

実務家のための形式手法

厳密な仕様記述を志すための形式手法入門 第二版

実践法：モデル化の手順と事例

VDMを想定した**モデル化の手順**の概要を知り、適用事例を通してその理解を進めるモジュール

■ 事前知識・経験

- 形式手法(VDM)導入計画もしくは関心
- プログラミング、集合論の**基礎**
- ソフトウェア開発と形式手法導入に関する基礎知識
- VDM++ と VDMTools の基礎知識

■ 学習目標

- モデル化手順の概要を知る
- 手順の適用イメージを持つ

■ 主な学習項目

- モデル化の手順の概要
- 仕様フレームワークの概要
- 事例を通した手順のポイントの確認

1. VDMを使った開発手順の概要

- 仕様記述
- 仕様フレームワーク
 - 要求辞書階層の例
 - 業務論理階層の例
 - 要求辞書としてのVDMソース
- 仕様ライブラリ
- 仕様テスト
 - 組合せテスト
 - 回帰テスト
- プログラミング
- プログラムのテスト
- 管理

2. 仕様記述

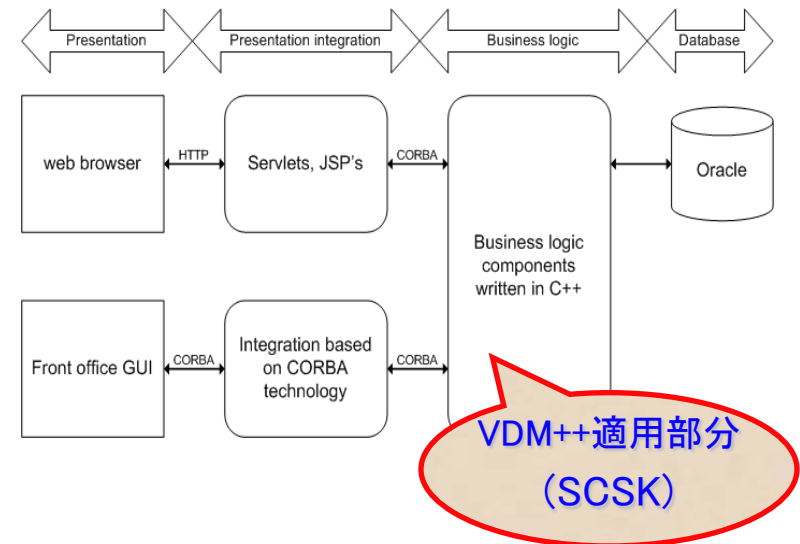
■ 仕様フレームワーク

● 目的

- 再利用性と保守性の向上
- VDM技術レベルと仕様の階層を対応
- 実装側フレームワーク(Gofo)と対応(SCSK)
- DBインタフェースの隠蔽(SCSK)

● フレームワーク概要

- GUIを捨象
- 対象は UseCaseレベル仕様(SCSK)
- 対象は API 仕様(FeliCa)



