エクスプレス予約システムの問題点を推測し、修正した仕様

佐原　伸

SCSK (株)

目次

1 はじめに 4
1.1 本来どうすべきだったのか？ 4

2 特急券予約システムクラス 5
2.1 責任 5
2.2 クラス定義と構成子定義 5
2.3 操作：予約する 5
2.4 操作：特急券を得る 6
2.5 操作：特急券を得る 6
2.6 操作：クレジットカードを切り替える 6

3 共通定義クラス 8
3.1 責任 8

4 特急券予約 Domain クラス 9
4.1 責任 9
4.2 クラス定義 9
4.3 型およびインスタンス変数定義：予約表 9
4.4 関数：予約を得る 9
4.5 関数：予約を得る 9
4.6 関数：予約を得る 10
4.7 関数：予約表を更新する 10
4.8 関数：予約表に追加する 11
4.9 関数：予約がある契約である 11
4.10 関数：契約が存在する 11
4.11 関数：予約集合に追加する 12

5 特急券予約 DomainData クラス 13
5.1 責任 . . . 13
5.2 クラス定義 . . . 13
5.3 操作：予約集合を得る . . . 13

6 特急券予約クラス 14
6.1 責任 . . . 14
6.2 クラス定義 . . . 14
6.3 型定義：予約内容 . . . 14
6.4 インスタンス変数定義：ID, 契約, 予約内容 . . . 14
6.5 構成子 . . . 14
6.6 アクセッサー . . . 15

7 契約クラス 16
7.1 責任 . . . 16
7.2 クラス定義 . . . 16
7.3 インスタンス変数定義：おサイフケータイ, 予約会員証 . . . 16
7.4 構成子 . . . 16
7.5 アクセッサー . . . 16

8 カードクラス 18
8.1 責任 . . . 18
8.2 クラス定義 . . . 18

9 クレジットカードクラス 19
9.1 責任 . . . 19
9.2 クラス定義 . . . 19
9.3 構成子 . . . 19

10 予約会員証 20
10.1 責任 . . . 20
10.2 クラス定義 . . . 20
10.3 構成子 . . . 20

11 TestApp クラス 21
11.1 責任 . . . 21
11.2 クラス定義 . . . 21
11.3 操作：run . . . 21

12 TestCaseT0001 クラス 23
12.1 責任 . . . 23

13 TestCaseT0002 クラス 24

13.1 責任 24

14 TestCaseT0003 クラス 25
14.1 責任 25

15 TestCaseT0004 クラス 28
15.1 責任 28

16 まとめ 31
16.1 統計情報 31

17 参考文献等 32

1　はじめに

鉄道会社 A の特急券予約システムに関する「特急券予約システムシステムの問題点を推測した仕様」(第**??** 章参照) を、本来あるべき仕様として書き直した仕様である。

