報検索ロジック.ユーザ権限取得();

        -- 検索結果の一覧件数を取得(1回目の検索)
        let - = iコーポレートアクション情報検索ロジック.一覧件数取得(wコーポレートアクション検索テーブルDAO, wコーポレートアクション通知種別テーブルDAO, "", iコーポレートアクション情報検索画面.get検索条件()) in skip;

        -- ログインユーザ権限の変更(登録権、閲覧権の削除)
        -- 状況としては、サイト管理者によってログインユーザ権限が変更された状況
        iコーポレートアクション情報検索画面.set登録権(false);
        iコーポレートアクション情報検索画面.set閲覧権(false);

        -- 検索結果を取得(2回目の検索)
        -- このタイミングで、一覧取得の事前条件違反が発生する見込み
        let - = iコーポレートアクション情報検索ロジック.一覧取得(wコーポレートアクション検索テーブルDAO, wコーポレートアクション通知種別テーブルDAO, "", iコーポレートアクション情報検索画面.get検索条件()) in skip;
    );
```

【正3】に相当する操作を実行

ユーザのアクセス権限を変更

【正5】に相当する操作を実行

```
-- 検索結果が上限を超える場合のテストシナリオ
public test2 : () ==> ()
test2() ==
  def
    db1 = new DB();
    db2 = new DB();
    w コーポレートアクション検索テーブル DAO : コーポレートアクション検索テーブル DAO = new コーポレートアクション検索テーブル DAO(db1);
    w コーポレートアクション通知種別テーブル DAO : コーポレートアクション通知種別テーブル DAO = new コーポレートアクション通知種別テーブル DAO(db2);
    rec1 : コーポレートアクション検索テーブル = new コーポレートアクション検索テーブル();
    rec2 : コーポレートアクション検索テーブル = new コーポレートアクション検索テーブル();
  in (
    db1.createTable(w コーポレートアクション検索テーブル DAO.getTableName());
    db2.createTable(w コーポレートアクション通知種別テーブル DAO.getTableName());

    rec1.set 書類 ID("1111");
    rec1.set 版数(1);
    rec1.set 公開状態(<公開>);
    rec1.set 削除状態(<なし>);
    rec1.set 銘柄コード("");
    rec1.set 銘柄名("");
    rec1.set 国名("");
    rec1.set 権利種類("");
    rec1.set 権利確定日(mk_token("20111201"));
    rec1.set 権利落ち日(mk_token("20111201"));
    rec1.set 現地配当金支払日(mk_token("20111201"));
    rec1.set 配当金円転日(mk_token("20111201"));
    rec1.set 国内配当金支払開始（予定）日(mk_token("20111201"));
    rec1.set 現地株式割当日(mk_token("20111201"));
    rec1.set 国内株式割当日(mk_token("20111201"));
    rec1.set 総会開催日(mk_token("20111201"));
    rec1.set 議決権代理行使指図書締切日(mk_token("20111201"));
    rec1.set 株式等売却代金円転日１(mk_token("20111201"));
    rec1.set 株式等売却代金国内支払開始（予定）日１(mk_token("20111201"));
    rec1.set 株式等売却代金円転日２(mk_token("20111201"));
```

CA 情報検索の処理中に、DB に新しいデータが追加された場合の実行シナリオ

DB に登録するテストデータを作成

```
rec1.set 株式等売却代金国内支払開始（予定）日２(mk_token("20111201"));
rec1.set 締切日１(mk_token("20111201"));
rec1.set 締切日２(mk_token("20111201"));
rec1.set 締切日３(mk_token("20111201"));
rec1.set 締切日４(mk_token("20111201"));
rec1.set 締切日５(mk_token("20111201"));

w コーポレートアクション検索テーブル DAO.insert(rec1);
```

作成したテストデータを DB に登録

```
i コーポレートアクション情報検索画面.検索条件入力("", "", "", <指定無し>, <全て>, <日付指定無し>, mk_token(""), mk_token(""), "1111");

-- 検索開始前のログインユーザ権限の取得
i コーポレートアクション情報検索ロジック.ユーザ権限取得();
```

【正 3】に相当する操作を実行

```
-- 検索結果の一覧件数を取得(1 回目の検索)
let - = i コーポレートアクション情報検索ロジック.一覧件数取得(w コーポレートアクション検索テーブル DAO, w コーポレートアクション通知種別テーブル DAO, "", i コーポレートアクション情報検索画面.get 検索条件()) in skip;
```

DB に登録する追加のデータを作成

```
rec2.set 書類 ID("1111");
rec2.set 版数(2);
rec2.set 公開状態(<公開>);
rec2.set 削除状態(<なし>);
rec2.set 銘柄コード("");
rec2.set 銘柄名("");
rec2.set 国名("");
rec2.set 権利種類("");
rec2.set 権利確定日(mk_token("20111201"));
rec2.set 権利落ち日(mk_token("20111201"));
rec2.set 現地配当金支払日(mk_token("20111201"));
rec2.set 配当金円転日(mk_token("20111201"));
rec2.set 国内配当金支払開始（予定）日(mk_token("20111201"));
rec2.set 現地株式割当日(mk_token("20111201"));
rec2.set 国内株式割当日(mk_token("20111201"));
rec2.set 総会開催日(mk_token("20111201"));
```

```
        rec2.set 議決権代理行使指図書締切日(mk_token("20111201"));
        rec2.set 株式等売却代金円転日１(mk_token("20111201"));
        rec2.set 株式等売却代金国内支払開始（予定）日１(mk_token("20111201"));
        rec2.set 株式等売却代金円転日２(mk_token("20111201"));
        rec2.set 株式等売却代金国内支払開始（予定）日２(mk_token("20111201"));
        rec2.set 締切日１(mk_token("20111201"));
        rec2.set 締切日２(mk_token("20111201"));
        rec2.set 締切日３(mk_token("20111201"));
        rec2.set 締切日４(mk_token("20111201"));
        rec2.set 締切日５(mk_token("20111201"));

        -- コーポレートアクション検索テーブルに新しいレコードを追加
        -- 状況としては、別のユーザによって新しいコーポレートアクション情報が登録された状況
        w コーポレートアクション検索テーブル DAO.insert(rec2);
                                                    新しいデータを DB に追加
        -- 検索結果を取得(2 回目の検索)
        -- このタイミングで、一覧取得の事後条件違反が発生する見込み
        let - = i コーポレートアクション情報検索ロジック.一覧取得(w コーポレートアクション検索テーブル DAO, w コーポレートアクション通知種別テーブル DAO, "", i コーポレートアクション情報検索画面.get 検索条件()) in skip;
    );                                              【正 5】に相当する操作を実行

end コーポレートアクション情報検索テスト
```

図 5-39 記述したテストケース

✧ 抽出した問題点

静的検査、動的検査で抽出した外部設計書の問題点は、以下の５つ。

I. CA 情報検索の規定最大件数を超えた検索結果を取得する

(a) 抽出した内容

CA 情報検索中に CA 情報登録が実行されると、規定最大件数を超えた検索結果を取得して、これを検索結果として画面出力してしまう。

(b) システムへの影響

CA 情報検索結果一覧画面では、規定最大件数を超えた件数を出力する場合の仕様が定義されていない。その為、一覧画面が正しく表示されなくなる可能性が高い。

(c) 抽出方法

テスティングにおいて、CA 情報検索を実行中に DB にデータが登録される実行シナリオを実行した際、【正 5】に相当する操作の事後条件違反が発生した（図 5-40）。

事後条件違反の原因を分析した結果、規定最大件数を超える場合がある事を抽出した。

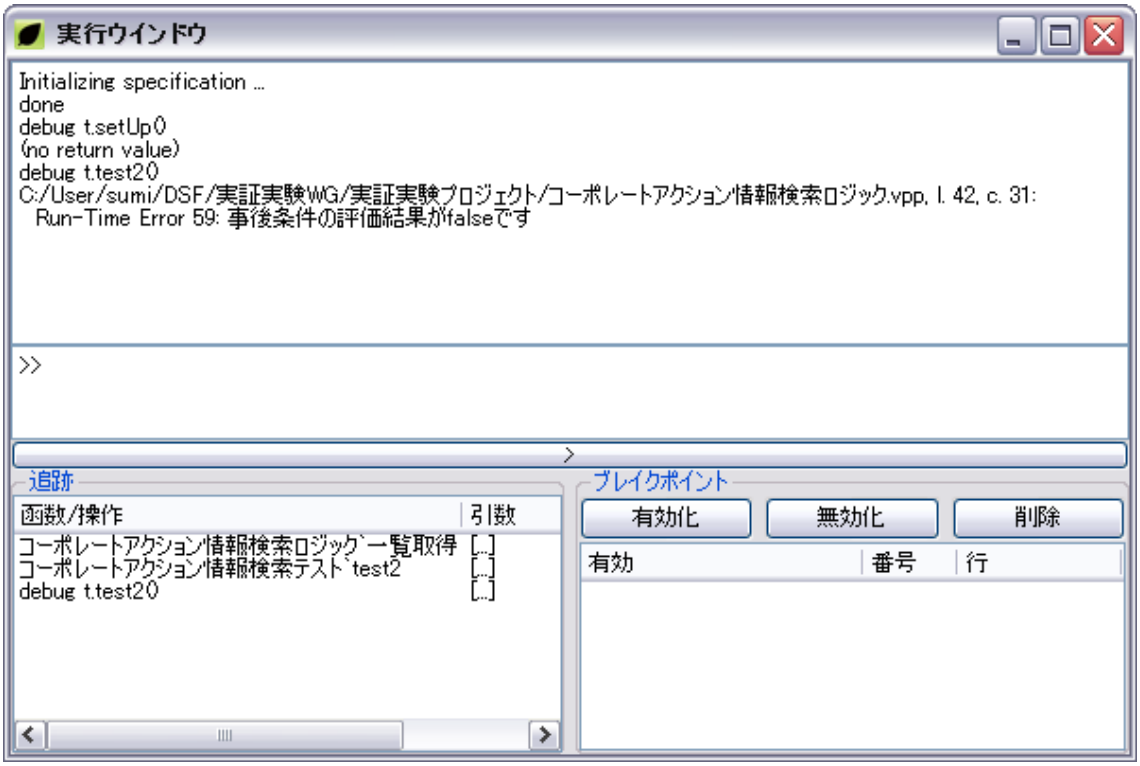


図 5-40　【正 5】の事後条件違反発生

II. アクセス権限を持たないユーザによって CA 情報検索が実行される

(a) 抽出した内容

CA 情報検索中にユーザのアクセス権限が変更されると、閲覧権も登録権も持たないユーザが CA 情報検索結果を取得できる

(b) システムへの影響

アクセス権限を持たないユーザが CA 情報検索結果を取得できる為、情報セキュリティの問題（情報の漏えい）が発生する可能性がある。

(c) 抽出方法

テスティングにおいて、CA 情報検索を実行中にアクセス権限が変更される実行シナリオを実行した際、【正 5】に相当する操作の事前条件違反が発生した（図 5-41）。

事前条件違反の原因を分析した結果、アクセス権限を持たないユーザが CA 情報検索の

結果を取得できる事を抽出した。

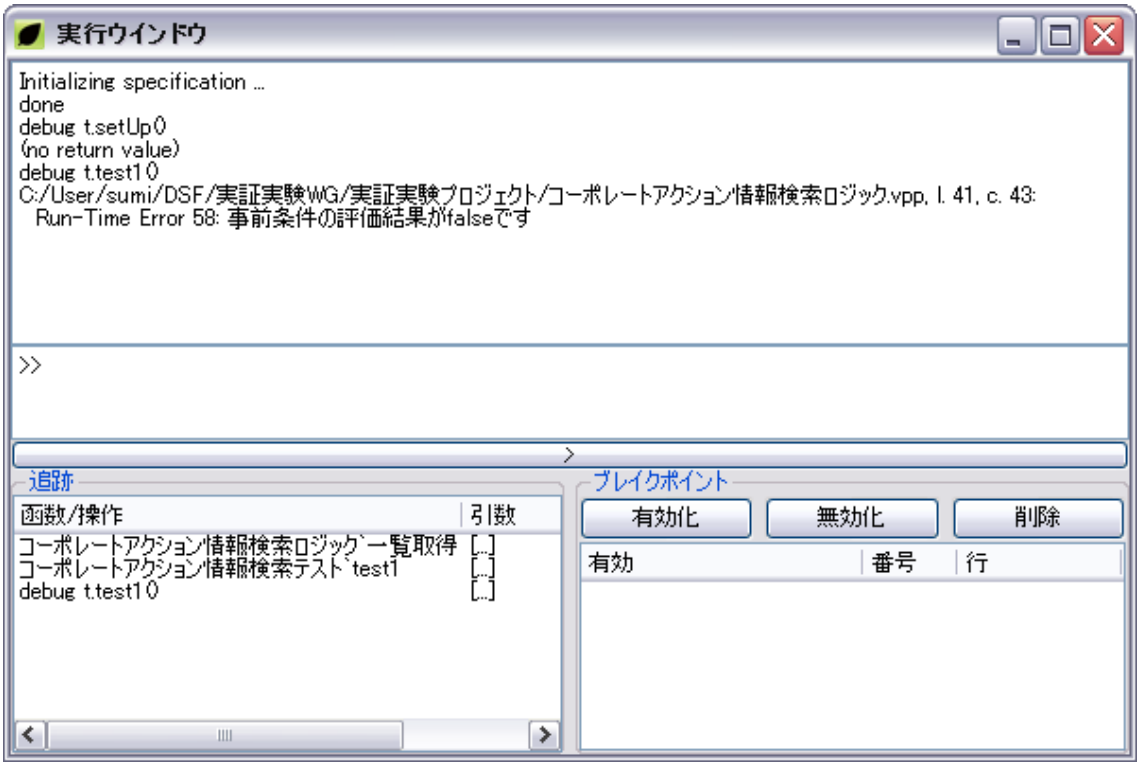


図 5-41 【正 5】の事前条件違反発生

III. アクセス権限による処理内容の分岐の曖昧さ

(a) 抽出した内容

ユーザが登録権と閲覧権の両方を持つ場合に、CA 情報検索として実行する処理内容が明確に定義されていない。

アクション定義では、登録権を持つ場合、閲覧権を持つ場合の 2 通りの処理内容のみ記述されている。その為、両方を持つ場合の処理がいずれか一方のみを行うべきなのか、両方を行うべきなのかが判断できない。

(b) システムへの影響

アクセス権限による処理内容の分岐が曖昧である為、仕様で期待されている検索結果を得られない実装が行われる可能性がある。

(c) 抽出方法

【正 3】、【正 5】に相当する操作の陽仕様を記述しようとした際に、アクセス権限による条件分岐が曖昧である事に気付いた。

IV. CA 情報検索の規定最大件数の定義の曖昧さ

(a) 抽出した内容

CA 情報検索では、権利確定日と議決権代理行使指図書の 2 つの日付について検索を行うが、検索結果の規定最大件数が 2 つの日付の検索結果の合計に対する定義なのか、それぞれの検索結果に対する定義なのかが曖昧。

(b) システムへの影響

規定最大件数の定義が曖昧な為、検索結果の件数による判定を誤って実行する可能性がある。

(c) 抽出方法

検索結果の件数による分岐に相当する操作の陽仕様を記述しようとした際に、条件判定の対象となる検索結果の定義が曖昧である事に気付いた。

V. アクション詳細定義の記述内容の矛盾

(a) 抽出した内容

CA 情報検索のアクション定義において、CA 情報検索結果一覧画面の項目に相当する出力内容を定義している箇所に、CA 情報検索結果一覧画面のアクション定義には含まれていない項目が記述されている。

(b) システムへの影響

CA 情報検索と CA 情報検索一覧画面の両者の定義に矛盾がある為、2 つの画面間で正しくデータ（CA 情報検索結果）を授受できない可能性がある。

(c) 抽出方法

CA 情報検索のアクション定義の形式記述を作成する為に外部設計書の調査を行っている際に、記述内容に矛盾がある事に気付いた。

5.3.4.5 作業者 E

✧ 目的

外部設計書から読み取れる要求仕様[12]を記述し、内部矛盾がないか、要求に適合するか検証する。これは、記述対象とした外部設計書の対象範囲が詳細過ぎるにも関わらず、記述内容が曖昧すぎるため、「設計」内容を記述し検証を行うことは、実証実験の工数・期間

[12] ここでは、通常より狭義の定義で「システムが何をするかを記述した VDM++による形式記述」を要求仕様と呼ぶ。

内では困難と判断したためである。

✧ 記述対象

他のメンバーと記述内容が重複せず、コーポレートアクション情報全体の形式記述と検証を行えるようにするため、コーポレートアクション情報のうち、強制削除機能を記述した。

I. 記述対象としなかった設計情報

ユースケースレベルの機能を記述し、画面処理などの GUI は目的とする要求仕様記述には役立たないため、記述しないこととした。

ユーザ権限は記述しないこととした。他チームが重点的に記述する予定で、かつ、記述量が大きくなり、工数的に間に合わないと判断したためである。

要求仕様記述にはあまり必要ないため、DB 接続エラー、リモート接続エラーなどの、アプリケーション階層以外のエラーは記述しないこととした。

その他、工数の関係上、重要で無いと思われる機能の記述を割愛した。

✧ 形式記述の作成

I. 記述方針

コーポレートアクション情報の形式記述では、システムの特徴から考えて、以下のような問題点があることが類推できたので、その部分を重点的に記述することとした。

- 書類の版と公開日時まわりに多数の制約が発生する可能性が有り、それらに関係した矛盾が発生する可能性がある。
- 「関連リンク一括削除」を行うことにより、種々の制約条件を犯してしまうことによる矛盾が発生する可能性が強い。

II. 提供手順のテーラリング

作業者 E は VDM++記述に慣れているため、基本的手順は提供手順に従うものの、中間成果物を作成することは避けて工数を縮小することにした。

外部設計書の解読を進める過程で VDM++のソースコード断片を作成していき、VDMTools の構文チェック・型チェックによる静的チェックを多用することで、仕様の品質を担保した。

III. 作成した形式記述

強制削除の VDM++による形式記述のトップレベル操作「強制削除する」の本体は図 5-42 のようになる。

強制削除に必要な情報を「強制削除に必要な情報を得る」操作で集め、「DTRSSCA0602_コーポレートアクション検索テープル更新」操作と、「FTRBCBC1104_関連リンク一括削除」操作を使って、データベースに反映するだけである。