■ 仕様ライブラリ作成

- 暦、日時、列、文字列、待ち行列、ハッシュ表など

3. 仕様フレームワーク

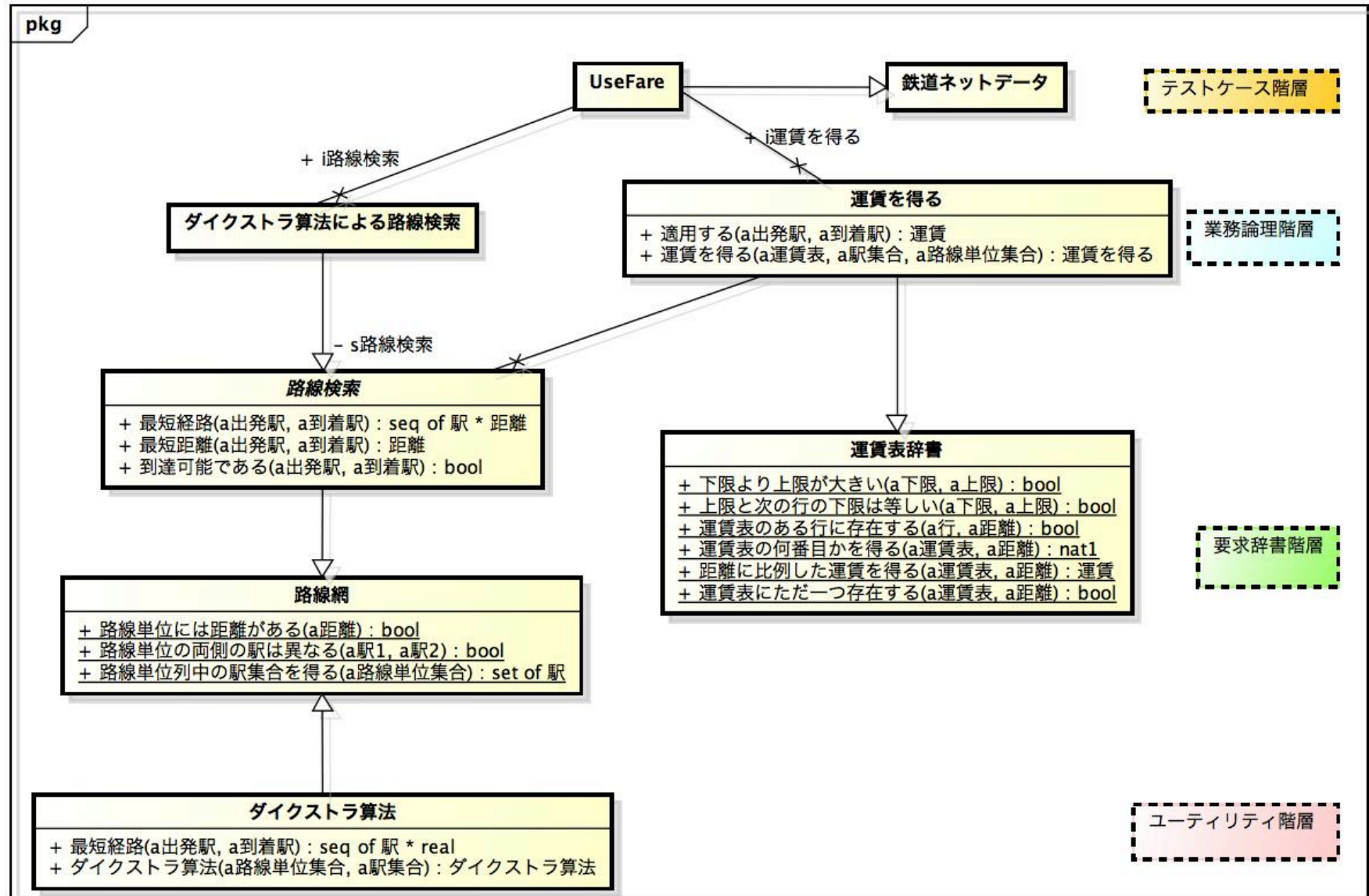
■ 証券業務の例 (SCSK)

階層	充足する記述領域	VDM技量
テストケース	回帰テスト	初級
シナリオ	事後条件	初級
業務論理	業務アプリケーション	初級
要求辞書	業務知識兼用語辞書	中級
仮想DBアクセス応用	登録,抽出,削除,訂正	上級
仮想DBアクセス基本	undo, old valueアクセス, 差分	上級
基本レコード構造	必要データ	初級

■ 運賃計算の例

階層	充足する記述領域	VDM技量
テストケース	回帰テスト	初級
業務論理	業務アプリケーション	初級
要求辞書	業務知識兼用語辞書	中級
ユーティリティ	共通機能	上級

運賃計算クラス図とフレームワークの関係



class 運賃表辞書 運賃表示辞書クラス

types

public 運賃 = nat;

public 行 :: レコードの定義

f下限 : 路線網`距離

f上限 : 路線網`距離

f運賃 :- 運賃; f~, w~, s~などは、私的な命名規則
それぞれ、欄(field)、局所名(work)、インスタンス変数(status)

public 運賃表 = seq of 行

inv w運賃表 == 不変条件 w運賃表の要素全てについて、下記の条件式が成り立つ

forall i, j in set inds w運賃表 & 全称限量式(すべての~について~であればtrue)

下限より上限が大きい(w運賃表(i).f下限, w運賃表(i).f上限) and

j = i + 1 => 上限と次の行の下限は等しい(w運賃表(i).f上限, w運賃表(j).f下限);

運賃表の例 = [
mk_運賃表辞書`行(0,1,150),
mk_運賃表辞書`行(1,3,160),
mk_運賃表辞書`行(3,6,190),...
mk_運賃表辞書`行(15,v最大値,300)
]

functions

関数定義

public static 距離に比例した運賃を得る : 運賃表 * 路線網`距離 -> 運賃

引数と返り値の型

距離に比例した運賃を得る(a運賃表, a距離) ==

let n = 運賃表の何番目かを得る(a運賃表, a距離) in a運賃表(n).f運賃

pre

事前条件

運賃表にただ一つ存在する(a運賃表, a距離)

post

事後条件

let n = 運賃表の何番目かを得る(a運賃表, a距離) in

RESULT = a運賃表(n).f運賃;

RESULTは返り値を表す

public static 運賃表にただ一つ存在する : 運賃表 * 路線網`距離 -> bool

運賃表にただ一つ存在する(a運賃表, a距離) ==

exists1 i in set inds a運賃表 & 運賃表のある行に存在する(a運賃表(i), a距離);

...