1.1　本来どうすべきだったのか？

修正後のクラス図 1 で見るように、本来、特急券予約システムは契約[*1]とリンクが設定されているべきであり、契約がクレジットカードとリンクしているべきである。

予約会員証も契約を介してクレジットカードを参照できるようにすべきである。

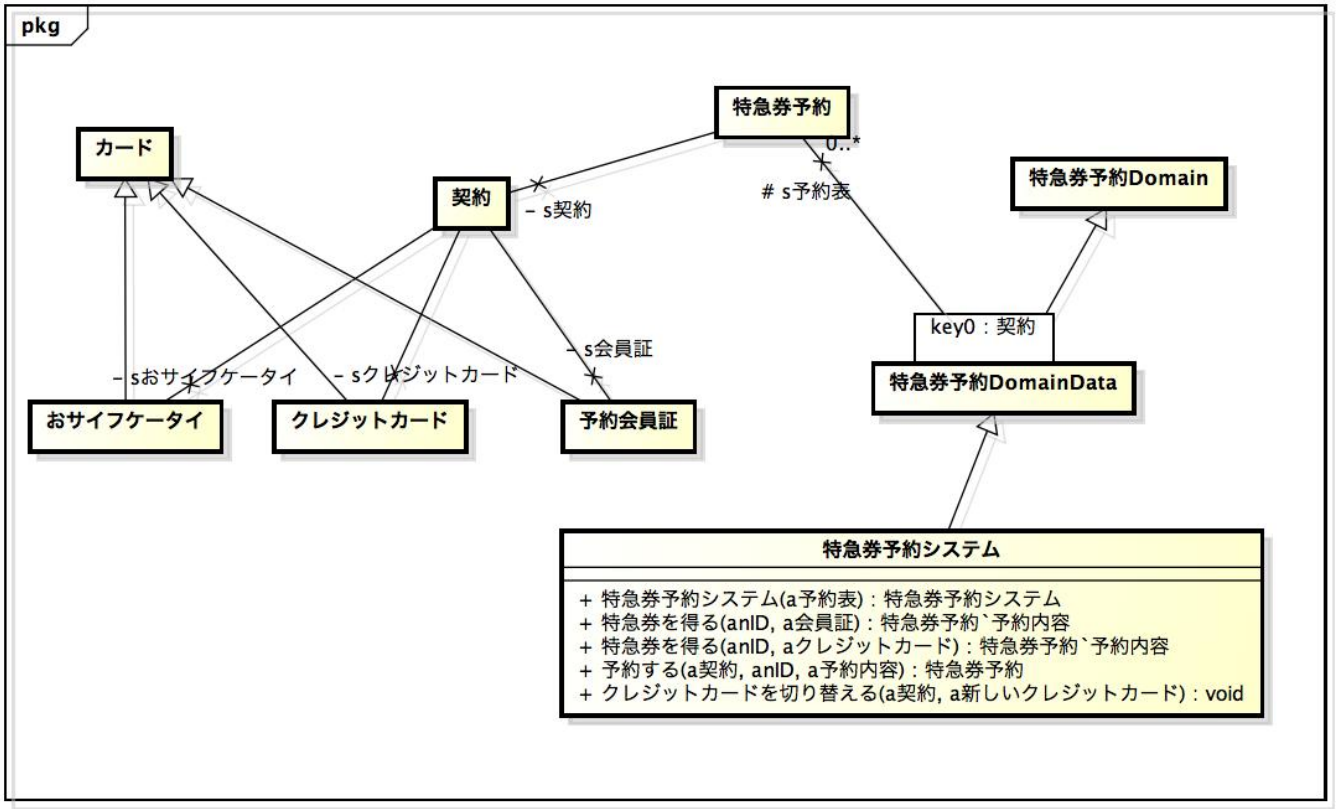

図 1　修正後のクラス図

このようにしておけば、契約に変更があっても、古い特急券予約は新しい契約を介して、新しいクレジットカードにアクセスでき、同じく予約会員証も新しいクレジットカードにアクセスできる。

修正前のクラス図**??** と比べれば、再利用性と保守性が増しているにもかかわらず、構造自体は特に複雑になっているわけではないことが分かる。

[*1] 口座と言ってもよいが、本モデルでは契約という名前にした

2　特急券予約システムクラス

2.1　責任

改良した特急券予約システムを、構造化日本語仕様レベルで表す。陽仕様の実行に関わる部分は、1 階層下の仕様階層である、スーパークラスの「特急券予約 Domain」で記述した。

2.2　クラス定義と構成子定義

```
.....................................................................................
class
特急券予約システム is subclass of 特急券予約 DomainData
operations
public
  特急券予約システム : 予約表 ==> 特急券予約システム
  特急券予約システム (a 予約表) ==
    (   s 予約表 := a 予約表
    );
.....................................................................................
```

2.3　操作：予約する

予約を行う。

```
.....................................................................................
public
  予約する : 契約 * ID * 特急券予約 ‘予約内容 ==> 特急券予約
  予約する (a 契約, anID, a 予約内容) ==
    (   def w 特急券予約 = new 特急券予約 (anID, a 契約, 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) ;
.....................................................................................
```

2.4 操作：特急券を得る

クレジットカードで特急券を得る。

```
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 ()
  post RESULT = 予約を得る (s 予約表, anID, a クレジットカード).予約内容を得る () ;
```

2.5 操作：特急券を得る

予約会員証で特急券を得る。

```
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 = w 予約.契約を得る ().会員証を得る () ;
```

2.6 操作：クレジットカードを切り替える

本操作は、一般ユーザーは使えず、鉄道会社 A の担当者だけが使える。

```
public
  クレジットカードを切り替える : 契約 * クレジットカード ==> ()
  クレジットカードを切り替える (a 契約, a 新しいクレジットカード) == a 契約.
    クレジットカードを設定する (a 新しいクレジットカード)
  pre  契約が存在する (a 契約, s 予約表)
end
```