```
public 強制削除する : DB * ユーザ * [書類ID型] ==> bool
強制削除する(aDB, aユーザ, a書類ID) == (
  let w入力チェック結果 = 入力内容が適切である(a書類ID) in
  if w入力チェック結果 <> true then (
    display(a書類ID ^ w入力チェック結果);
    return false
    )
  else (
    dcl wコーポレートアクション検索DAO : コーポレートアクション検索DAO
                  := new コーポレートアクション検索DAO(aDB),
       w関連リンクテーブルDAO : 関連リンクテーブルDAO := new 関連リンクテーブルDAO(aDB),
       w提供書類テーブルDAO : 提供書類テーブルDAO := new 提供書類テーブルDAO(aDB);
    def mk_(wコーポレートアクション検索テーブル, w関連リンク情報_戻り値列) =
       強制削除に必要な情報を得る(
            w関連リンクテーブルDAO, w提供書類テーブルDAO,
            wコーポレートアクション検索DAO, aユーザ, a書類ID)
    in
    if wコーポレートアクション検索テーブル = nil then (
      display("MSGTRBCBC40501:{" ^ a書類ID ^
            "}で指定された｛コーポレートアクション情報｝が見つかりません。");
      return false
      )
    else (
      DTRSSCA0602_コーポレートアクション検索テープル更新(
            wコーポレートアクション検索DAO, wコーポレートアクション検索テーブル);
      new 業務共通_関連リンク機能().FTRBCBC1104_関連リンク一括削除(
            w関連リンクテーブルDAO, a書類ID);
      return true
    );
  );
)
```

図 5-42 「強制削除する」の本体

「強制削除する」の事前条件は、以下である。

```
pre
  サイト管理者である(a ユーザ)
```

図 5-43 「強制削除する」操作の事前条件

「強制削除する」の事後条件は、図 5-44 に示すとおりである。

基本的には、「未公開文書である」以外は、「関連書類が削除されている」ことが表明されている。

書類の版や公開日時に関する制約条件があるはずだが、外部設計書からそれらの制約条件を読み取ることはできなかった。

```
post
  let w 関連リンクテーブル DAO : 関連リンクテーブル DAO = new 関連リンクテーブル DAO(aDB),
    w 提供書類テーブル DAO : 提供書類テーブル DAO = new 提供書類テーブル DAO(aDB),
    w コーポレートアクション検索 DAO = new コーポレートアクション検索 DAO(aDB),
    mk_(w コーポレートアクション検索テーブル, w 関連リンク情報_戻り値列) =
      強制削除に必要な情報を得る(
                w 関連リンクテーブル DAO, w 提供書類テーブル DAO,
                w コーポレートアクション検索 DAO, a ユーザ, a 書類 ID),
    w 書類種別 ID = w コーポレートアクション検索テーブル.get 書類種別 ID()
  in
  if w コーポレートアクション検索テーブル = nil then
    RESULT = false
  else (
    if 未公開文書である(w コーポレートアクション検索テーブル) then
      メール送信停止状態である()
    else
      関連書類が削除されている(
            w 関連リンクテーブル DAO, w 提供書類テーブル DAO,
            w コーポレートアクション検索 DAO, w 書類種別 ID, a ユーザ, a 書類 ID)
    );
```

図 5-44 「強制削除する」操作の事後条件

事後条件のうちの「関連書類が削除されている」は、図 5-45 のようになる。

データベースから関連の情報が削除されているという、自明な事後条件であるが、その

際にチェックすべき制約条件を外部設計書から読み取ることはできなかった。

```
関連書類が削除されている :
    関連リンクテーブル DAO * 提供書類テーブル DAO * コーポレートアクション検索 DAO *
    書類`書類種別 ID 型 * ユーザ * 書類 ID 型 ==> bool
関連書類が削除されている(a 関連リンクテーブル DAO, a 提供書類テーブル DAO, a コーポレートアクション検索 DAO, a 書類種別 ID, a ユーザ, a 書類 ID) ==
  return
    コーポレートアクション検索テーブルから論理削除されている(
        a コーポレートアクション検索 DAO, a 書類 ID) and
    コーポレートアクション通知種別テーブルから論理削除されている() and
    関連書類テーブルから論理削除されている(
        a 関連リンクテーブル DAO, a 提供書類テーブル DAO, a 書類種別 ID, a ユーザ, a 書類 ID) and
    メール送信テーブルから論理削除されている()
```

図 5-45 「関連書類が削除されている」操作

業務共通機能「関連リンク登録_エラーチェックあり」の形式記述は図 5-46 のようになる。

関連書類数の最大値のみ外部設計書に記述されていたので、そのチェック方法のみ記述している。