添字

end 運賃表辞書

class 運賃を得る is subclass of 運賃表辞書

instance variables

public s運賃表 : 運賃表 := []; インスタンス変数

operations

public 運賃を得る : 運賃表 ==> 運賃を得る

運賃を得る(a運賃表) == (
 s運賃表 := a運賃表;

 return self 自身(インスタンス)を返す
);

public 適用する : 路線網`距離 ==> 運賃

適用する(a距離) ==

 return 距離に比例した運賃を得る(s運賃表, a距離);

end 運賃を得る

■ 要求辞書

- 要求辞書 = 用語辞書 + 述語
- VDMソースは、それ自体が、要求辞書になっている

4. 仕様ライブラリ

- 以下のライブラリ3000行ほどを、あらかじめ作成
 - 文字、文字列ライブラリ
 - VDM++ には文字順がないため、作成
 - 暦、日時、期間ライブラリ
 - 通常のソフトウェア作成に必要なほとんどの機能を持ったもの
 - 列ライブラリ
 - 合計、平均、ソート、その他の列の処理
 - 待ち行列、ハッシュ表ライブラリ
 - 数値計算関連ライブラリ
 - 実数の桁数、微分、ニュートン法など
 - その他、業務に関連したライブラリ

5. 仕様テスト

- VDMToolsで、静的チェック
 - 構文チェック、型チェック
 - 証明課題生成を行ってレビュー
- VDMToolsで、動的チェック
 - VDM++インタプリタとデバッガで実行可能仕様をテスト
 - 動的に、不変条件・事前条件・事後条件をチェック
 - 組合せテスト機能を利用して、組合せテストを実行(SCSK)
 - 回帰テスト支援ライブラリを改良して回帰テストを実行
 - － 仕様修正に対して効果大
 - コードカバレッジ(網羅性検査)機能で、テスト達成率を測定
 - － C0(命令)レベルで85%(FeliCa)から95%(SCSK)を達成
 - 検出した矛盾点を、要求定義に反映
- VDM仕様に基づいて、ランダム評価ツールを作成し、数億件のテストケースを生成し、動的チェック(FeliCa)
 - 仕様エミュレータを内蔵
 - 実装に対し、ランダムにコマンドを送信し、応答が正しいか確認

- 指定されたテストパターンから、テストケースの組合せを生成し、テストする
- エラーになったものと、同じパターンのテストケースは実行せず、テストの実行効率を上げている

```
class UseUniqueNumber
instance variables
sUN : 発番者 := new 発番者();
```

traces

S1 :

let n in set {1,2,3,4} in sUN.発番する(n){10,12}

S2 :

let n in set {2} in sUN.発番する(n)+

10回から12回繰り返す

S3 :

let n in set {3} in sUN.発番する(n)*

1回から既定回数繰り返す

S4 :

let n in set {4} in sUN.発番する(n){10002}

10002回繰り返す

```
end UseUniqueNumber
```

組合せテスト 実行例

Combinatorial Testing - UniqNumber/UseUniqueNumber.vdmpp - Overture Tools - /Users/sahara/workspace

VDM Explorer

- fare
- Fare0
- GC
- LibraryByDSFeventb
- LinearInterpolation
- Saturation
- UniqNum
 - generated
 - Character.vdmpp
 - CommonDefinition.vdmpp
 - Function.vdmpp
 - Integer.vdmpp
 - IO.vdmpp
 - Makefile
 - Object.vdmpp
 - Real.vdmpp
 - Sequence.vdmpp
 - String.vdmpp
 - test.script
 - TestCase.vdmpp
 - TestDriver.vdmpp
 - TestLogger.vdmpp
 - UniqueNumber.opt
 - UniqueNumber.pdf
 - UniqueNumber.prj
 - UniqueNumber.tex
 - UniqueNumber.vdmpp
 - UniqueNumber.wgm
 - UniqueNumberT.vdmpp
 - UseUniqueNumber.vdmpp
 - VDMTESTLOG.TXT
 - UniqNumber
 - generated
 - Character.vpp
 - CommonDefinition.vpp
 - Integer.vpp
 - Sequence.vpp
 - String.vpp
 - UniqueNumber.vpp
 - UseUniqueNumber.vdmpp

UniqueNumber.vpp

```
1 class UseUniqueNumber
2
3 instance variables
4 sUN : 発番者 := new 発番者();
5
6 traces
7
8 S1 :
9   let n in set {1,2,3,4} in sUN.発番する(n){10,12}
10
11 S2 :
12   let n in set {2} in sUN.発番する(n)+
13
14 S3 :
15   let n in set {3} in sUN.発番する(n)*
16
17 S4 :
18   let n in set {4} in sUN.発番する(n){10002}
19
20 end UseUniqueNumber
```