特急券予約システム

Test Suite : vdm.tc

Class : 特急券予約システム

Name	#Calls	Coverage
特急券予約システム ‘予約する	72	√
特急券予約システム ‘特急券予約システム	36	√
特急券予約システム ‘クレジットカードを切り替える	9	√
特急券予約システム ‘特急券を得る	18	√
特急券予約システム ‘特急券を得る	117	√
Total Coverage		**100%**

3 共通定義クラス

3.1 責任

特急券予約モデルで共通に使用する定義を表す。

```
class
共通定義
types
 public ID = [token];
 public 暗証番号 = [token]
operations
public
 print : seq of char ==> ()
 print(a 文字列) ==
   let - = new IO().echo(a 文字列) in
   skip
end
共通定義
```

Test Suite : vdm.tc

Class : 共通定義

Name	#Calls	Coverage
共通定義 ‘print	18	√
Total Coverage		**100%**

4　特急券予約 Domain クラス

4.1　責任

特急券予約 Domain を表す、上の階層の仕様で使える名詞と述語の用語辞書、すなわち要求辞書である。

4.2　クラス定義

```
class
特急券予約 Domain is subclass of 共通定義
```

4.3　型およびインスタンス変数定義：予約表

```
types
  public 予約表 = map 契約 to set of 特急券予約
```

4.4　関数：予約を得る

ID を指定して、予約を得る。

```
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) ==
    (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 ;
```

4.5　関数：予約を得る

クレジットカードを指定して、予約を得る。

```
public
```

```
予約を得る : 予約表 * ID * クレジットカード -> 特急券予約
予約を得る (a 予約表, anID, a クレジットカード) ==
  (let  w 予約 in set dunion rng a 予約表 be st
          w 予約.クレジットカードを得る () = a クレジットカード and
         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.クレジットカードを得る () = a クレジットカード and
       w 予約.ID を得る () = anID ;
```

4.6 関数：予約を得る

予約会員証を指定して、予約を得る。

```
public
予約を得る : 予約表 * ID * 予約会員証 -> 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.会員証を得る () = a 予約会員証 and
         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.会員証を得る () = a 予約会員証 and
       w 予約.ID を得る () = anID ;
```

4.7 関数：予約表を更新する

契約と特急券予約を指定して、予約表を更新する。

```
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
```

```
  予約表が更新されている : 予約表 * 契約 * 特急券予約 * 予約表 +> bool
  予約表が更新されている (a 予約表, a 契約, a 特急券予約, a 更新後予約表) ==
    a 更新後予約表 = 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 })};
```

4.8　関数：予約表に追加する

契約と特急券予約を指定して、予約表に追加する。

```
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) ;
public
  予約表に追加されている : 予約表 * 契約 * 特急券予約 * 予約表 +> bool
  予約表に追加されている (a 予約表, a 契約, a 特急券予約, a 更新後予約表) ==
    a 更新後予約表 = a 予約表 munion {a 契約 |-> {a 特急券予約 }};
```

4.9　関数：予約がある契約である

予約がある契約か判定する。

```
public
  予約がある契約である : 契約 * 予約表 +> bool
  予約がある契約である (a 契約, a 予約表) ==
    契約が存在する (a 契約, a 予約表) and
    a 予約表 (a 契約) <> {};
```

4.10　関数：契約が存在する

契約が存在するか判定する。

```
public
```

```
  契約が存在する : 契約 * 予約表 +> bool
  契約が存在する (a 契約, a 予約表) ==
    a 契約 in set dom a 予約表;
```

4.11　関数：予約集合に追加する

既存予約集合と新規予約集合を指定して、予約集合に追加する。

```
public
  予約集合に追加する : set of 特急券予約 * set of 特急券予約 +> set of 特急券予約
  予約集合に追加する (a 既存予約集合, a 新規予約集合) ==
    a 既存予約集合 union a 新規予約集合
end
特急券予約 Domain
```