```
public FTRBCBC1102_関連リンク登録_エラーチェックあり :
    関連リンクテーブル DAO * 書類`書類 ID 型 * seq of 関連リンク情報_入力 ==> ()
FTRBCBC1102_関連リンク登録_エラーチェックあり(
     a 関連リンクテーブル DAO, a 書類 ID, a 関連リンク情報_入力列) == (
  dcl w 関連リンクテーブル : 関連リンクテーブル := new 関連リンクテーブル();
     w 関連リンクテーブル.set 書類 ID(a 書類 ID);
  def w 関連リンクテーブル集合 = a 関連リンクテーブル DAO.find(w 関連リンクテーブル);
     w リンクされる最大数文字列 = VDMUtil`val2seq_of_char[nat1](v リンクされる最大数)
  in (
    if 関連書類数が過大である(w 関連リンクテーブル集合, a 書類 ID) then
      display("MSDTRBCBC40901:指定した書類" ^ "{" ^ a 書類 ID ^ "}は、すでに" ^
            w リンクされる最大数文字列 ^ "件の書類より関連づけられています。")
    elseif  not 関連書類がある(w 関連リンクテーブル集合, a 書類 ID) then
      display("MSDTRBCBC40902:関連書類" ^ "{" ^ a 書類 ID ^ "が見つかりません。");
    関連リンクテーブルに追加する(a 関連リンクテーブル DAO, a 関連リンク情報_入力列, a 書類 ID)
  )
);
```

図 5-46 「関連リンク登録_エラーチェックあり」操作

✧ 静的検査

VDMTools の構文チェックと型チェック機能、および証明課題生成機能を使い検査した。
生成された証明課題は、条件式をレビューすることで検査した。

✧ 動的検査

外部設計書を VDM++形式記述に置き換えていく作業の中で、プログラミング工程のコーディングに匹敵する工数が掛かることが分かり、かつ、業務共通_関連リンク機能という、強制削除の中心となる共通機能が、詳細設計工程で仕様が記述されたとのことで、形式記述の仕様実行は、かなりの Q&A を行わなければ不可能となったため、動的検査は行わないこととした。

✧ 発見した問題点

以下の問題点は全て、外部設計書の内容を VDM++による形式記述として、記述しようとした瞬間に発見されたものである。

I. 削除状態の定義

(a) 内容

外部設計書に削除状態 ≠ 非公開と記述されているが、コード設計書の削除状態に非公開という値は無い。

(b) システムへの影響

設計の衝突により、適切に実装されず、利用者に意図しないアクセスを許してしまう可能性がある。

II. 書類種別の定義

(a) 内容

書類種別がコード設計書には通知・提供書類と記述してあり、数行上には規則関係という値もあると記述されているため、「書類種別」と「通知・提供種類」と「規則関係」の関係がよくわからない。

(b) システムへの影響

設計の衝突により、適切に実装されず、利用者に意図しないアクセスを許してしまう可能性がある。

III. 関連リンク情報の定義

(a) 内容

外部設計書の関連リンクに関する記述で、同じ名前で内容の異なる関連リンク情報が 2 種類ある。入力の関連リンク情報は、関連リンク機能 ID・関連リンク書類 ID・連番のようだが、出力の関連リンク情報は、関連リンク機能 ID・関連リンク書類 ID・版数・書類タイトル・関連リンク方向 ・アクションパスとなっている。

(b) システムへの影響

設計の不足により、適切に実装されず、利用者に意図しないアクセスを許してしまう可能性がある。

IV. 関連リンク情報の取得方法

(a) 内容

外部設計書の関連リンクに関する記述で、出力の関連リンク情報にある版数・書類タイトル・関連リンク方向 ・アクションパスは、関連リンクレコードにも存在しないが、どこから取ってくるのか記述されていない。

(b) **システムへの影響**

設計の不足により、適切に実装されず、利用者に意図しないアクセスを許してしまう可能性がある。

V. エラーを返す方法

(a) **内容**

外部設計書の関連リンクに関する記述で、エラーを返す方法が書かれていない。

(b) **システムへの影響**

設計の不足により、適切に実装されず、意図しないエラーが発生する可能性がある。

VI. 権利種類・掲載期限・関連書類選択の定義と取得方法

(a) **内容**

外部設計書で、権利種類、掲載期限、関連書類選択を取得(業務共通)とあり、汎用コードマスタのコード取得の仕様としては、入力（区分、コード)、出力(データ）とあるだけで、権利種類、掲載期限、関連書類選択をどうやって得るか分からない。

また、権利種類、掲載期限、関連書類選択の用語の意味が定義されていない。

(b) **システムへの影響**

設計の不足により、適切に実装されず、利用者に意図しないアクセスを許してしまう可能性がある。

VII. コード設計

(a) **内容**

用語の定義が無く、コード設計書にもコードの名前と値が記述されているのみであり、用語の意味や意図が分からないことが多い。

(b) **システムへの影響**

設計の曖昧さにより、適切に実装されず、意図しないエラーが発生する可能性がある。

VIII. 排他チェック

(a) **内容**

外部設計書で、アクション詳細 ID カラムに、DTRSSCA0104 が排他チェックとなっているが、同一文書の他の箇所では「入力値（書類種別 ID、書類 ID、版数）を条件に、コーポレートアクション検索テーブルより書類情報を取得」となっている。

(b) **システムへの影響**

設計の衝突により、適切に実装されず、意図しないエラーが発生する可能性がある。

IX. 書類種別チェック ID の役割

(a) 内容

外部設計書の関連リンクに関する記述で、書類種別 ID が関連リンク参照インタフェースの入力となっているが、何に使うのか記述されていない。

(b) システムへの影響

設計の不足により、適切に実装されず、意図しないエラーが発生する可能性がある。

5.4 実験結果

5.4.1 工数

本実験において、各作業にかけた工数を表 5-6 に示す。

表 5-6 各工程にかけた工数

作業名	工数（人 H）					
	A	B	C	D	E	計
既存ドキュメントの調査	13.0	15.0	48.0	7.0	44.0	127.0
形式記述作成計画立案	6.0	10.0	2.5	8.0	3.5	30.0
形式記述の作成（陰仕様）	2.0	8.0	0	2.0	1.0	13.0
形式記述の作成（陽仕様）	21.0	14.0	21.0	7.5	9.0	72.5
テスティング	0	0	7.0	4.5	0	11.5
計	42.0	47.0	78.5	29.0	57.5	254.0

5.4.2 設計書と成果物

5.4.2.1 規模

本実験で用いた、本書の対象範囲に関する設計書、作成した中間成果物や形式記述の量を表 5-7 に記す。なお、外部設計書の頁数は翻訳対象ベースで算出している。

表 5-7 成果物の規模

入出区分	成果物	規模
入力	外部設計書	翻訳頁数約 300（参照ベースでは約 700）、 機能数 8、画面数 34、テーブル数 5
出力	形式記述	行数 10,866
	DB フレームワーク	行数 1,082
	テストケース	行数 1,110

5.4.2.2　生産性

前節の入力設計書の規模と形式手法の適用にかかった工数との間には以下の関係がある。

- 生産性（人時/頁）＝ 254.0[人時]／300[頁] ≒ 0.85[人時／頁]（翻訳ベース）
- 生産性（人時/頁）＝ 254.0[人時]／700[頁] ≒ 0.36[人時／頁]（参照ベース）

5.4.3　指摘事項

5.4.3.1　指摘事項種別

本実験において、検出した指摘事項の数を表 5-8 に示す。全 31 個の指摘事項のうち、作業者間で指摘が重複したものは 1 件であった。したがって、重複分を排除した抽出数は 30 個となる。

表 5-8 成果物の規模

指摘事項種別	指摘事項抽出数（個）					
	A	B	C	D	E	計
設計内容の衝突	2	0	3	0	3	8
設計内容の不足	1	2	3	3	5	14
設計内容の曖昧さ	0	1	2	2	1	6
誤字・脱字など	0	1	1	1	0	3
計	3	4	9	6	9	31

* 指摘事項種別

- 設計内容の衝突：（データ構造や判断の条件など）同じ意味を持つべき事柄が記述箇所によって異なる意味をもつように書かれている
- 設計内容の不足：本来あるべき設計がなされていない （複数個所の記述に食い違いがあり、一方の記述の不足により、他方で想定外の状況になることが、想像できる）
- 設計内容の曖昧さ：設計書の書き方が曖昧であるため、人によって異なる解釈が生じる

5.4.3.2　作業と指摘事項の関係

前節の指摘事項が、形式手法を適用する際のどの作業で見つかったかを表 5-9 に示す。

表 5-9 工程別の指摘事項発見数

作業	指摘事項の個数				
	計	曖昧さ	衝突	不足	その他
既存ドキュメントの調査	10 個	1 個	3 個	5 個	1 個
形式記述作成計画立案	1 個	0 個	0 個	1 個	0 個
形式記述の作成（陰仕様）	3 個	1 個	0 個	2 個	0 個
形式記述の作成（陽仕様）	15 個	3 個	5 個	5 個	2 個
テスティング	2 個	1 個	0 個	1 個	0 個
合計	31 個	6 個	8 個	14 個	3 個

5.4.3.3　指摘事項の抽出効率

前節の指摘事項の個数と形式手法の適用にかかった工数との間には以下の関係がある。

- 抽出効率（工数ベース）＝ 31[個数]／254.0[人 H] ＝ 0.12[個数／人 H]
- 抽出効率（指摘事項ベース）＝ 254.0[人 H]／31[個数] ＝ 8.19[人 H／個数]

5.4.3.4　指摘事項の例

各作業者が抽出した指摘事項は 5.3.4 節を参照のこと。

5.5　考察

5.5.1　形式手法の効果

本実験で抽出した指摘事項を分析した結果、形式手法（VDM++による仕様記述と検証）には以下に述べる効果があると考えられる。

✧ 効果：従来手法では内部設計工程以降に見つかる指摘事項が、形式手法では上流設計で抽出できる

本実験で検出した 31 件の指摘事項のうち、VDM++記述を作成するまでに得られたものは

29 件と大半を占めている。形式的な言語によって記述することを前提に検査対象となる外部設計書を調査することで、従来手法とは異なる観点から問題点を発見することができたと考えられる。

5.5.2 形式化のための要件

本実験では、VDM++による仕様記述を作成するにあたり、「抽象化」「用語」「検証可能性」の観点から記述対象を精査した。本節では、これらの観点に関して得られた形式化のための要件を述べる。

5.5.2.1 抽象化

抽象化されているとは、言い換えれば、重要な事柄についてのみ仕様が記述されていて、詳細設計書で記述されるべき項目は記述されていないこと、ということになる。これは、構造化分析・設計の時代から重要視され、オブジェクト指向分析・設計や形式手法でも、最も重要な概念として引き継がれている考え方である。

ある工程の仕様書に、その工程に書くべきでない些末な項目を記述すると、仕様書の規模が大きくなり、作成・更新・保守が困難になり、検証も困難になる。

その工程に適した抽象化を行い、後工程で記述すべき項目は捨象して、検証可能な仕様を作成するのが、構造化分析・設計の時代からの仕様書作成の王道である。

重要な事柄の一番目は、基本的概念である。今回のシステムで言えば、アクセス権の概念や、書類の関連リンクの概念、書類と版と公開日時の関係などが基本概念であった。

次に重要な事柄は、システムのキー項目について仕様が記述されている必要があるということである。ここで言うキー項目は、仕様中の重要な条件判断に用いられる項目と言っても良いだろう。また、データベースのキー項目も多くは対象となる。逆に、それ以外の項目は、入力の際のチェック仕様や、出力の際の編集仕様に登場するだけで、手戻りが必要になるような大きな欠陥に関与することは少ないので、後工程に任せるべきである。外部設計書の場合は、このような項目は詳細設計書に設計を任せればよい。

5.5.2.2 用語

今回の作業で、一番の困難を招いたのは、用語集がなかったことである。ある項目が何であるかの記述がないため、仕様の肝心な所の理解が出来ないか、理解するのにかなりの時間を要することになった。

リアルタイム構造化分析・設計以降の手法では、要求辞書という用語集を拡大した概念が登場し、この要求辞書を要求分析（要件定義）工程で作成することが必要とされている。

要求辞書は、従来の用語集とは異なり、名詞だけでなく、動詞（述語）も辞書として定義する。VDM++に置き換えて言うと、名詞に相当するクラスや型の定義だけでなく、動詞

（述語）に相当するルーチン（関数と操作）のシグネチャ（ルーチン名・引数・返値といったインタフェース）を定義するということである。

これを、外部設計書作成工程の前で定義しておく訳である。そして、外部設計書作成工程では、これに、外部設計で必要な項目を追加していく。

本来は、前工程で作成されているべき検証済の要求辞書がなく、かつ、外部設計工程で追加されるべき項目も定義されていなかったのが、今回の形式記述作成に、かなりの負荷を与えた。

このような要求辞書は、開発チーム内のコミュニケーション・ミスを劇的に減らし、品質の向上と生産性の向上をもたらすことが、ソフトウェア工学の世界ではかなり昔から証明されている。

5.5.2.3　検証可能性

ソフトウェア開発の各工程では、その終了段階で検証を行わなければならない。後工程に持ち越した欠陥が、手戻りのために大きな金銭的ロスをもたらすことが分かっているからである。