CT Overview

Refresh

- UniqNumber
 - UseUniqueNumber
 - S1 (12 skipped 1)
 - Test 000001
 - Test 000002
 - Test 000003
 - Test 000004
 - Test 000005
 - Test 000006
 - Test 000007
 - Test 000008
 - Test 000009
 - Test 000010
 - Test 000011
 - Test 000012
 - S2 (5)
 - Test 000001
 - Test 000002
 - Test 000003
 - Test 000004
 - Test 000005
 - S3 (6)
 - Test 000001
 - Test 000002
 - Test 000003
 - Test 000004
 - Test 000005
 - Test 000006
 - S4 (1)
 - Test 000001

Problem

Trace Test case	Result
sUN.発番する(1)	"7"
sUN.発番する(1)	"8"
sUN.発番する(1)	"9"
sUN.発番する(1)	Error 4036: ERROR statement reached in '発番者' at line 17:3

■ ソフトウェアが退化していないか確認するテスト

- テストケースと結果を記述し、一致するか確認する

回帰テスト記述例

```
...
def w2予約内容 = wシステム.特急券を得る(mk_token(<予約ID01>), w古いクレジットカード)
in assertTrue("test01 古いクレジットカードで特急券を得ることに失敗",
  w2予約内容 = mk_token(<最初の予約内容>));

trap <RuntimeError> with
  print("¥ttest01 テストの意図通り、予約会員証で特急券を得ることに失敗¥n")
in (
  wシステム.クレジットカードを切り替える(w契約, w変更後のクレジットカード);
  def w3予約内容 = wシステム.特急券を得る(mk_token(<予約ID01>), w会員証) in
  assertTrue("¥ttest01 テスト意図に反し、予約会員証で特急券を得ることに成功するが・・・¥n",
    w3予約内容 = mk_token(<最初の予約内容>)
  )
);
...
```

Initializing specification ... done

>> d new TestApp().run()

Start test – 特急券予約モデルの回帰テスト

Start test - TestCaseT0001 通常の予約成功

End test - TestCaseT0001 通常の予約成功

Start test - TestCaseT0002 クレジットカードを切り替えたときの予約成功

End test - TestCaseT0002 クレジットカードを切り替えたときの予約成功

Start test - TestCaseT0003 クレジットカードを切り替え、予約からクレジットカードで特急券を得るが、予約と異なるクレジットカードでは失敗

/Users/sahara/workspace/ExpressReservationExpress/ReservationSystemDomain.vpp, l. 46, c. 2:

Run-Time Error 58: 事前条件の評価結果がfalseです

test01 テスト意図通り、当初のクレジットカードで特急券を得ることに失敗

End test - TestCaseT0003 クレジットカードを切り替え、予約からクレジットカードで特急券を得るが、予約と異なるクレジットカードでは失敗

Start test - TestCaseT0004 クレジットカードを切り替え、予約から予約会員証で特急券を得るが失敗

/Users/sahara/workspace/ExpressReservationExpress/ReservationSystemDomain.vpp, l. 62, c. 2:

Run-Time Error 58: 事前条件の評価結果がfalseです

test01 テストの意図通り、予約会員証で特急券を得ることに失敗

End test - TestCaseT0004 クレジットカードを切り替え、予約から予約会員証で特急券を得るが失敗

End test - 特急券予約モデルの回帰テスト

*** 全回帰テストケース成功。***(no return value)

- 実装用フレームワーク(Gofo)を作成
- VDM++仕様を見て、手作業でプログラム作成
 - VDM++ からツールによる C++ 生成も可能だが
 - 独自のミドルウェアに合わせて調整する時間的余裕が無かった(SCSK)
 - 生成されるプログラムの性能は、恐らく問題ないと予想
 - プロジェクト工数の5%程度の作業に時間を取られなくなかった
 - 生成されたC++プログラムの性能が、十分でなかった(FeliCa)
- 単純な画面生成プログラムを作成し、GUI生成(SCSK)
- SCSKでは、コード生成機能を大幅に改良し、エンタープライズシステム用Java生成フレームワークを開発している。

■ 単体テスト

- **purifyでソースの静的チェック**
 - メモリの破壊可能性などをチェック
- **C++コードカバレッジ機能でテスト達成率測定**
 - **Excelデータからテストケースを生成する簡単なプログラムを作成**
 - **VDM++テストケースを、C++プログラムのExcelの単体テストケースに手作業で変換**
 - **C0レベルで98%カバー**
- **仕様エミュレータを内蔵(FeliCa)**
 - **実装に対し、ランダムにコマンドを送信し、応答が正しいか確認 (FeliCa)**