Test Suite :　vdm.tc

Class :　特急券予約 Domain

Name	#Calls	Coverage
特急券予約 Domain‘契約が存在する	216	√
特急券予約 Domain‘予約表に追加する	63	75%
特急券予約 Domain‘予約表を更新する	9	√
特急券予約 Domain‘予約集合に追加する	27	√
特急券予約 Domain‘予約がある契約である	207	√
特急券予約 Domain‘予約表が更新されている	18	√
特急券予約 Domain‘予約表に追加されている	126	√
特急券予約 Domain‘予約を得る	0	0%
特急券予約 Domain‘予約を得る	36	√
特急券予約 Domain‘予約を得る	315	√
Total Coverage		**83%**

5 特急券予約 DomainData クラス

5.1 責任

特急券予約 Domain で扱うデータを表す。

5.2 クラス定義

```
class
特急券予約 DomainData is subclass of 特急券予約 Domain
instance variables
  protected s 予約表 : 予約表 := { |-> };
```

5.3 操作：予約集合を得る

契約を指定して、予約集合を得る。

```
operations
public
  予約集合を得る : 契約 ==> set of 特急券予約
  予約集合を得る (a 契約) ==
    if dom s 予約表 = {}
    then return {}
    else return s 予約表 (a 契約)
  pre  dom s 予約表 <> {} => a 契約 in set dom s 予約表
end
特急券予約 DomainData
```

Test Suite : vdm.tc

Class : 特急券予約 DomainData

Name	#Calls	Coverage
特急券予約 DomainData‘予約集合を得る	36	60%
Total Coverage		**60%**

6　特急券予約クラス

6.1　責任

特急券予約 1 件を表す。

6.2　クラス定義

```
.....................................................................................
class
特急券予約 is subclass of 共通定義
.....................................................................................
```

6.3　型定義：予約内容

```
.....................................................................................
types
  public 予約内容 = token
.....................................................................................
```

6.4　インスタンス変数定義：ID, 契約, 予約内容

```
.....................................................................................
instance variables
  sID : ID;
  s 契約 : 契約;
  s 予約内容 : 予約内容;

.....................................................................................
```

6.5　構成子

```
.....................................................................................
operations
public
  特急券予約 : ID * 契約 * 特急券予約 ‘ 予約内容 ==> 特急券予約
  特急券予約 (anID, a 契約, a 予約内容) == atomic
    (  sID := anID;
       s 契約 := a 契約;
       s 予約内容 := a 予約内容
    );
```

6.6 アクセッサー

```
public
  ID を得る : () ==> ID
  ID を得る () ==
    return sID;
public
  契約を得る : () ==> 契約
  契約を得る () ==
    return s 契約;
public
  予約内容を得る : () ==> 予約内容
  予約内容を得る () ==
    return s 予約内容;
public
  クレジットカードを得る : () ==> クレジットカード
  クレジットカードを得る () ==
    return s 契約.クレジットカードを得る ();
public
  会員証を得る : () ==> 予約会員証
  会員証を得る () ==
    return s 契約.会員証を得る ()
end
```

特急券予約

Test Suite : vdm.tc

Class : 特急券予約

Name	#Calls	Coverage
特急券予約 ‘ID を得る	729	√
特急券予約 ‘契約を得る	18	√
特急券予約 ‘特急券予約	72	√
特急券予約 ‘会員証を得る	144	√
特急券予約 ‘予約内容を得る	216	√
特急券予約 ‘クレジットカードを得る	837	√
Total Coverage		**100%**

7　契約クラス

7.1　責任

特急券予約の契約 1 件を表す。

7.2　クラス定義

```
class
契約 is subclass of 共通定義
```

7.3　インスタンス変数定義：おサイフケータイ, 予約会員証

```
instance variables
  s おサイフケータイ : おサイフケータイ;
  s クレジットカード : クレジットカード;
  s 会員証 : 予約会員証;
```

7.4　構成子

実際には、契約が結ばれてから予約会員証が送られてくるまでの時差があるが、簡単化のため無視する。

```
operations
public
  契約 : おサイフケータイ * クレジットカード * 予約会員証 ==> 契約
  契約 (a おサイフケータイ, a クレジットカード, a 会員証) == atomic
    (  s おサイフケータイ := a おサイフケータイ;
       s クレジットカード := a クレジットカード;
       s 会員証 := a 会員証
    );
```

7.5　アクセッサー

```
public
```

```
  会員証を得る : () ==> 予約会員証
  会員証を得る () ==
    return s 会員証;
.....
.....
public
  クレジットカードを得る : () ==> クレジットカード
  クレジットカードを得る () ==
    return s クレジットカード;
public
  クレジットカードを設定する : クレジットカード ==> ()
  クレジットカードを設定する (a クレジットカード) ==
    s クレジットカード := a クレジットカード
end
```

契約

.....