検証可能な仕様を記述するためには、何を持って正しいとするかという仕様が記述されていなければならない。検証すべき性質の記述が無ければならないのである。

例えば、版数で言えば、連番になっているべきか、桁数はいくつか、版のマージが許されるか、版の枝分かれがあるかといった仕様が記述されていなければ、検証は不可能である。

この検証のために、一般には仕様のレビューが行われているが、これはVDM++で言えば、構文チェックの一部と、型チェックのごく一部を検証しているにすぎない。検証としては極めて不十分であり、変更余波を把握しきれないという意味で、仕様の修正にも弱い。

形式手法を使わない場合でも、最低限、入力データを想定した仕様のトレースを行うインスペクションを行わなければならない。そのためには、入力と出力、およびそれらを変換する仕様と、検証条件の仕様が書かれていなければならない。すなわち、ある抽象レベルで実行可能な仕様と検証条件（事後条件や不変条件）が書かれていなければならない。

5.6　まとめ

本実験では、外部設計書で用いられる用語の整合性、対象システムの機能と制約条件、仕様の漏れを検証した。従来手法では内部設計以降で見つかる指摘事項が、形式手法を用いることでより上流の工程で抽出できることを示した。

5.7 付録

5.7.1 DBFramework について

5.7.1.1 はじめに

✧ **DBFramework とは？**

本書で解説する本フレームワークは、データベースの基本的な以下の動作を VDM++で表現し、動作させることが可能なフレームワークである。

- Create：レコードの作成
- Read：レコードの検索
- Update：レコードの更新
- Delete：レコードの削除

このフレームワークは、エンタープライズ系システムにおいて、データベースが頻繁に用いられることから、同系システムの仕様記述にデータベースの動作をシミュレートすることが必要であったことから作られた。

厳密なデータベースの振る舞いを正確に表し、動作するわけではないが、最低限の機能を有している。これは、仕様記述に集中するために機能を限定しているためである。

✧ **DBFramework が有する機能**

本フレームワークは、以下の機能を有する。

- レコードに対する CRUD
- 簡易コミット、ロールバック
- 複数データベースを用いる

✧ **クラス構成**

ここで、本フレームワークのクラス構成を、図 5-47 に示す。本フレームワークは DB・DAO・Record の 3 クラスから構成されている。

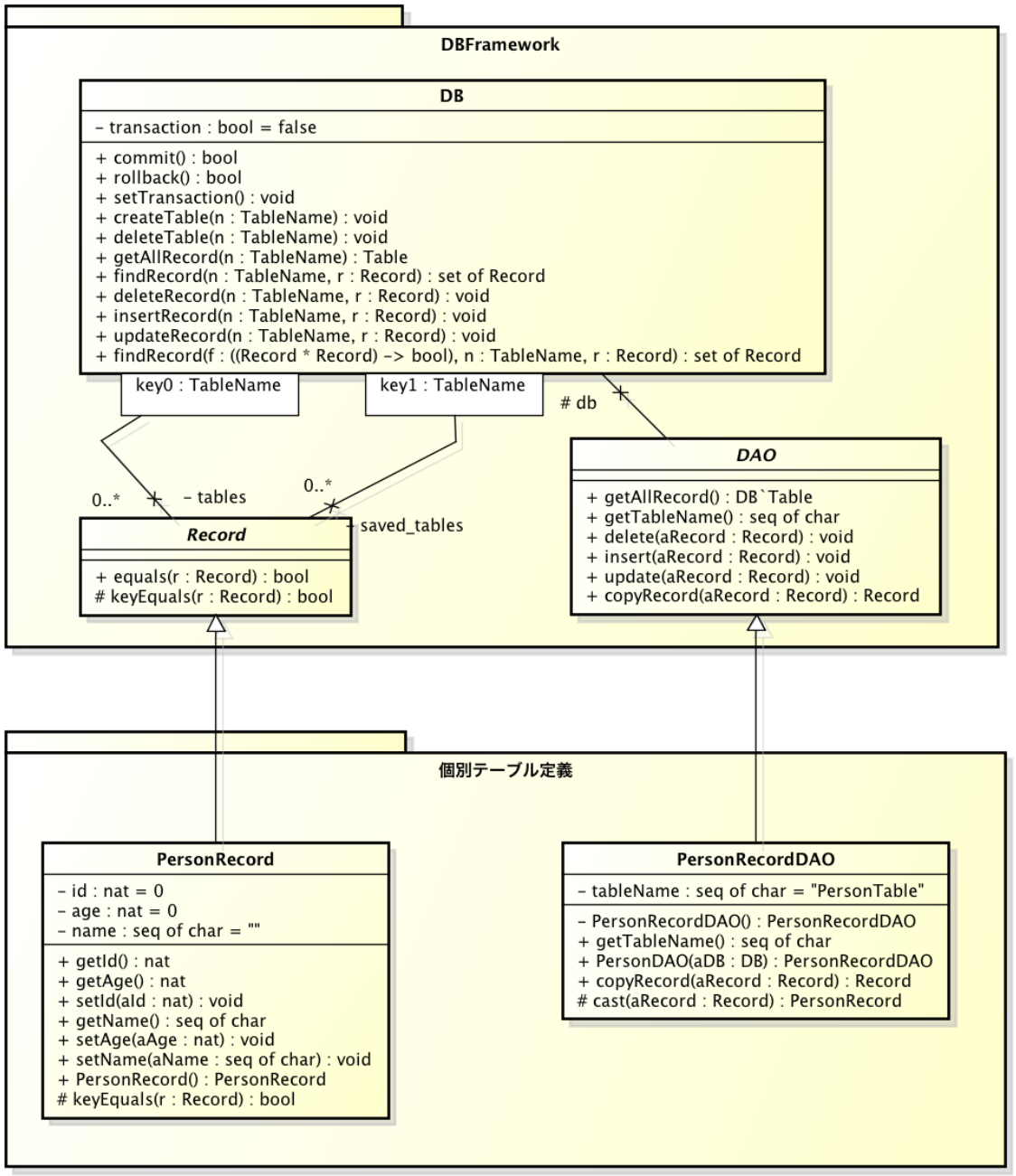


図 5-47 クラス図

DB クラスは、文字通りデータベースを表し、複数のテーブルを保持し、それらのトランザクションを管理する。各テーブルは、テーブル名から Record クラスの集合への写像として表現される。

DAO クラスは、DB が保持するテーブルと 1 対 1 の関係にあり、エンタープライズ系システムで良く用いられる Data Access Object のような動作をする。すなわち、あるテーブルに対する CRUD 処理は、本クラスから実行される。

Record クラスは、データベースのテーブルが保持する 1 行ずつのデータ= レコードを表す。本クラスには、equals というレコードが同じデータであるかを判定する操作を持ち、この判断はレコード間でプライマリーキーが同じであるかでなされる。

5.7.1.2 各クラスの概要

ここで、DBFramework を構成する各クラスについて説明する。

✧ **DB クラス**

DB を表すクラス。必要に応じて、本クラスを継承した子クラスを作ることで、別の DB を表現することも可能。

✧ **DAO クラス**

DAO を表す親クラス。このクラスを継承したクラス経由で DB への CRUD 処理を実行する。

✧ **Record クラス**

各テーブルの 1 レコードを表す親クラス。各テーブルのカラムを全てインスタンス変数として定義する。

なお、equals 操作では、プライマリキーが同じであればレコード自体が同じであると判定する。そのため本クラスを継承するクラスでは、keyEquals 操作をオーバーライドする必要がある。

5.7.1.3 DBFramework に対する VDM 記述の省力化策

本フレームワークを用いて DB のテーブルを表現するには、データ型や構成・名前が異なる多様なデータの集合を扱わなければならない。そのため、VDM++記述を手作業で記述する場合は、似たような記述を数多くする必要が出てくるが、これらを手作業で行うのは、効率が悪い。

しかし、これらのクラスは定型的であることから、自動化による省力化が可能である。ここでは、その省力化策として、Excel 表から DAO クラスと Record クラスの自動生成をするツールを紹介する。

このツールを使うことで、DAO クラス・Record クラスに対する手作業はほぼなくなる。

✧ **生成ツール「DBFrameworkGenerator」**

先に解説した通り、本フレームワークの DAO クラスと Record クラスを Excel 表から自動生成するツール「DBFrameworkGenerator」は、図 5-48 のようなツールである。

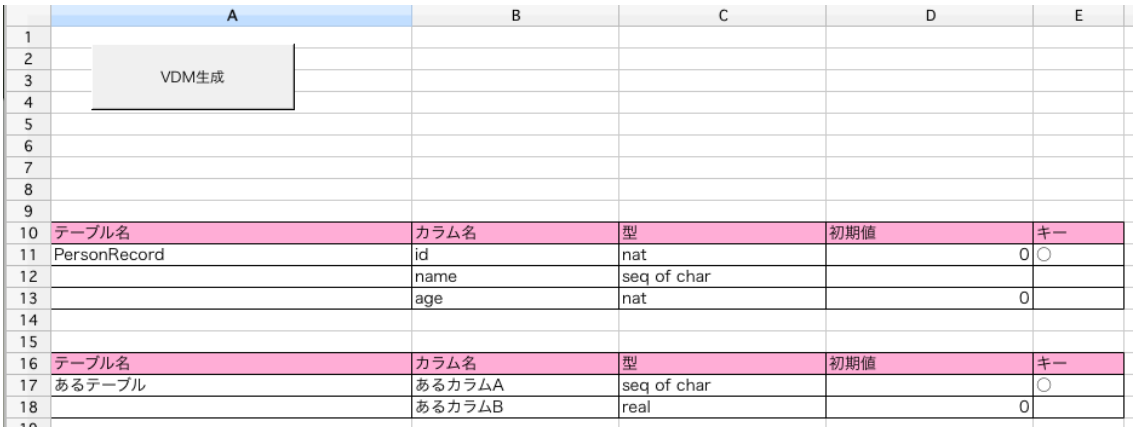


図 5-48 DBFrameworkGenerator

図 5-48 のように、テーブル名・カラム名・カラムの型・初期値・キーをそれぞれ入れることができる。このように入力をし、上部にある「VDM 生成」ボタンを押下すると、各テーブルに合わせた DAO クラスと Record クラスが生成される。

6. 参考実験（その 1）

6.1 目的

本実験は、対象システムが運用されるまで見逃していた改善事項を、形式手法で抽出できるかを確認することを目的とした。そのため、本実験では次のようなルールを設けた。

- 実験運営チームはあらかじめ、ターゲットとする改善事項を一つ設定する
- 形式手法適用者は、上記ターゲットを抽出できた時点で作業完了とする
- 形式手法適用者には、ターゲットとした改善事項の内容は伝えない

ここで実験運営チームがターゲットとして設定した改善事項は、対象システムの開発者によってリストアップされた、運用されるまで見逃していた改善事項の中から選定したものである。

6.2 体制

本実験を行った作業者のスキル（経験）を表 6-1 に記す。なお、本実験の作業者は、既に一度 B1 チームとして実験を行っており、対象システムに対する理解は十分持っている。

表 6-1 体制

作業者	スキル （経験）			
	業務 AP 開発	類似 AP 開発	形式手法	採用手法
A	1 年	0 年	3 年※1	2.5 年

- 業務 AP 開発：業務でのアプリケーション開発の経験年数
- 類似 AP 開発：本実験で対象としたアプリケーションに類似したアプリケーションの開発に限った経験年数
- 形式手法：形式手法全般の経験年数
- 採用手法：本実験で採用した形式手法である Event-B に限ったの経験年数

※1：3 年の内訳

1.250 年…TopSE※2 での講習（1 回 1.5 時間の講義を 48 回、計 72 時間）

0.250 年…上記講習の修了制作

0.125 年…DSF※3 内での講習（1 回 4 時間の講義を 8 回、計 32 時間）

1.375 年…DSF での実務

※2：TopSE

http://www.topse.jp/

※3：DSF

http://www.nttdata.co.jp/dsf/

6.3 作業

6.3.1 作業の概要

本実験では、図 6-1 のような一連の作業を行った。

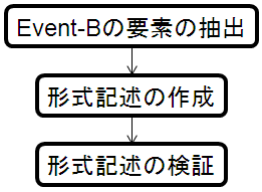


図 6-1 作業の流れ

この作業の流れは、『形式手法適用手順（Event-B 編）【リリース版】』 [1]の「適用法 1」をもとに、本実験の目的外である「工程成果物の反映」と、作業の簡略化のために「リファインメント戦略の立案」、「形式記述の修正」を除いたものである。

本実験における各作業の内容を表 6-2 に記す。

表 6-2 作業の内容

作業名	内容
Event-B の要素の抽出	形式手法を適用する対象について、「イベント」にするシステムのアクション、「不変条件」にするデータ制約、およびそれらを記述するのに必要な「キャリアセット」や「変数」などを抽出する。
形式記述の作成	抽出した Event-B の要素を、Event-B の文法に従って、形式記述にする。
形式記述の検証	形式記述に対して、モデル検査を実施する。モデル検査で指摘事項の抽出に至らなかった場合は、続けて定理証明を実施する。

6.3.2 Event-B の要素の抽出

本実験の作業者は、B1 チームとして一度実験の対象となる外部設計書を読んでいるため、外部設計書に対する理解が、ある程度進んでいる状態で作業を開始している。

6.3.2.1 イベントの抽出

はじめに、与えられた外部設計書から Event-B のイベントとなる要素を抽出した。イベントの候補となるアクションは、図 6-2 のように定義されていた。

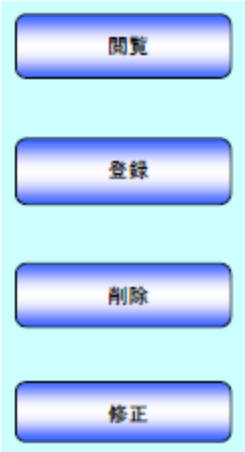


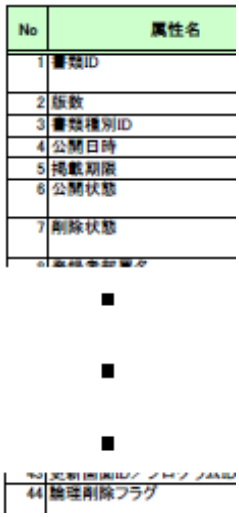
図 6-2 イベントの候補となるアクション一覧（外部設計書より抜粋）

本実験では、まず「登録」、「削除」、「修正」を形式化し、その範囲でターゲットが見つからなければ、「閲覧」を加えて形式化するという計画で作業を行った。「登録」、「削除」、

「修正」を形式化することで、データベース上のデータがどのようになっているかを確認し、「閲覧」を形式化することで、さらにそのデータがユーザにどのように見えるかを確認する。

6.3.2.2　集合・公理・不変条件の抽出

次に、「登録」、「削除」、「修正」により変更されるデータを調査した。「登録」、「削除」、「修正」がアクセスするデータベース上のテーブルは、「現地保管機関テーブル」と「ディレクトリ通知種別テーブル」であったため、これらのテーブルの定義を調べると、図 6-3、図 6-4 のような属性が定義されていた。



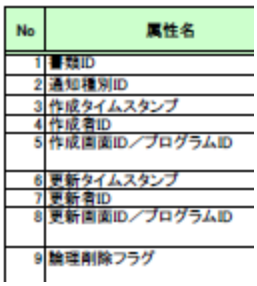
No	属性名
1	書類ID
2	版数
3	書類種別ID
4	公開日時
5	掲載期限
6	公開状態
7	削除状態
44	論理削除フラグ

■

■

■

図 6-3 現地保管機関テーブルの定義



No	属性名
1	書類ID
2	通知種別ID
3	作成タイムスタンプ
4	作成者ID
5	作成画面ID／プログラムID
6	更新タイムスタンプ
7	更新者ID
8	更新画面ID／プログラムID
9	論理削除フラグ

図 6-4 ディレクトリ通知種別テーブルの定義

これらの属性の中から、まずは「閲覧」にのみ影響するものを後回しにし、「登録」、「削除」、「修正」に関係する「書類 ID」、「版数」、「公開状態」、「削除状態」、「公開種別」、「論理削除フラグ」のみを抽出した。

次に、「状態遷移図」や、二つのテーブル間のデータ制約から、不変条件を抽出した。

✧　「状態遷移図」からの不変条件の抽出

「状態遷移図」には図 6-5 のように、「書類」の「公開状態」、「削除状態」、「論理削除フラグ」の組み合わせや、それらの属性が「登録」、「削除」、「修正」によってどのように変化するかが書かれていた。

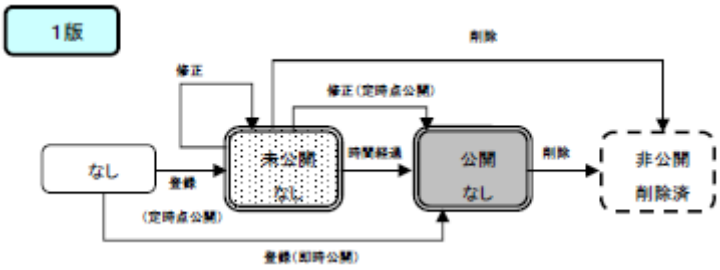


図 6-5 状態遷移図（抜粋）

「状態遷移図」からは、ここに書かれた属性の組み合わせに従い、例えば「公開状態」が「未公開」であるならば、「削除状態」は「なし」といったデータの制約を、不変条件にした。

✧ 「二つのテーブル間での整合性」からの不変条件の抽出

図 6-3、図 6-4 の二つのテーブルには、共通して「書類 ID」と「論理削除フラグ」という属性が存在している。このとき、二つのテーブルで「書類 ID」が一致しているデータは、同じ「書類」のデータを表していることから、「論理削除フラグ」の値も一致しているという不変条件を抽出した。

6.3.3 形式記述の作成

次に、これまでの作業で定義してきたものを、Event-B の文法に従って清書した。この作業は特に何事もなく進み、記述の過程で指摘事項を抽出することはなかった。

6.3.4 形式記述の検証

次に、作成した形式記述に対して、モデル検査を実施した。モデル検査でターゲットを抽出できなかった場合は定理証明も実施するつもりでいたが、本実験ではモデル検査のみでターゲットの抽出に至った。本節では、本実験で抽出したターゲットの説明と、それをどのようにして抽出したかを説明する。

本実験で抽出した改善事項は、6.3.2.2 節の「二つのテーブル間での整合性」から抽出された不変条件の違反であった。具体的には、「現地保管機関テーブル」の、ある「書類 ID」の「論理削除フラグ」が OFF であるにも関わらず、「ディレクトリ通知種別テーブル」の、同じ「書類 ID」の「論理削除フラグ」が ON になる、という改善事項であった。実際にこの改善事項の状況を引き起こす削除アクションのフローチャートを、図 6-6 に示す。

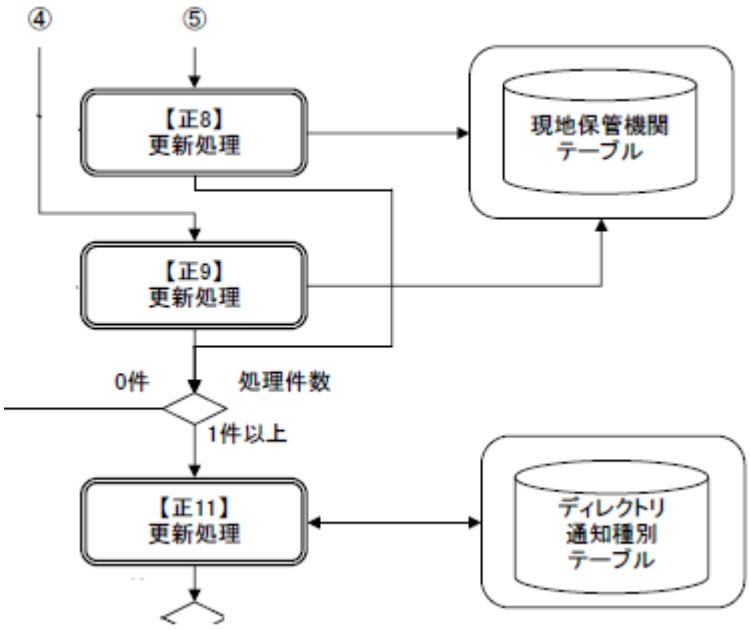


図 6-6 削除アクションのフローチャート（抜粋）

このフローチャートによると、「現地保管機関テーブル」へのアクセスは、「正 8」または「正 9」のどちらか一方によって行われ、その後「正 11」によって「ディレクトリ通知種別テーブル」へのアクセスが行われる。すなわち、フローの組み合わせは、次の二通りである。

- 「正 8」→「正 11」
- 「正 9」→「正 11」

「正 8」、「正 9」、「正 11」の、それぞれの詳細定義は、図 6-7、図 6-8、図 6-9 のようになっていた。

カラム名	セット内容