■ 連結テスト

- 要求定義担当者が、業務知識と Word で作成したユースケースから、テストケース生成(SCSK)

■ 総合テスト

- 評価環境を構築(FeliCa)
 - ICチップファームウェアと、実装と、形式仕様とが、同様の動作をすることを確認
- VDM仕様に基づいて、ランダム評価ツールを作成し、数億件のテストケースを生成し、動的チェック(FeliCa)
- 業務専門家全員で総合テストケース作成 (SCSK)

■ 版管理ツールcvsを使用

- VDM++仕様、ソースプログラム、Wordなどのドキュメント、テストケースなど、全てのリソースを版管理

■ Swiki(Squeak Wiki)サーバーを使って、共同作業(SCSK)

■ 単体テスト以後の、全欠陥をnotesで記録し、追跡(SCSK)

- ISO 9000

9. 対象を如何にモデル化するか？

■ VDM++によるモデル化手順の例

■ 特急券予約の例

- 用語
- モデル化の範囲
- クラス図
- 仕様の階層
- システム
- 回帰テストケース
- 証明課題生成
- 問題考察と修正
- 統計情報

- モデル化の範囲を決める
 - 例えば、「エラー処理の詳細を除くユースケースレベルの要求仕様」
- 仕様を分割統治する
 - 階層化とモジュール化(仕様フレームワークを使う)を行う
- 主に名詞から型を定義する
 - 属性の参照に留まらない機能を持つ型は、クラスとする
- システムが状態を持つ場合、その状態を定義する
- 型や状態の不変条件を記述する
- 主に、述語(動詞)から、関数・操作のシグネチャの概略を描く
- 関数・操作の事前条件・事後条件を記述する(陰仕様作成)
- 静的検証を行う
 - 構文・型・証明課題チェック
- 関数・操作の本体を記述する(陽仕様作成)
- モデルの正当性と妥当性を、動的に検証する
 - インタープリタ、デバッガ
- 要求をレビューし、漏れがないか再検討する

- おサイフケータイ(鉄道会社B)で特急券予約(鉄道会社A)を使っていた
- 会社Cクレジットカードが廃止になり会社Dクレジットカードに変更して下さいとの連絡があった
- 鉄道会社Aサポートセンターに電話した
 - おサイフケータイの登録クレジットカードを変更すれば、2日後には特急券予約が使えますとのこと
 - おサイフケータイでクレジットカードを変更し、2日後に使おうとしたが、特急券予約できなかった
 - 「予約できなかったんですが」「カードを変更したら、特急券予約を新規に契約して下さい」「新規にすると会費がかかるのでは?」「はい」「カード会社の都合で変更するのに、それはおかしいでしょう?」「では、無料にします」
 - 「カード変更前に予約した特急券に引換えようとしたらできないのですが?」「予約会員証で引換できるようになったので、それで引換えて下さい。暗証番号はクレジットカードのものを使って下さい」
- T駅での問答
 - 「会員証で引換えできないですねー。おかしいな。クレジットカードでやってみましょう。駄目ですねー。」「古いクレジットカードでは?」「あ、できましたね。はい、切符です。」