Test Suite : vdm.tc

Class : 契約

Name	#Calls	Coverage
契約 ‘契約	63	√
契約 ‘会員証を得る	198	√
契約 ‘クレジットカードを得る	873	√
契約 ‘クレジットカードを設定する	9	√
Total Coverage		**100%**

8 カードクラス

8.1 責任

カードの抽象クラス。

8.2 クラス定義

```
.....................................................................................
class
カード is subclass of 共通定義
instance variables
  protected sID : ID := nil;
  protected s 暗証番号 : 暗証番号 := nil;

end
カード
.....................................................................................
```

Test Suite : vdm.tc

Class : カード

Name	#Calls	Coverage
Total Coverage		**undefined**

9 クレジットカードクラス

9.1 責任

クレジットカード 1 件を表す。

9.2 クラス定義

```
class
クレジットカード is subclass of カード
```

9.3 構成子

```
operations
public
  クレジットカード : ID * 暗証番号 ==> クレジットカード
  クレジットカード(anID, a 暗証番号) == atomic
    (  sID := anID;
       s 暗証番号 := a 暗証番号
    )
end
クレジットカード
```

Test Suite : vdm.tc

Class : クレジットカード

Name	#Calls	Coverage
クレジットカード`クレジットカード	63	√
Total Coverage		**100%**

10 予約会員証

10.1 責任

予約会員証 1 件を表す。

10.2 クラス定義

```
class
予約会員証 is subclass of カード
```

10.3 構成子

```
operations
public
  予約会員証 : ID * 暗証番号 ==> 予約会員証
  予約会員証 (anID, a 暗証番号) == atomic
    (  sID := anID;
       s 暗証番号 := a 暗証番号
    )
end
予約会員証
```

Test Suite : vdm.tc

Class : 予約会員証

Name	#Calls	Coverage
予約会員証 ‘予約会員証	63	√
Total Coverage		**100%**

11 TestApp クラス

11.1 責任

回帰テストを行う。

11.2 クラス定義

```
.....
class
TestApp
.....
```

11.3 操作 : run

回帰テストケースを TestSuite に追加し、実行し、成功したか判定する。

```
.....
operations
public static
 run : () ==> ()
 run () ==
   (   dcl ts : TestSuite := new TestSuite ("特急券予約モデルの回帰テスト \n"),
           tr : TestResult := new TestResult ();
       tr.addListener(new PrintTestListener ());
       ts.addTest(new TestCaseT0001 ("TestCaseT0001 通常の予約成功 \n"));
       ts.addTest(new TestCaseT0002 ("TestCaseT0002 クレジットカードを切り替えたときの予約
成功 \n"));
       ts.addTest(new TestCaseT0003 ("TestCaseT0003 クレジットカードを切り替え、新しいクレ
ジットカードで新しい予約の特急券を得るが、古いクレジットカードでは失敗 \n"));
       ts.addTest(new TestCaseT0004 ("TestCaseT0004 クレジットカードを切り替え、新しいクレ
ジットカードで古い予約の特急券を得るが、古いクレジットカードでは失敗 \n"));
       ts.run(tr);
       if tr.wasSuccessful () = true
       then def - = new IO ().echo ("＊＊＊ 全回帰テストケース成功。 ＊＊＊") in
            skip
       else def - = new IO ().echo ("＊＊＊ 失敗したテストケースあり!! ＊＊＊") in
            skip
   )
end
TestApp
```