削除状態	"なし"
条件	1. 書類ID=(入力値)書類ID And 2. 書類種別ID=(保存値)書類種別ID And 3. 公開状態="公開" And 4. 削除状態="変更中"

図 6-7 「現地保管機関テーブル」にアクセスする「正 8」の詳細定義（抜粋）

カラム名	セット内容
公開状態	"非公開"
削除状態	"削除済"
論理削除フラグ	ON

条件	1. 書類ID=(入力値)書類ID And 2. 書類種別ID=(保存値)書類種別ID And 3. 論理削除フラグOFF

図 6-8 「現地保管機関テーブル」にアクセスする「正 9」の詳細定義（抜粋）

カラム名	セット内容
論理削除フラグ	ON

条件	1. 書類ID=(入力値)書類ID

図 6-9 「ディレクトリ通知種別テーブル」にアクセスする「正 11」の詳細定義（抜粋）

これによると、「正 9」→「正 11」のフローでは、「現地保管機関テーブル」の「論理削除フラグ」も「ディレクトリ通知種別テーブル」の「論理削除フラグ」も、どちらも必ず ON になるためデータの不一致は起こらないが、「正 8」→「正 11」のフローでは、「正 8」が「論理削除フラグ」の値を変更しないように設計されているため、「現地保管機関テーブル」の「論理削除フラグ」の値が OFF の状態で「正 8」→「正 11」のフローが実行されると、データの不一致が起こる。

本実験で行ったモデル検査では、実際に「現地保管機関テーブル」の「論理削除フラグ」の値が OFF の状態でこのフローを流れる反例を見つけることに成功した。その状況を図 6-10 に示す。

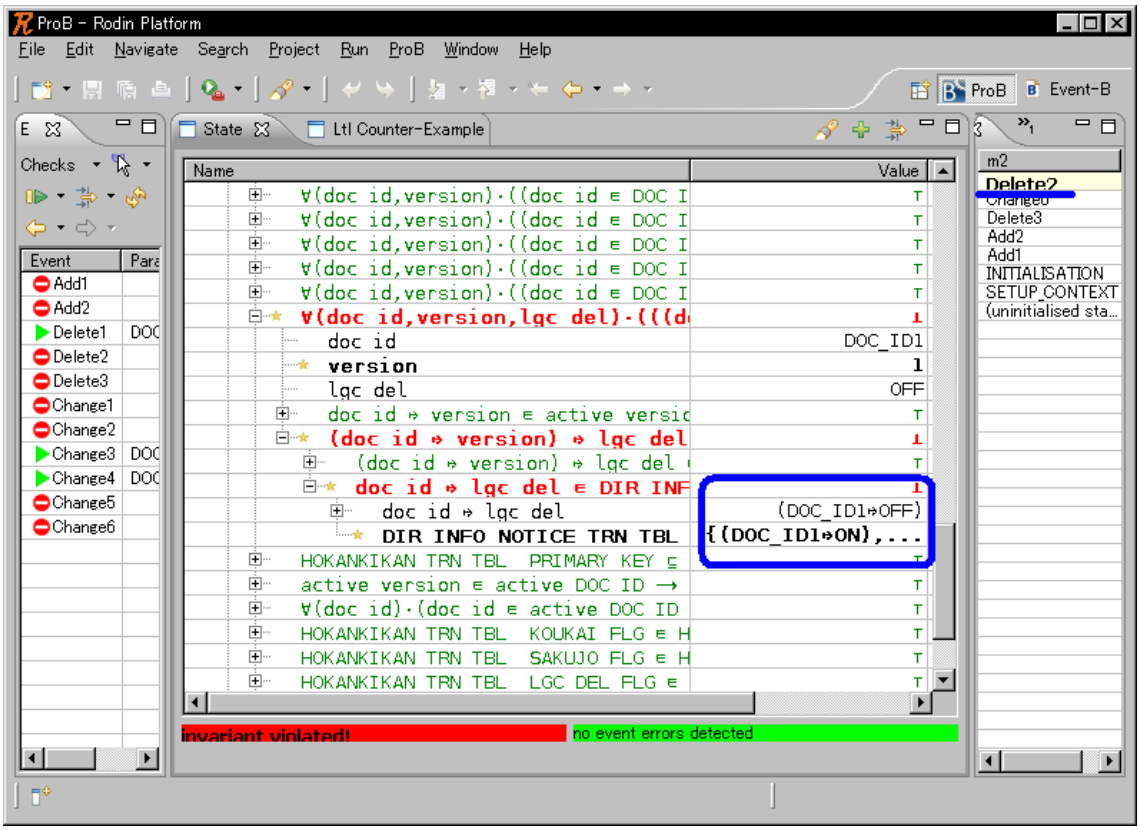


図 6-10 反例

図 6-10 の右列を下から順に見ていくと、初期状態から Add1（登録）や Change6（修正）などを経て、Delete2（図 6-6 の削除アクション）によって違反が起きたことがわかる。中列の詳細によると、モデル検査器が「DOC_ID1」と名付けた「書類 ID」の「論理削除フラグ」が、OFF と ON で食い違っていることが示されている。

6.4 実験結果

本章では、実験にかかった工数、実験対象の規模、指摘した改善事項の数などから、検証の生産性、改善事項の指摘効率について考察する。

6.4.1 工数

表 6-3 工数

作業名	工数（人 H）
Event-B の要素の抽出	0.5
形式記述の作成	1.5
形式記述の検証	0.5
	計 2.5

6.4.2 規模

6.4.2.1 形式記述の対象とした文書の規模

表 6-4 実験対象の規模

	個数	ページ数	
業務	3	48	16 ページ/業務
テーブル	2	3	
状態遷移図	1	5	※
		計 56	

※：状態遷移図のページ数には、状態遷移図に関連する補足情報のページも含む

6.4.2.2 作成した形式記述の規模

表 6-5 形式記述の規模

行数※	686
イベント数	11
変数数	8
不変条件数	51

※：Rodin Platform の Pretty Print に表示されたものを計測。コメントは含まない。

6.5 考察

本実験は、B1 チームの作業者が、B1 チームとしての実験を終えた後、形式化範囲を変えて新たに実験を行ったものである。そこで、B1 チームと本実験との比較について考察する。

B1 チームの時は合計 107.5 人 H の工数をかけたに対し、本実験では 2.5 人 H で実験を完了している。実験の目的が違うことから単純な比較はできないが、概ね次のような要因が影響したのではないかと考える。

✧ 本実験は対象システムへの理解が進んだ状態で開始した

B1 チームで得た対象システムへの理解を引き継いで本実験を行っているため、形式化範囲が変わっているとはいえ、新しい範囲の理解もスムーズに進んだ。このことは、対象の仕様について理解さえしていれば、形式手法の適用自体には大した負荷がかからないということを示している。

✧ 検証手段を制限した

B1 チームではモデル検査と定理証明を併用しているのに対し、本実験ではモデル検査のみを行っている。B1 チームの検証にかかった工数を見てもわかるように、Event-B の検証で最も工数がかかるのは対話的定理証明である。本実験がモデル検査のみで改善事項の抽出に至ったことは、もし工数上の制限が厳しい状況であれば、モデル検査だけでも実施すればそれなりの効果が得られるということを示している。

6.6 まとめ

本実験は、形式手法の一つである Event-B が、運用後まで見逃される類の改善事項を外部設計書から抽出できるかどうかを確認することを目的とし、外部設計書に書かれたシステムのアクションが、同じく外部設計書に書かれたデータの制約を守っているかどうかを検証するというシナリオで、Event-B の適用実験を行った。以降に、本実験の結果をまとめる。

✧ 形式手法で改善事項を抽出できた

本実験では、56 ページの外部設計書（表 6-4）から、ターゲットの改善事項を、2.5 人 H の工数（表 6-3）で抽出した。

✧ 非常に限られた工数でも形式手法の適用は可能であることを示した

本実験では、2.5 人 H の工数（表 6-3）で、形式記述の作成から検証までを行い、改善事項を抽出した。このことで、対象の仕様についての理解さえしていれば形式手法の適用自体は容易であることと、検証手段を制限してもそれなりの効果が得られること（6.5 節）を示した。

7. 参考実験（その２）

7.1 目的

本実験は、外部設計書を形式仕様言語に翻訳するのではなく、外部設計書から読み取れる設計内容を Event-B を使って再構成するアプローチをとった。完成された外部設計書からの直訳でない方が形式手法の特徴が明確になると予測されることに加え、形式手法の実適用においては設計の進行とともに形式記述を作成し設計内容を検証する場合が多いと想定されることから、現実的なアプローチであると考える。一方、題材とした外部設計書は本来第三者によるチェックを意図したものではなく、再構成した形式記述と比較して設計内容の精度を議論することは意味がないという立場から、本実験は外部設計書の改善事項（設計の不足、設計の衝突、設計の曖昧さ）の抽出には関わらない。本実験の目的は、Event-B の記述力が外部設計書を再構成できる程度に高いこと、記述内容が検証によって裏付けられること、記述と検証が妥当な作業時間によって達成できること、を確認することにある。

本実験では、対象システムが提供するアクセス権管理機構が正しく設計されていることを、形式手法を使って検証する作業を行った。具体的には、外部設計書に記載されたアクセス権階層とアクセス権設定／剥奪処理の両者を形式仕様言語を使って記述するとともに、後者によって前者に違反するアクセス権が設定されないことを、形式的に検証した。本実験の目的は、この作業を通じて以下を確認することである。

・　対象とする設計書の必要な部分が、形式仕様言語によって表現できること
・　形式記述の整合性を確認するために必要な検証が、十分な厳密性のもとに行えること
・　上記の記述と検証が、一般的な計算機環境の下で妥当な作業時間内に行えること

ここで、対象システムが提供するアクセス権管理機構が正しく設計されていることの保証は、本実験の目的ではないことに留意していただきたい。与えられた外部設計書は保証を意図しておらず、保証のための評価に適した厳密性を持つものではない。

また、アクセス権管理機構を含むセキュリティ機構全体の設計への形式手法の適用と評価も、本実験の目的ではない。本実験の作業では、与えられた外部設計書のアクセス権階層を基準として、それを満たすアクセス権設定／剥奪処理が行われているとき、アクセス権管理機構が正しく設計されているとした。セキュリティ分析の段階から Event-B を用い、システムのセキュリティ機構と外部環境（管理者など）の役割分担のもとに、想定される攻撃に対抗する仕組みが構成されていることを検証するのは、Event-B の特性を確認する点からも興味深いテーマである。しかし、与えられた外部設計書を大きく変更する再設計は上記の本実験の目的には適合しないうえ、対象業務とシステムのドメイン知識を持つ専門家との共同作業が必要であり、今回の実証実験の範囲外に位置する。

本実験において、何を確認するために、どのような工程成果物に対して、どの形式手法を用いたかを表 7-1 に記す。

表 7-1　目的

採用した手法	Event-B	
（特徴）	Event-B では、不変条件を満たす状態から、あるイベントを１回実行し、実行後の状態が不変条件を満たすことを検証できる。	
確認すること	業務におけるデータの処理が、データの構造と値に関する制約を満たす。	
（確認方法）	データに関する制約と業務機能をそれぞれ Event-B の「不変条件」「イベント」と捉えることにより、複数の業務機能がデータの一貫性を守ることを記述・検証できる。	
確認の対象	設計ドキュメント	形式記述
	画面アクション明細（アクセス権設定／剥奪処理）	抽象機械 の イベント
	エンティティ定義（アクセス権階層）	抽象機械の 定数、変数、不変条件
（備考）	上記は「機能要件の合意形成ガイド」(IPA SEC)における下記の技術領域・工程成果物に相当する。 ・システム化振舞い・システム化業務説明 ・データモデル・エンティティ定義	

外部設計書のアクセス権階層に関する記述は難解であり、個別の処理設計に記載されたアクセス権設定／剥奪処理がアクセス権階層の制約を満たすことは自明ではない。この問題に対して Event-B を適用することで、Event-B の表現力と検証の効果を確認することが、本実証実験の狙いである。

7.2　体制

本実験を行った作業者の役割やスキルなどを表 7-2 に記す。

表 7-2

作業者	役割*1	スキル （経験）			
		業務 AP 開発	類似 AP 開発	形式手法	採用手法
A	形式記述者 形式検証者	0 年	1 年	20 年	5 年

*1 役割とその担当する作業は以下のとおりであるが、本実験では形式記述者が形式検証者を兼ねた。

形式記述者 ： 『形式記述』の作成と修正を行う

形式検証者 ： 『形式記述』に関する検証を行う

7.3 作業

7.3.1 作業の概要

本実験では、図 7-1 のような一連の作業を行った。ただし、7.3.2 節に示す理由により、この一連の作業を２回行っている。１回目は設計書の理解、２回目は設計書の再構成が目的である。

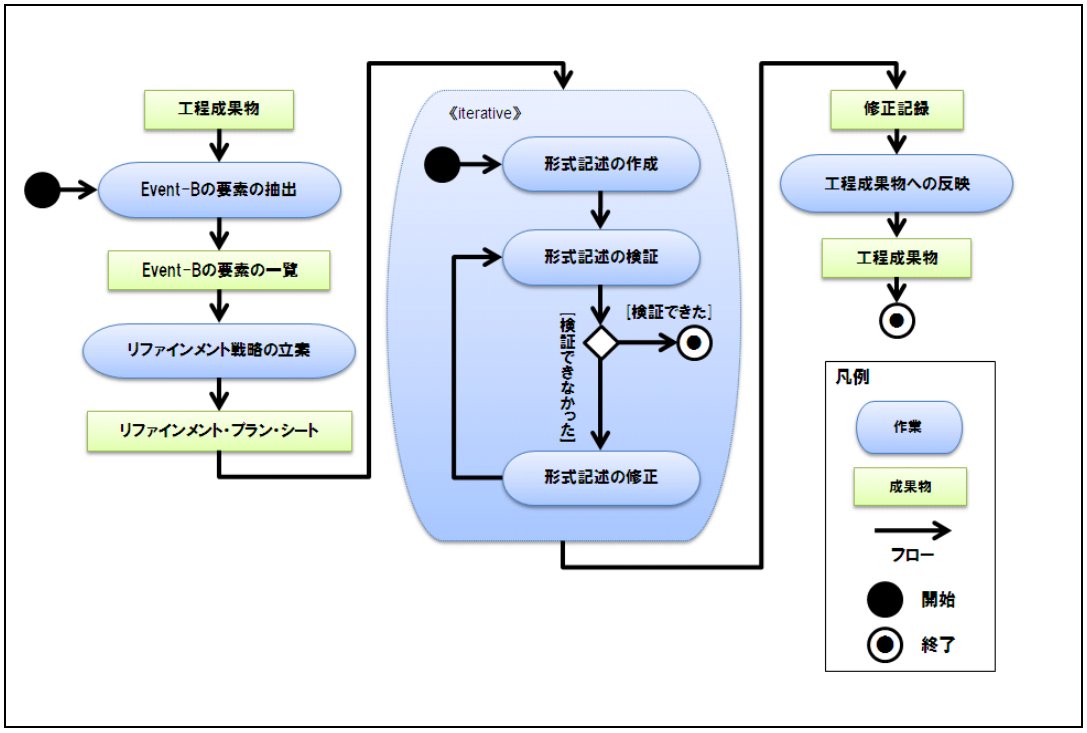


図 7-1 記述/検証作業の流れ

各作業の内容については、『形式手法適用手順（Event-B 編）【リリース版】』 [1]を参照のこと。

7.3.2 外部設計書の再構成

課題:

本実験の対象はセキュリティに関わる機能であり、その設計の正しさを確認するためには、システムの運用を含む正確な情報が必要となる。一方、本実験で参照した外部設計書は十分なドメイン知識を持つ開発担当者によって読まれることを前提としており、ドメインの非専門家には判断のつかない記述も含まれる。特に、一般に設計書は「〇〇は××である」と記述されるが、××であるのは〇〇のみなのか、△△も××であり得るのか（その場合にはさらに、××であるのは〇〇と△△のみなのかが問題となる）は必ずしも明確に記述されないので、検証を厳密に行うためには不足している情報を補う必要がある。本来のシステム開発では、ドメイン知識を持つ開発担当者が形式記述を記述する、あるいは形式記述者がドメイン知識を持つ開発担当者と共同で作業するので、このような混乱は起こり難い。しかし、今回は実験であるため、作業者にはドメイン知識がなく、またドメイン知識を持つ題材提供者と作業者が密に共同作業することも現実的ではなかった。

例:

外部設計書には、サイト管理者はユーザグループに運営者属性を与え得ることが記載されている。このとき、同一サイトについて複数のユーザグループに運営者属性を与えるとサイト運営が混乱することも予想されるが、一つのサイトについて運営者属性を持つサイトは唯一つであることは、外部設計書には記載されていない。

また、題材システムでは「職務」というオリジナルの概念が用いられている。権限（アクセス権）は職務を通じて与えられる、と記載されている点では、ANSI 標準の Role Based Access Control Model の「役割」（role）に類似するが、オリジナルの概念であるため役割と同様に解釈することはできない。例えば、ユーザグループ属性の剥奪時には当該ユーザグループ内で定義された当該サイトに関する全職務を削除する、と外部設計書に記載されているが、職務を役割と解釈した場合には、ユーザグループは一つのサイトについても複数のユーザグループ属性を持ち得るので全職務の削除は必ずしも適切ではない。

対策:

本実験では、検証のためには不足していると考えられる情報について、次のように対応した。

- 題材提供者に確認し、その回答にしたがう
- 外部設計書の記述と乖離しない範囲で、作業者の推測によって情報を補う
- 作業者が設計内容の明確化に必要と判断した部分は、外部設計書との相違が軽微な範囲で変更を加える

このような方針により、形式手法を使って外部設計書の実験対象部分を再構成した。作業者による再構成は、次の２ステップで行った。

ステップ１　設計書の理解

・ 外部設計書の実験対象部分から要素を抽出する
・ 要素間の関連を分析するとともに必要な情報を補足する
・ 上記を形式記述／検証することで補足した情報の不整合を摘出する

ステップ２　設計書の再構成

・ ステップ１の結果をもとに形式記述に必要な要素と補足情報を集積する
・ 形式記述に要素を配置する順序（リファインメント戦略）を検討する
・ 形式記述と検証を実施する

実験対象は複雑であり、外部設計書の日本語と図表によって設計内容を理解し、それらと整合するよう情報を補足ないし変更することは難しかった。そこで、ステップ１では実験対象の理解に Event-B を用い、記述と検証を通じて理解の誤りを検出した。この記述は断片的なものであり、実験対象全体を表すのに適した構造を持っていない。そこで、ステップ２ではステップ１の形式記述を一旦破棄して、リファインメント戦略を立てて改めて Event-B で記述と検証を行った。