問題領域の用語

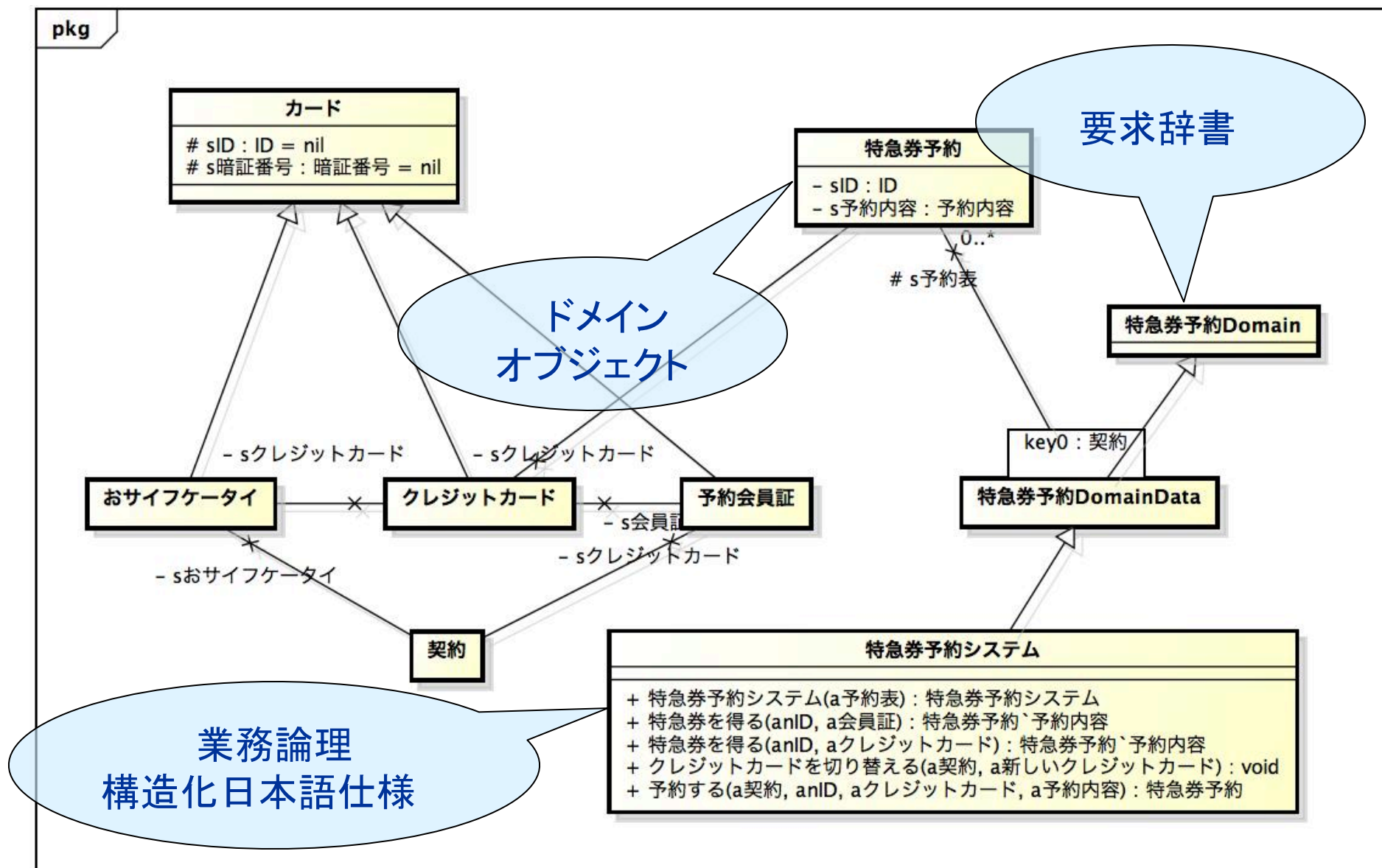
- 鉄道会社A、鉄道会社B
- おサイフケータイ
- 特急券予約
- 予約会員証、ICカード、予約カード
- クレジットカード
 - 会社Cクレジットカード、会社Dクレジットカード
- 特急券

モデル化対象用語

- おサイフケータイ
- 特急券予約
- 予約会員証
- クレジットカード
- 特急券

- 予約する
- 特急券を得る
- クレジットカードを切り替える

特急券予約 クラス図



階層	充足する記述領域	備考
テスト	回帰テスト	
業務論理	業務アプリケーション	構造化日本語仕様
要求辞書	業務知識兼用語辞書	ドメイン・オブジェクトも含まれる
ユーティリティ	共通機能	

```
class
特急券予約システム is subclass of 特急券予約DomainData
...
public
予約する： 契約 * ID * クレジットカード * 特急券予約`予約内容 ==> 特急券予約
予約する(a契約, anID, aクレジットカード, a予約内容) ==
( def w特急券予約 = new 特急券予約(anID, aクレジットカード, a予約内容) in
  (
    if 予約がある契約である(a契約, s予約表)
    then s予約表 := 予約表を更新する(s予約表, a契約, w特急券予約)
    else s予約表 := 予約表に追加する(s予約表, a契約, w特急券予約);
    return w特急券予約
  )
)
post if 予約がある契約である(a契約, s予約表~)
then 予約表が更新されている(s予約表~, a契約, RESULT, s予約表)
else 予約表に追加されている(s予約表~, a契約, RESULT, s予約表);
```

public

特急券を得る : ID * クレジットカード ==> 特急券予約`予約内容

特急券を得る(anID, aクレジットカード) ==

```
( def w予約 = 予約を得る(s予約表, anID, aクレジットカード) in
  return w予約.予約内容を得る()
)
```

```
pre let w予約 = 予約を得る(s予約表, anID, aクレジットカード) in
  aクレジットカード = w予約.クレジットカードを得る();
```