Test Suite : vdm.tc

Class : TestApp

Name	#Calls	Coverage
TestApp‘run	9	85%
Total Coverage		**85%**

12 TestCaseT0001 クラス

12.1 責任

通常の予約をテストする。

```
.....................................................................................
class
TestCaseT0001 is subclass of TestCase,共通定義
operations
public
 TestCaseT0001 : seq of char ==> TestCaseT0001
 TestCaseT0001(name) ==
   setName(name);
public
 test01 : () ==> ()
 test01() ==
   (   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会員証 = new 予約会員証(mk_token(<会員証 ID01>),mk_token(<会員証 PW01>));
          w契約 = new 契約(wおサイフケータイ,wクレジットカード,w会員証);
          wシステム = new 特急券予約システム({ |-> }) in
       (   assertTrue("\ttest01 契約に失敗",
              wシステム.予約集合を得る(w契約) = {} and
              w契約.会員証を得る() = w会員証);
          def - = wシステム.予約する(w契約,mk_token(<予約 ID01>),mk_token(<最初の予約内
容>)) in
          assertTrue("\ttest01 予約に失敗",
              wシステム.特急券を得る(mk_token(<予約 ID01>),wクレジットカード) = mk_token(<
最初の予約内容>));
          def - = wシステム.予約する(w契約,mk_token(<予約 ID02>),mk_token(<次の予約内容
>)) in
          assertTrue("\ttest01 2回目の予約に失敗",
              wシステム.特急券を得る(mk_token(<予約 ID02>),wクレジットカード) = mk_token(<
次の予約内容>))
       )
   )
end
TestCaseT0001
```

Test Suite : vdm.tc
Class : TestCaseT0001

Name	#Calls	Coverage
TestCaseT0001‘test01	9	√
TestCaseT0001‘TestCaseT0001	9	√
Total Coverage		**100%**

13 TestCaseT0002 クラス

13.1 責任

クレジットカードを切り替えたときの予約をテストする。

```
class
TestCaseT0002 is subclass of TestCase, 共通定義
operations
public
  TestCaseT0002 : seq of char ==> TestCaseT0002
  TestCaseT0002(name) ==
    setName(name);
public
  test01 : () ==> ()
  test01() ==
    (   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 会員証 = new 予約会員証(mk_token(<会員証 ID01>), mk_token(<会員証 PW01>));
            w 契約 = new 契約(w おサイフケータイ, w クレジットカード, w 会員証);
```

```
            w システム = new 特急券予約システム ({ |-> }) in
        (   assertTrue("\ttest01 契約に失敗",
                w システム.予約集合を得る (w 契約) = {} and
                w 契約.会員証を得る () = w 会員証) ;
            def - = w システム.予約する (w 契約, mk_token (<予約 ID01>), mk_token (<最初の予約内
容>)) in
            assertTrue("\ttest01 予約に失敗",
                w システム.特急券を得る (mk_token (<予約 ID01>), w クレジットカード) = mk_token (<
最初の予約内容>)) ;
            def w2 クレジットカード = new クレジットカード (mk_token (<クレジット ID02>), mk_token (<
クレジット PW02>));
                w2 会員証 = new 予約会員証 (mk_token (<会員証 ID02>), mk_token (<会員証 PW02>));
                w2 契約 = new 契約 (w おサイフケータイ, w2 クレジットカード, w2 会員証) in
            (   def - = w システム.予約する (w2 契約, mk_token (<予約 ID02>), mk_token (<次の予約
内容>)) in
                assertTrue("\ttest01 2 回目の予約に失敗",
                    w システム.特急券を得る (mk_token (<予 約 ID02>), w2 ク レ ジ ッ ト カ ー ド
) = mk_token (<次の予約内容>))
            )
        )
    )
end
TestCaseT0002
```

Test Suite : vdm.tc

Class : TestCaseT0002

Name	#Calls	Coverage
TestCaseT0002‘test01	9	√
TestCaseT0002‘TestCaseT0002	9	√
Total Coverage		**100%**