結果:

上記の運営者属性の例は題材提供者に確認し、業務上は複数のユーザグループに同一サイトの運営者属性を与えることはないが、システム上はこれを禁止しない設計である、との回答を得た。この回答をもとに、再構成した形式記述でも運営者属性の重複を禁止していない。一方、職務については作業者の判断で変更を加え、ドメイン知識のない作業者にも性質が分かっている（ANSI 標準ドキュメントは厳密性のため形式仕様記述言語 Z で記述されている） Role Based Access Control Model の「役割」と同様に解釈した。これにともない、ユーザグループ属性剥奪時には剥奪された属性に関わるアクセス権のみ削除するものとしている。その他の必要な情報は、作業者の判断で補った。この結果、外部設計書の実験対象部分について、検証の裏づけのある形式記述を再構成することができた。

ここで、対象とした外部設計書に明記されていない事項の解釈は誤っている可能性がある。さらに、外部設計書の全てを参照しているわけではないので、参照しなかった部分で外部設計書との乖離が起こっている可能性もある。すなわちステップ２では、ステップ１の結果を使って、もとの外部設計書と似て非なる設計書を形式手法を使って再構成していると言える。このことは、記述内容の同一性が言えない点から、非形式的に書かれた外部設計書の改善事項抽出に関する実験データ収集には適さない。一方、外部設計書の作成過程を形式手法を使ってたどることになるので、現実の設計作業に近い進め方で作業データを収集することができ、本実験の目的には適合している。

7.3.3　設計書と形式記述の関係

7.3.3.1　アクセス権階層

本実験によって再構成したアクセス権階層の概要を、図 7-2 に示す。図中の５つの円は集合を意味し、サイト集合、ユーザグループ集合、サブグループ集合、２つの職務集合を表す。また、矢印は集合の要素間に対応関係があることを表し、両方向の矢印は多対多の関係を表す一方、片方向の矢印は多対１の関係を表す。例えば、両方向の矢印であるサイトアクセス権は、サイト集合の要素（サイト）とユーザグループ集合の要素（ユーザグループ）の間に多対多の関係があることを意味し、片方向の矢印であるサブグループ構成は、サブグループ集合の要素（サブグループ）とユーザグループ集合の要素（ユーザグループ）の間に多対１の関係があることを意味する。後述するように、両方向の矢印（relation）では、対応関係に関与しない要素が存在しても良い。一方、片方向の矢印（total function）では始点側の全ての要素が対応関係に関与するが、終点側には対応関係に関与しない要素が存在しても良い。各集合の右側には、その集合の要素が持つ属性を記載した。例えば、ユーザグループはユーザグループ属性とユーザグループアクセス権を属性として持つ。職務は、サブグループの下位にあるもの（以下ではサブグループ職務と呼ぶ）はサブグループ職務アクセス権を、ユーザグループの下位にあるもの（以下ではユーザグループ職務と呼ぶ）はユーザグループ職務アクセス権を属性として持つ。

システムの利用者は、管理者から職務を与えられることによって、その職務に許されたアクセス権（職務の属性として与えられる） を持つことができる。アクセス権は、サイト内のオブジェクトに対する操作の許可、例えば書類の閲覧権など、である。

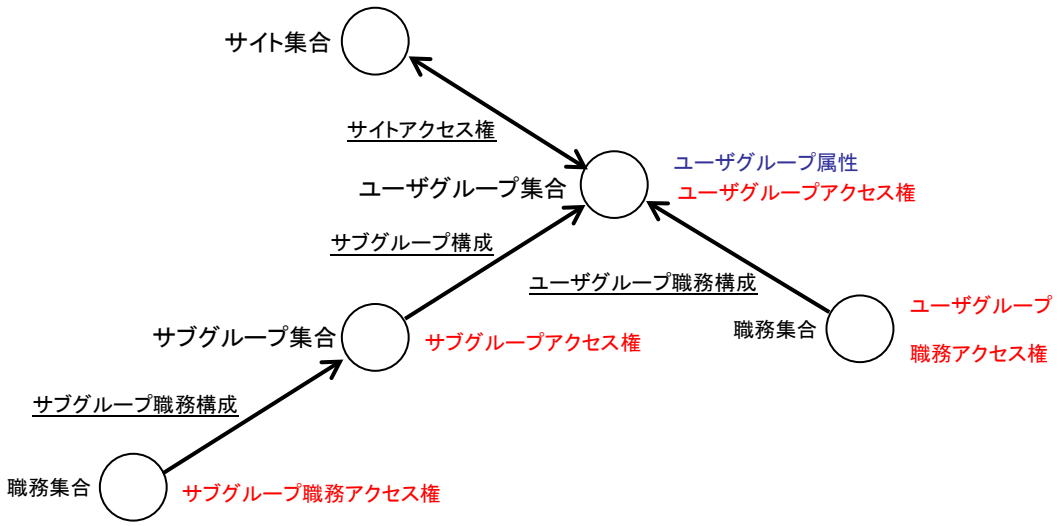


図 7-2　アクセス権階層の概要

以下では、各要素を簡単に説明するとともに、Event-B を使った記述を枠内に示す。簡単のため、下記では Event-B 記述から不変条件のみを抽出し、定数や変数の宣言は省略した。Event-B の表記については、『形式手法イディオム集(Event-B 編)【リリース版】』 [6]を参照のこと。

１． ユーザグループ

ユーザグループは、サイトアクセス権によってサイトと対応付けられる。ここで、１つのユーザグループが複数のサイトに対してサイトアクセス権を持っても良いし、どのサイトへのアクセス権を持たなくても良い。同様に、複数のユーザグループが１つのサイトにサイトアクセス権を持っても良いし、どのユーザグループもサイトアクセス権を持たないサイトが存在しても良い。

inv-a1 ： サイトアクセス権 ∈ ユーザグループ集合 <-> サイト集合

２． サブグループ

ユーザグループは、サブグループを持つ。ここで、ユーザグループが持つサブグループは、０個以上任意個であるとする。各サブグループは、ただ１つのユーザグループと関連付けられる。すなわち、複数のユーザグループと関連付いたサブグループは存在せず、ユーザグループと関連付かないサブグループも存在しない。

inv-b1 ： サブグループ構成 ∈ サブグループ集合 --> ユーザグループ集合

３． ユーザグループ職務

ユーザグループは、サイトアクセス権が与えられたサイトに関して、ユーザグループ職務を持つ。ここで、ユーザグループが持つユーザグループ職務は、０個以上任意個であるとする。ユーザグループ職務は、ただ１つのユーザグループと関連付けられる。

inv-c1 ： ユーザグループ職務構成 ∈ 職務集合 +-> ユーザグループ集合
inv-c2 ： ∀d・(d ∈ dom(ユーザグループ職務構成) ⇒ サイト固有職務(d) ∈ サイトアクセス権[{ユーザグループ職務構成(d)}])

inv-c2 は、ユーザグループ職務が、サイトアクセス権が与えられたサイトに関するものでなければならないことを表す。

４．サブグループ職務

サブグループは、サイトアクセス権が与えられたサイトに関して、サブグループ職務を持つ。ここで、サブグループが持つサブグループ職務は、０個以上任意個であるとする。サブグループ職務は、ただ１つのサブグループと関連付けられる。

inv-d1 ： サブグループ職務構成 ∈ 職務集合 +-> サブグループ集合

inv-d2	:	dom(ユーザグループ職務構成) ∪ dom(サブグループ職務構成) = 職務集合
inv-d3	:	dom(ユーザグループ職務構成) ∩ dom(サブグループ職務構成) = {}
inv-d4	:	∀d・(d ∈ dom(サブグループ職務構成) ⇒ サイト固有職務(d) ∈ サイトアクセス権[{サブグループ構成(サブグループ職務構成(d))}])

inv-d2 は、ユーザグループ職務とサブグループ職務以外には職務がないことを、inv-d3 はユーザグループ職務とサブグループ職務に重なりがないことを表す。inv-d4 は、サブグループ職務が、サイトアクセス権が与えられたサイトに関するものでなければならないことを表す。

５． ユーザグループ属性

ユーザグループには、サイトアクセス権が与えられたサイトが提供するユーザグループ属性が与えられる。一般に、１つのサイトが提供するユーザグループ属性は複数あるが、同時に持ち得るユーザ属性（同時使用可能属性）は制限されている。例えば運営者属性を持つユーザグループは、そのサイトの他の属性を持つことができない。一方、前述のように、当該サイトに対する運営者属性を他のユーザグループが持つことは、仕様上は許される。ユーザグループ属性は、許可するアクセス権を規定する。ここで、一つのユーザ属性が許可するアクセス権は複数あるとともに、一つのアクセス権を許可するユーザグループ属性も複数あり得るとする。

inv-e1	:	ユーザグループ属性 ∈ ユーザグループ集合 <-> 属性集合
inv-e2	:	∀ug・(ug ∈ dom(ユーザグループ属性) ⇒ ユーザグループ属性[{ug}] ⊆ サイト固有属性~[サイトアクセス権[{ug}]])
inv-e3	:	∀s,ug・(s ∈ サイト集合 ∧ ug ∈ dom(ユーザグループ属性) ⇒ ∃as・(as ∈ 同時使用可能属性 ∧ (ユーザグループ属性[{ug}] ∩ サイト固有属性~[{s}] ⊆ 同時使用可能属性集合~[{as}]))

inv-e2 は、ユーザグループが持つユーザグループ属性が、サイトアクセス権が与えられたサイトから提供されたものでなければならないことを表す。inv-e3 は、それらのユーザグループ属性が、同時使用可能な属性の組でなければならないことを表す。

６． ユーザグループアクセス権

ユーザグループは、与えられたユーザグループ属性に規定されている範囲内で、アクセス権を持つ。

inv-f1	:	ユーザグループアクセス権 ∈ ユーザグループ集合 <-> 権限集合
inv-f2	:	∀ug・(ug ∈ dom(ユーザグループアクセス権) ⇒ ユーザグループアクセス権[{ug}] ⊆ 属性権限[ユーザグループ属性[{ug}]])

inv-f2 は、ユーザグループアクセス権（外部設計書のサイト内アクセス権に対応）が、

ユーザグループに与えられたユーザグループ属性に規定されたアクセス権の範囲内でなければならないことを表す。

７．　サブグループアクセス権

サブグループは、関連付けられたユーザグループのユーザグループアクセス権の範囲内で、アクセス権を持つ。

inv-g1	：	サブグループアクセス権 ∈ サブグループ集合 <-> 権限集合
inv-g2	：	∀sg・(sg ∈ dom(サブグループアクセス権) ⇒ サブグループアクセス権[{sg}] ⊆ ユーザグループアクセス権[サブグループ構成(sg)])

inv-g2 は、サブグループが持つアクセス権は、そのサブグループに関連付けられたユーザグループのユーザグループアクセス権の範囲内でなければならないことを表す。

８．　ユーザグループ職務アクセス権

ユーザグループ職務は、関連付けられたユーザグループのユーザグループアクセス権の範囲内で、アクセス権を持つ。

inv-h1	：	ユーザグループ職務アクセス権 ∈ 職務集合 <-> 権限集合
inv-h2	：	dom(ユーザグループ職務アクセス権) ⊆ dom(ユーザグループ職務構成)
inv-h3	：	∀ud,pm・(ud ∈ dom(ユーザグループ職務アクセス権) ∧ pm ∈ ユーザグループ職務アクセス権[{ud}] ⇒ サイト固有職務(ud} = サイト固有権限(pm))
inv-h4	：	∀ud・(ud ∈ dom(ユーザグループ職務アクセス権) ⇒ ユーザグループ職務アクセス権[{ud}] ⊆ ユーザグループアクセス権[{ユーザグループ職務構成(ud)}])

inv-h1 は、ユーザグループ職務と職務に与えられるアクセス権の対応を表す。外部設計書には、職務はアクセス権を１つ以上含むことが規定されているが、Role Based Access Control にしたがってアクセス権の剥奪と職務の削除を分離すため、アクセス権を持たない職務も存在するとした。inv-h2 は、ユーザグループ職務アクセス権を持ち得るのはユーザグループ直下の職務に限られることを表す。inv-h3 は、ユーザグループ職務に与えられるアクセス権は、その職務に関わるサイトのものでなければならないことを表す。inv-h4 は、ユーザグループ職務アクセス権は、ユーザグループアクセス権の範囲内でなければならないことを表す。

９．　サブグループ職務アクセス権

サブグループ職務は、関連付けられたサブグループのサブグループアクセス権の範囲内で、アクセス権を持つ。

inv-i1	：	サブグループ職務アクセス権 ∈ 職務集合 <-> 権限集合
inv-i2	：	dom(サブグループ職務アクセス権) ⊆ dom(サブグループ職務構成)

inv-i3	：	∀sd,pm・(sd ∈ dom(サブグループ職務アクセス権) ∧ pm ∈ サブグループ職務アクセス権[{sd}] ⇒ サイト固有職務(sd) = サイト固有権限(pm))
inv-i4	：	∀sd・(sd ∈ dom(サブグループ職務アクセス権) ⇒ サブグループ職務アクセス権[{sd}] ⊆ サブグループアクセス権[{サブグループ職務構成(sd)}])

inv-i1 は、サブグループ職務と職務に与えられるアクセス権の対応を表し、inv-h1 と同様に、アクセス権を持たない職務も存在するとした。inv-i2 は、サブグループ職務アクセス権を持ち得るのはサブグループに関連付けられた職務に限られることを表す。inv-i3 は、サブグループ職務に与えられるアクセス権は、その職務に関わるサイトのものでなければならないことを表す。inv-i4 は、サブグループ職務アクセス権は、サブグループのアクセス権の範囲内でなければならないことを表す。

7.3.3.2　アクセス権設定／剥奪処理

本実験に関わるアクセス権設定／剥奪処理と、その実行権限を持つ管理者の一覧を表 7-3 に示す。なお、本実験の対象はこれらの処理により 7.3.3.1 節のアクセス権階層を満たすアクセス権が設定されることの検証であり、これらの処理自身に対する実行権限の管理は対象ではない。

表 7-3　アクセス権設定／剥奪処理

処理	管理者
サイトアクセス権設定／剥奪	システム管理者
ユーザグループ属性設定／剥奪	サイト管理者
ユーザグループアクセス権設定／剥奪	サイト管理者
サブグループ設定	ユーザグループ管理者
サブグループアクセス権設定／剥奪	ユーザグループ管理者
ユーザグループ職務設定／削除	ユーザグループ管理者
ユーザグループ職務アクセス権設定／剥奪	ユーザグループ管理者
サブグループ職務設定／削除	サブグループ管理者
サブグループ職務アクセス権設定／剥奪	サブグループ管理者

以下では、典型的な処理について Event-B による記述を枠内に示す。簡単のため、下記の Event-B 記述ではイベントの中核部分のみを掲載する。

１．サイトアクセス権剥奪処理

ユーザグループからサイトアクセス権を剥奪する場合、当該ユーザグループに関する以下

の情報を更新する。

① サイトアクセス権
② ユーザグループ属性
③ ユーザグループアクセス権
④ サブグループアクセス権
⑤ ユーザグループ職務アクセス権
⑥ サブグループ職務アクセス権