public

特急券を得る : ID * 予約会員証 ==> 特急券予約`予約内容

特急券を得る(anID, a会員証) ==

```
( def w予約 = 予約を得る(s予約表, anID, a会員証) in
  return w予約.予約内容を得る()
)
```

```
pre let w予約 = 予約を得る(s予約表, anID, a会員証) in
  a会員証.クレジットカードを得る() = w予約.クレジットカードを得る();
```

```
class
特急券予約Domain is subclass of 共通定義
types
public 予約表 = map 契約 to set of 特急券予約;
functions
public
  予約を得る : 予約表 * ID -> 特急券予約
  予約を得る(a予約表, anID, a予約会員証) ==
    ( let w予約 in set dunion rng a予約表 be st w予約.IDを得る() = anID in w予約)
  pre exists w予約 in set dunion rng a予約表 & w予約.IDを得る() = anID;

public
  予約表を更新する : 予約表 * 契約 * 特急券予約 -> 予約表
  予約表を更新する(a予約表, a契約, a特急券予約) ==
    a予約表 ++ {a契約 |-> 予約集合に追加する(a予約表(a契約), {a特急券予約})}
  pre a契約 in set dom a予約表
  post 予約表が更新されている(a予約表, a契約, a特急券予約, RESULT);

public
  予約表に追加する : 予約表 * 契約 * 特急券予約 -> 予約表
  予約表に追加する(a予約表, a契約, a特急券予約) ==
    a予約表 munion {a契約 |-> {a特急券予約}}
  pre not 予約がある契約である(a契約, a予約表) and
    forall w契約1 in set dom a予約表, w契約2 in set dom {a契約 |-> {a特急券予約}} &
      w契約1 = w契約2 => a予約表(w契約1) = {a契約 |-> {a特急券予約}}(w契約2)
  post 予約表に追加されている(a予約表, a契約, a特急券予約, RESULT);
```

意図的に事前条件エラーを発生させ <RuntimeError>を起こさせた例

```
def
    wクレジットカード = new クレジットカード(mk_token(<クレジットID01>), mk_token(<クレジットPW01>));
    wおサイフケータイ = new おサイフケータイ(mk_token(<おサイフケータイID01>), wクレジットカード);
    w会員証 = new 予約会員証(mk_token(<会員証ID01>), mk_token(<会員証PW01>), wクレジットカード);
    w契約 = new 契約(wおサイフケータイ, w会員証);
    wシステム = new 特急券予約システム({|->}) ... in
...
wシステム.予約する(w契約, mk_token(<予約ID01>), w古いクレジットカード, mk_token(<最初の予約内容>))
...
trap <RuntimeError> with
    print("¥ttest01 テストの意図通り、予約会員証で特急券を得ることに失敗¥n")
in (
    wシステム.クレジットカードを切り替える(w契約, w変更後のクレジットカード);
    def w3予約内容 = wシステム.特急券を得る(mk_token(<予約ID01>), w会員証)
    in
        assertTrue("¥ttest01 テスト意図に反し、予約会員証で特急券を得ることに成功するが...¥n",
            w3予約内容 = mk_token(<最初の予約内容>)
        )
    );
```

証明課題

選択	クラス	メンバー	位置	指標	型
<input type="radio"/>	特急券予約Domain	予約を得る	function	2	non-emptiness of let be such binding
<input type="radio"/>	特急券予約Domain	予約を得る	function	3	non-emptiness of let be such binding
<input type="radio"/>	特急券予約Domain	予約表に追加されている	function	1	compatible maps
<input checked="" type="radio"/>	特急券予約Domain	予約表が更新されている	function	1	map application
<input type="radio"/>	特急券予約Domain	予約表を更新する	function	1	post condition
<input type="radio"/>	特急券予約Domain	予約表に追加する	function	1	compatible maps

選択

クラス

メンバー

位置

型

選択可能

No

Yes

選択済

(forall a予約表 : 予約表, a契約 : 契約, a特急券予約 : 特急券予約, a更新後予約表 : 予約表 & a契約 in set dom (a予約表))

証明課題で指摘された事項を事前条件に反映

```

public 予約表を更新する : 予約表 * 契約 * 特急券予約 -> 予約表
予約表を更新する(a予約表, a契約, a特急券予約) =
    a予約表 ++ {a契約 |> 予約集合に追加する(a予約表(a契約), {a特急券予約})}
pre
    a契約 in set dom a予約表
post
    予約表が更新されている(a予約表, a契約, a特急券予約, RESULT);
    
```

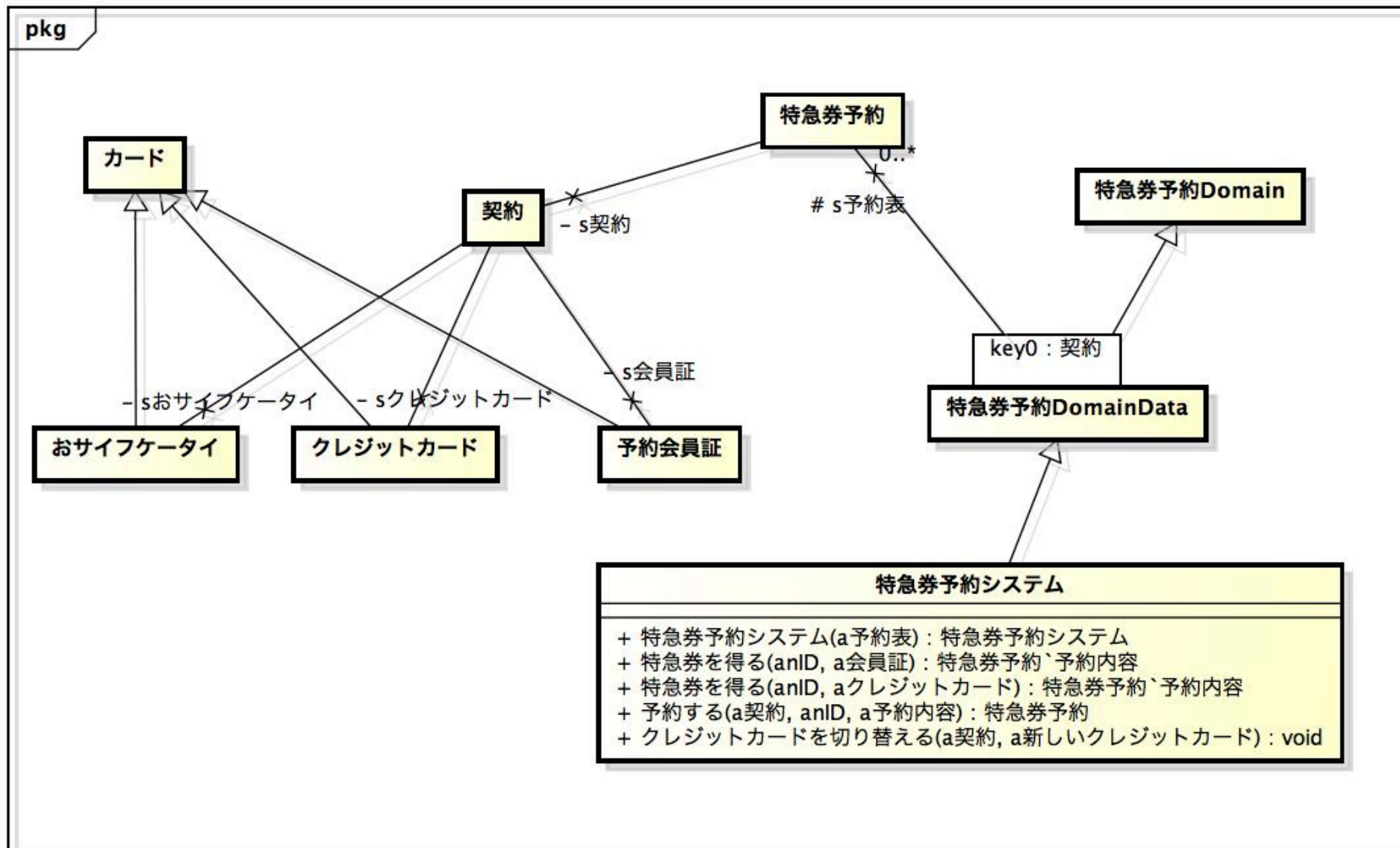
■ 要求仕様モデルの考慮不足

- おサイフケータイのクレジットカードを変更すると、特急券予約を新規に契約せねばならず、予約会員証も新規に作成
 - 変更という概念が無く、ユーザーに不便を強いている
- 個々の予約が、クレジットカードにリンク
 - クレジットカード変更に弱く、ユーザーにも駅員にも分かりにくい
- 予約会員証からクレジットカードにリンク
 - クレジットカード変更に弱く、ユーザーにも駅員にも分かりにくい



- 契約が、クレジットカードや予約会員証とリンクすべき
- 個々の特急券予約は、契約を介してクレジットカード情報に到達すべき

特急券予約 修正後のクラス図



	量	備考
VDM++行数	501行	注釈抜き
工数	3日	モデル作成工数1日
修正工数	1日	モデル修正工数1日

実際に問題を解決した工数より少ない

VDMを想定し、モデル化の手順の概要を知り、適用事例の説明を通して、各自が適用イメージを持てることを目標に以下を学習

- モデル化の手順の概要
- 仕様フレームワークの概要
- 事例を通した手順のポイントの確認

実務家のための形式手法

厳密な仕様記述を志すための形式手法入門

実践法：モデル化の手順と事例

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

上流品質技術部会 人材育成WG(編)

2013年3月 第二版発行

記載されている個々の情報についての著作権及び商標はそれぞれの権利者に帰属するものです
なお、本書の内容は将来予告なしに変更することがあります