```
ANY
        ユーザグループ
        サイト
WHERE
  grd1  :  ユーザグループ |-> サイト ∈ サイトアクセス権
THEN
  act1  :  サイトアクセス権 := サイトアクセス権 ¥ {ユーザグループ |-> サイト}

  act2  :  職務集合 := 職務集合 ¥ ((ユーザグループ職務構成~[{ユーザグループ}] ∪
           サブグループ職務構成~[サブグループ構成~[{ユーザグループ}]]) ∩ サイト
           固有職務~[{サイト}])

  act3  :  ユーザグループ職務構成 := (ユーザグループ職務構成~[{ユーザグループ}]
           ∩ サイト固有職務~[{サイト}]) <<| ユーザグループ職務構成

  act4  :  サブグループ職務構成:= (サブグループ職務構成~[サブグループ構成~[{ユー
           ザグループ}]] ∩ サイト固有職務~[{サイト}]) <<| サブグループ職務構成

  act5  :  ユーザグループ属性 := ({ユーザグループ} <<| ユーザグループ属性) ∪
           (({ユーザグループ} <| ユーザグループ属性) |>> サイト固有属性~[{サイ
           ト}])

  act6  :  ユーザグループアクセス権 := ({ユーザグループ} <<| ユーザグループアクセ
           ス権) ∪ (({ユーザグループ} <| ユーザグループアクセス権) |>> サイト固有
           権限~[{サイト}])
```

```
  act7  :  サブグループアクセス権 := (サブグループ構成~[{ユーザグループ}] <<| サブ
           グループアクセス権) ∪ ((サブグループ構成~[{ユーザグループ}] <| サブグ
           ループアクセス権) |>> サイト固有権限~[{サイト}])

  act8  :  ユーザグループ職務アクセス権 := (ユーザグループ職務構成~[{ユーザグルー
           プ}] ∩ サイト固有職務~[{サイト}]) <<| ユーザグループ職務アクセス権

  act9  :  サブグループ職務アクセス権 := (サブグループ職務構成~[サブグループ構成
           ~[{ユーザグループ}]] ∩ サイト固有職務~[{サイト}]) <<| サブグループ職
           務アクセス権
END
```

サイトアクセス権はユーザグループが持つ属性およびアクセス権の根本に関わるため、サイトアクセス権剥奪の影響は多岐に及ぶ。これらのアクセス権更新の結果がアクセス権階層に違反しないことは、形式検証によって確認できる。

２．ユーザグループ属性剥奪処理

ユーザグループ属性を剥奪する場合、当該ユーザグループ、サイトに関する以下の情報を更新する。

① ユーザグループ属性
② ユーザグループアクセス権
③ サブグループアクセス権
④ ユーザグループ職務アクセス権
⑤ サブグループ職務アクセス権

```
ANY
           属性
           ユーザグループ
           サイト
WHERE
  grd1  :  ユーザグループ ∈ ユーザグループ集合
  grd2  :  サイト ∈ サイト集合
  grd3  :  属性 ∈ ユーザグループ属性[{ユーザグループ}]
THEN
  act1  :  ユーザグループ属性 := ユーザグループ属性 ￥ {ユーザグループ |-> 属性}

  act2  :  ユーザグループアクセス権 := ({ユーザグループ} <<| ユーザグループアクセ
```

```
        ス権) ∪ ({ユーザグループ} × (属性権限[ユーザグループ属性[{ユーザグループ}] ¥ {属性}]))

  act3  :  サブグループアクセス権 := (サブグループ構成~[{ユーザグループ}] <<| サブグループアクセス権) ∪ ((サブグループ構成~[{ユーザグループ}] <| サブグループアクセス権) |> 属性権限[ユーザグループ属性[{ユーザグループ}] ¥ {属性}])

  act4  :  ユーザグループ職務アクセス権 := (ユーザグループ職務構成~[{ユーザグループ}] <<| ユーザグループ職務アクセス権) ∪ ((ユーザグループ職務構成~[{ユーザグループ}] <| ユーザグループ職務アクセス権) |> 属性権限[ユーザグループ属性[{ユーザグループ}] ¥ {属性}])

  act5  :  サブグループ職務アクセス権 := (サブグループ職務構成~[サブグループ構成~[{ユーザグループ}]] <<| サブグループ職務アクセス権) ∪ ((サブグループ職務構成~[サブグループ構成~[{ユーザグループ}]] <| サブグループ職務アクセス権) |> 属性権限[ユーザグループ属性[{ユーザグループ}] ¥ {属性}])
END
```

act1 **はユーザグループ属性の削除、** act2 **と** act3 **はそれにともなうユーザグループアクセス権とサブグループアクセス権の削除である。再構成した仕様ではユーザグループアクセス権をユーザグループ属性の下位に置いたので、整合性を保つためこれらの処理を付加した。**act4 **と** act5 **は職務アクセス権の変更であり、剥奪されたユーザグループ属性に関わり他のユーザグループ属性によってリカバーされないアクセス権のみを削除している。**

３．ユーザグループアクセス権設定処理

入力された内容により、ユーザグループアクセス権を更新する。

```
ANY
        アクセス権
        サイト
        ユーザグループ
WHERE
  grd1  :  ユーザグループ ∈ ユーザグループ集合
  grd2  :  サイト ∈ サイトアクセス権[{ユーザグループ}]
  grd3  :  アクセス権 ∈ 属性権限[(サイト固有属性~[{サイト}] ∩ ユーザグループ属性[{ユーザグループ}])]
THEN
```

```
  act1  :  ユーザグループアクセス権 := ユーザグループアクセス権 ∪ {ユーザグループ |-> アクセス権}
END
```

再構成した形式記述ではユーザグループアクセス権をユーザグループ属性の下位に置いたことにより、ユーザグループには既にユーザグループ属性が与えられて、その範囲内でユーザグループアクセス権を設定する。設定するアクセス権が許容されるものであるかを grd3 で確認し、条件が満たされれば act1 でユーザグループアクセス権を更新する。

7.3.3.3 形式検証

本実験では、アクセス権階層に違反するアクセス権が、アクセス権設定／剥奪処理によって付与されないことを、形式検証によって確認した。アクセス権管理の点から見た主な確認事項と、それを表す 7.3.3.1 節の不変条件との対応を、表 7-4 に示す。7.3.3.2 節で列挙したアクセス権設定／剥奪処理の全てについてこれらの事項を厳密に確認することは人手では困難だが、不変条件として記述し形式検証を利用したことで、効率的に確認できた。

表 7-4 検証内容

確認事項	不変条件
ユーザグループ属性はサイトアクセス権を持つサイトの属性のサブセットであること	inv-e2
ユーザグループ属性は同時使用可能属性集合のサブセットであること	inv-e3
ユーザグループアクセス権はユーザグループ属性によって与えられる権限のサブセットであること	inv-f2
サブグループアクセス権はユーザグループアクセス権のサブセットであること	inv-g2
ユーザグループ職務アクセス権はユーザグループアクセス権のサブセットであること	inv-h4
サブグループ職務アクセス権はサブグループアクセス権のサブセットであること	inv-i4

Event-B では、イベントによる状態変化が起こっても不変条件が常に満たされることを、論理的に検証する。Event-B の代表的な支援ツールである RODIN Platform では、与えた形式記述から、状態変化後にも不変条件が満たされることを示すために証明しなければならない論理式（証明課題）を自動生成する。形式検証者は、RODIN Platform に組み込まれた定理証明ツールの支援のもと証明課題を証明するが、7.4.1 節に示すように多くの証明課題は定

理証明ツールが自動的に証明する。自動証明に失敗する証明課題は、場合分けなどの操作を形式検証者がツールに指示することで証明できるケースが多い。また、自動証明に失敗する証明課題には、形式記述に誤りがあって論理的に証明できないケースもある。証明過程を追跡することで設計のどこに誤りがあるかを分析できる点も、自然言語の設計書と比較した場合の大きなアドバンテージである。

7.4 実験結果

7.4.1 工数

本実験において、各作業にかけた工数を表 7-5、表 7-6 に示す。

表 7-5 ステップ１ 設計書の理解

作業名	工数	工数／合計
Event-B 要素の抽出	11.0 人 H	59%
形式記述の作成	4.5 人 H	24%
形式記述の検証	3.0 人 H	17%
合計	18.5 人 H	100%

表 7-6 ステップ２ 設計書の再構成

作業名	工数	工数／合計
Event-B 要素の抽出	7.0 人 H	26%
リファインメント戦略の立案	4.0 人 H	15%
形式記述の作成	11.0 人 H	41%
形式記述の検証	1.0 人 H	4%
形式記述の修正	4.0 人 H	15%
合計	27.0 人 H	100%

実験では、Event-B のツール RODIN Platform を、記述した仕様を保存するのと同時に証明課題の生成と自動証明を行う設定で使用した。このため、形式記述の検証の工数は、作業者による対話証明に要した時間である。証明課題の生成と自動証明に要した時間は、作業者による対話証明操作と比較して十分に短かった。なお、自動証明に失敗して対話証明を試みた結果、形式記述の誤りが発見され修正した場合の工数は、形式記述の作成ないし形式記述の修正の工数に算入した。

7.4.2　設計書と成果物

7.4.2.1　規模

本実験で用いた、記述対象の設計書ならびに作成した中間成果物や形式記述の量を表 7-7 に記す。

表 7-7

入出区分	成果物	規模
入力	外部設計書	エンティティ定義１６頁（他に参照２８頁）、画面アクション明細１１頁、確認事項数６、機能数１７
中間 （ステップ１）	形式記述	行数１０５ 証明課題数３２　（内、自動証明数２９）
出力 （ステップ２）	形式記述	行数２５７、リファインメントステップ数９ 証明課題数１８２　（内、自動証明数１５６）

ステップ２で作成した Event-B コンポーネントの概要は、以下である。ここで、証明課題の欄は（証明課題の総数／自動証明できた証明課題数／対話証明した証明課題数）を表す。表中の subject1 から permission4 までの９個の MACHINE は、この順にリファインされている。すなわち、subject1 をリファインしたものが subject2 であり、subject2 をリファインしたものが subject3、subject3 をリファインしたものが attribute1,... と連鎖している。最初の subject1 を第１段として、リファインメントステップ数を９とした。

種別	名称	概要	証明課題
CONTEXT	context1	ユーザグループとサブグループの定義	0/0/0
CONTEXT	context2	サイトの定義	0/0/0
CONTEXT	context3	職務の定義	0/0/0
CONTEXT	context4	属性の定義	0/0/0
CONTEXT	context5	同時使用可能属性の定義	4/4/0
CONTEXT	context6	アクセス権の定義	0/0/0
CONTEXT	context7	属性と権限の対応の定義	1/0/1
MACHINE	subject1	ユーザグループとサブグループの処理と不変条件の記述	3/3/0
MACHINE	subject2	サイトとサイトアクセス権の処理と不変条件の記述	4/4/0
MACHINE	subject3	職務の処理と不変条件の記述	43/36/7

MACHINE	attribute1	ユーザグループ属性の処理と不変条件の記述	10/10/0
MACHINE	attribute2	同時使用可能属性の処理と不変条件の記述	5/1/4
MACHINE	permission1	ユーザグループアクセス権の処理と不変条件の記述	15/13/2
MACHINE	permission2	サブグループアクセス権の処理と不変条件の記述	17/14/3
MACHINE	permission3	ユーザグループ職務のアクセス権の処理と不変条件の記述	38/34/4
MACHINE	permission4	サブグループ職務のアクセス権の処理と不変条件の記述	42/37/5

一方、入力となる外部設計書と、出力となるステップ２の形式記述の対応において、次の結果が得られる。

・ 記述について

１機能あたりの形式記述行数＝２５７[行]／１７[機能]＝１５．１（行）

・ 検証について

１機能あたりの証明課題数＝１８２[個]／１７[機能]＝１０．７（個）

7.4.2.2 作業効率

記述対象とした外部設計書 1 ページあたりの工数は、下記のようになった。ここで、工数はステップ１とステップ２の合計とした。

作業効率 ＝ ４５．５[人 H]／２７[ページ] ＝ １．７[人 H／ページ]

ただし、本実験では外部設計書の設計内容に追加と削除を加えて形式記述を作成しているうえ、外部設計書の解読に関わる工数が含まれていることから、上記は参考値にすぎない。参照した外部設計書も含めた作業効率は、下記になる。

作業効率 ＝ ４５．５[人 H]／５５[ページ] ＝ ０．８３[人 H／ページ]

一方、１機能あたりの工数は次である。

作業効率（機能数ベース）＝ ４５．５[人 H]／１７[機数] ＝ ２．７[人 H／機能]

作業者が適切なドメイン知識を持っている場合を想定して、上記から外部設計書の解読に関わる Event-B 要素の抽出の工数を除外し、形式記述と検証の工数について作業効率を計算すると、次のようになる。

作業効率（機能数ベース）＝ ２７．５[人 H]／１７[機能] ＝ １．６[人 H／機能]

ステップ１とステップ２を通じて、自動証明できた証明課題の割合は下記であった。

証明の自動化率　＝　１８５[自動証明]／２１４[証明課題]＊１００　＝　８６（%）

１つの証明に要する工数は次のようになる。7.4.1 節に述べたように自動証明には工数がかかっておらず、工数は作業者が行った対話証明の工数である。単位は人・分とした。

証明の作業効率　＝　４[人 H]／２１４[証明課題]　＝　１．１[人・分]

ステップ2に限定すると、ステップ1を通じて記述内容が整理されたことにより、１つの証明に要する工数はさらに減少している。

証明の作業効率　＝　１[人 H]／１８２[証明課題]　＝　０．３３[人・分]

7.5 考察

7.5.1 仕様記述について

本実証実験では、対象システムが提供するアクセス権管理機構が正しく設計されていることを、形式手法を使って確認する作業を行った。7.4.2.1 節に示したように形式記述の行数は、アクセス権階層とアクセス権設定／剥奪処理を合わせて 257 行であり、問題の規模は小さい。一方、7.3.3 節に示したようにアクセス権階層は複雑であり、これにともなってアクセス権設定／剥奪処理の記述も複雑になっており、問題の複雑度は高い。

実験では、外部設計書の対象部分について形式検証に耐えうる設計内容を抽出するとともに足りない情報を補い、必要な場合には外部設計書と大きく乖離しない範囲で変更を加えた。外部設計書と同等の設計内容を形式仕様言語を使って再構成するのが基本方針であり、形式仕様言語で表現し易い部分だけを抜き出すことは避けた。このため、得られた記述が不必要に複雑なものになっていることも事実であり、この点を考慮すると機能数ベースの作業効率 2.7 人 H は悪いとは言えない。

実験の結果、外部設計書の対象部分を表す上で、Event-B の記述力が不足することはなかった。本実験ではアクセス権階層とアクセス権設定／剥奪処理を記述し、それらの整合性を検証しているので、どちらかの記述に不備があれば整合性検証に支障が生じる。必要十分な検証が行えていることから、本実験の範囲では、Event-B は設計書を記述するための十分な記述力を持っていると結論できる。

7.3.3.2 節に例を示したように、日本語の記述と比べて Event-B の記述の方が詳細である。7.3.3.2 節では外部設計書を引用してはいないが、日本語の記述は基本的に同様である。これは、日本語の記述が「どの項目を」更新するかを規定しているのに対し、Event-B では「どの部分を」「どのように」更新するかまでも規定していることによる。これらの情報はアクセス権階層で規定されており、両者を同等の詳細度で記述することで、アクセス権階層と

アクセス権設定／剥奪処理の整合性を検証することが可能になる。なお、Event-B 記述が詳細なのは設計を正確に表すためであり、プログラムのように計算機で実行させるための詳細さとは異なる点に注意する必要がある。開発工程上の判断によって「どの項目を」更新するかのみを Event-B で記述することも可能であり、形式記述は必ずプログラムのように詳細に書かなければならない、ということではない。一方、日本語を使って Event-B 記述と同等の記述を行うことも可能であろうが、多くの場合は記述量が膨大になり、設計書としては理解し難く機械的な検証もできないので、メリットは少ないと思われる。

ここで、日本語は外部設計書を書くための十分な記述力を持っているか、という視点からの議論も必要であろう。外部設計書の対象部分は日本語と図表で記述されているが、対象ドメインの知識を持たない作業者には難解であった。7.4.1 節に示すように、全工数の 40% を Event-B 要素の抽出、すなわち外部設計書の理解、に費やしている。結局、作業者は設計書の部分部分を形式記述し、検証を通じて内容を確認することで要素間の関連を把握した（ステップ１）。このことは、外部設計書の不備を意味するものではない。形式仕様言語は豊かな内容を表現するコンパクトな記法を持っており、日本語を使って形式記述と同等の正確さで仕様を書こうとすると、かえって理解困難な記述になってしまう危険性が高い。日本語の設計書は、ドメイン知識を共有する開発者の間で、認識を確認するための手がかりとして利用すれば有効であろう。

一つの方向性としては、ドメイン知識を持つ開発者が設計および実装作業を行う際には形式仕様言語を用い、開発者間の認識の確認や外部への説明には日本語や図表を用いる、という使い分けも考えられる。この場合、開発は形式仕様言語を使って行うことになるので、日本語や図表で設計書を作成するコストは大きく減少し、結果的にトータルコストも減少する可能性がある。

7.5.2 形式検証について

本実験では、7.4.2 節に示したように、182 項目について論理的な証明を行った。この中には、7.3.3.3 節に示した６項目が全てのイベントによる状態変化後にも成り立つことの、論理的な証明も含まれている。設計内容の整合性を検証するために必要な証明課題が証明できたという点では、設計検証において定理証明手法が一定の役割を果たし得ることを示す結果である、と考えることができる。

7.1 節で述べたように、本実験はセキュリティ機構設計への形式手法の適用と評価を目的とするものではない。形式検証の観点からは、不変条件として書かれたデータ制約が全ての処理について常に満たされることを検証しているに過ぎず、セキュリティ固有の問題に特化した特殊な検証を行うものではない。一般に、計算機システムが扱うデータは相互に関連しており、処理によって関連が壊れないことが求められる場合は多い。しかし、データ間の関連が記述されることは少なく、設計者は頭の中にあるデータ間の関連を壊さない

よう処理を設計していることが多い。頭の中にあるデータ間の関連を処理が壊さないことの確認が、設計レビューとなる。形式検証は、このような設計レビューを機械化する点において、様々な分野に展開できる可能性を持つ。

定理証明ツールによる証明の自動化率は、7.4.2.2 節のように、86%であった。残りの 14%の証明は対話的に行ったが、基本的には形式検証者が場合分けなどの簡単な指示を与えるだけで、定理証明ツールが証明を進行させている。証明が容易であった理由として考えられるのは、多くのイベントは Event-B の関係（ないし関数）として表されたデータの特定の部分を更新するだけであり、さらに更新されない部分と更新される部分が互いに依存しないため、前者と後者を分けて証明できたことである。ただし、例えばサイトアクセス権の剥奪処理のように、特定のユーザグループについて、特定のサイトに関するアクセス権を剥奪する、と２段階の場合分けが起こるので、問題が簡単であったわけではない。なお、証明を容易にする観点からは、ユーザグループをサイト毎（１つのユーザグループは１つのサイトのみに属する）として利用者が指定された複数のユーザグループに属する等の改変も考えられるが、本実験では外部設計書からの乖離を最小限にすることを優先した。

実験では、定理証明ツールが自動証明できなかった 14%（29 個）の証明課題を対話証明するのに４[人 H]かかっており、対話証明１つあたりの工数は 8.3[人・分]であった。自動証明には工数がかかっていないので、証明課題１つあたりの証明工数は 1.1[人・分]となった。証明の難しさは形式記述の書き方にも依存するので一概には言えないが、設計書を再構成したステップ２に限れば 182 個の証明を平均 0.33[人・分]で終えていることから、証明のコストは現実的な範囲に収まり得ると結論できる。

7.5.3 リファインメントについて

7.4.2.1 節に示したように、本実験ではステップ２の形式記述を７段階の CONTEXT 記述と９段階の MACHINE 記述から構成した。各段階で何を書くかはリファインメント戦略の立案作業で検討したが、ステップ１で記述すべき要素の関連を分析済であったこともあり、記述は大きな手戻りなく円滑に進行した。リファインメントを有効に使うことは、各段階の記述の複雑さを減らすうえで、重要であった。証明課題の数が各段階に分散している点からも、リファインメントの効果がうかがえる。ただし、職務に関するコンポーネント subject3，permission3，permission4 では証明課題の数が突出しており、関連の分析が不十分であったと考えられる。

一般に、リファインメントは抽象的な仕様から具体的なプログラムを段階的に導出する手段として用いられ、Event-B のように仕様記述と検証をステップ・バイ・ステップで行う手段として用いるのは珍しい。このようなリファインメントは、使い方によっては高い効果が得られる一方、どのように使えば良いかはまだ研究途上にあるように思われる。本実験で対象としたアクセス権階層は、リファインメントの点からは比較的素直な構造を持っ

ており、階層の上から下へと記述を進めてゆけば良かった。一般には、記述すべき要素を俯瞰してリファインメント戦略を立てる必要があり、この点でも実開発ではドメインの専門家との共同作業が必要になる。

7.6 まとめ

本実験では、外部設計書に記載されたアクセス権階層とアクセス権設定／剥奪処理の両者を形式仕様言語を使って記述するとともに、後者によって前者に違反するアクセス権が設定されないことを、形式検証によって示す作業を行った。本実験では、この作業を通じて以下を確認することができた。

- 対象とする設計書の必要な部分が、形式仕様言語によって表現できる
- 形式記述の整合性を確認するために必要な検証が、十分な厳密性のもとに行える
- 上記の記述と検証が、一般的な計算機環境の下で妥当な作業時間内に行える

一方、本実験では作業者がドメイン知識を持たなかったため、無駄な作業も多く発生した。ドメイン知識を持つ開発者が形式記述を作成する、あるいは形式記述者がドメイン知識を持つ開発者と共同で作業することにより、より有効に形式手法を活用できる可能性があることも分かった。

参考文献

1. Dependable Software Forum . **形式手法適用手順（**Event-B **編）【リリース版】**. Copyright © 2011 Dependable Software Forum, 2011.

2. **独立行政法人情報処理推進機構ソフトウェア・エンジニアリング・センター**. **機能要件の合意形成ガイド**. (**オンライン**) 2010 **年**. http://sec.ipa.go.jp/reports/20100331.html.

3. Dependable Software Forum. **形式手法活用適用手順（**SPIN **編）【リリース版】**. Copyright © 2011 Dependable Software Forum, 2011.

4. Dependable Software Forum . **形式手法イディオム集（**SPIN **編）【リリース版】**. Copyright © 2011 Dependable Software Forum , 2011.

5. Dependable Software Forum. **形式手法適用手順（**VDM++**編）【リリース版】**. Copyright © 2011 Dependable Software Forum, 2011.

6. Dependable Software Forum. **形式手法イディオム集（**Event-B **編）【リリース版】**. Copyright © 2011 Dependable Software Forum , 2011.