14 TestCaseT0003 クラス

14.1 責任

クレジットカードを切り替え、予約した特急券を得る場合をテストする。古いクレジットカードでは「Error 58: 事前条件の評価結果が false です」という実行時エラーが発生し、変更後のクレジットカードでは、特急券を得ることができる。

```
class
TestCaseT0003 is subclass of TestCase,共通定義
operations
public
  TestCaseT0003 : seq of char ==> TestCaseT0003
  TestCaseT0003(name) ==
    setName(name);
public
  test01 : () ==> ()
  test01() ==
    (   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 会員証 = new 予約会員証(mk_token(<会員証 ID01>),mk_token(<会員証 PW01>));
            w 契約 = new 契約(w おサイフケータイ,w クレジットカード,w 会員証);
            w 古いクレジットカード = w 契約.クレジットカードを得る();
            w システム = new 特急券予約システム({ |-> }) in
        (   assertTrue("\ttest01 契約に失敗",
                 w システム.予約集合を得る(w 契約) = {} and
                 w 契約.会員証を得る() = w 会員証);
            def - = w システム.予約する(w 契約,mk_token(<予約 ID01>),mk_token(<最初の予約内
容>)) in
            assertTrue("\ttest01 予約に失敗",
                 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
) = mk_token(<最初の予約内容>));
            def w2 クレジットカード = new クレジットカード(mk_token(<クレジット ID02>),mk_token(<
クレジット PW02>));
                w2 会員証 = new 予約会員証(mk_token(<会員証 ID02>),mk_token(<会員証 PW02>));
                w2 契約 = new 契約(w おサイフケータイ,w2 クレジットカード,w2 会員証);
```

```
                w 変更後のクレジットカード = w2 契約.クレジットカードを得る () in
            (   def - = w システム.予約する (w2 契約, mk_token (<予約 ID02>), mk_token (<次の予約
内容>)) in
                assertTrue("\ttest01 2 回目の予約に失敗",
                    w システム.特急券を得る (mk_token (<予約 ID02>), w 変更後のクレジットカード
) = mk_token (<次の予約内容>));
                def w2 予約内容 = w システム.特急券を得る (mk_token (<予約 ID02>), w 変更後のクレ
ジットカード) in
                assertTrue("\ttest01 変更後の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 (<次の予約内容>));
                def w2 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 予約会員証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 テスト意図通り、古いクレジットカードで特急券を得ること
に失敗 \n") in
                def w3 予約内容 = w システム.特急券を得る (mk_token (<予約 ID02>), w 古いクレジッ
トカード) in
                assertTrue("\ttest01 テスト意図に反し、古いクレジットカードで特急券を得るこ
とに成功 \n",
                    w3 予約内容 = mk_token (<次の予約内容>))
            )
        )
    )
end
TestCaseT0003
```

Test Suite : vdm.tc

Class : TestCaseT0003

Name	#Calls	Coverage
TestCaseT0003‘test01	9	94%
TestCaseT0003‘TestCaseT0003	9	√
Total Coverage		**94%**

15 TestCaseT0004 クラス

15.1 責任

クレジットカードを切り替え、古いクレジットカードで予約した特急券を得る場合をテストする。新しいクレジットカードでは特急券を得ることができるが、古いクレジットカードでは「Error 58: 事前条件の評価結果が false です」という実行時エラーが発生するが、「test01 テストの意図通り、古いクレジットカードで特急券を得ることに失敗」というメッセージをログに表示して、回帰テストとしては成功する。

```
.....
class
TestCaseT0004 is subclass of TestCase, 共通定義
operations
public
 TestCaseT0004 : seq of char ==> TestCaseT0004
 TestCaseT0004(name) ==
   setName(name);
public
 test01 : () ==> ()
 test01() ==
   (   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 会員証 = new 予約会員証(mk_token(<会員証 ID01>),mk_token(<会員証 PW01>));
           w 契約 = new 契約(w おサイフケータイ,w クレジットカード,w 会員証);
           w 古いクレジットカード = w 契約.クレジットカードを得る();
           w システム = new 特急券予約システム({ |-> }) in
       (   assertTrue("\ttest01 契約に失敗",
                w システム.予約集合を得る(w 契約) = {} and
                w 契約.会員証を得る() = w 会員証);
           def - = w システム.予約する(w 契約,mk_token(<予約 ID01>),mk_token(<最初の予約内
容>)) in
           assertTrue("\ttest01 予約に失敗",
                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
) = mk_token(<最初の予約内容>));
           def w2 クレジットカード = new クレジットカード(mk_token(<クレジット ID02>),mk_token(<
クレジット PW02>));
               w2 会員証 = new 予約会員証(mk_token(<会員証 ID02>),mk_token(<会員証 PW02>));
               w2 契約 = new 契約(w おサイフケータイ,w2 クレジットカード,w2 会員証);
```

```
                w 変更後のクレジットカード = w2 契約.クレジットカードを得る () in
            (   def - = w システム.予約する (w2 契約, mk_token (<予約 ID02>), mk_token (<次の予約
内容>)) in
                assertTrue("\ttest01 2 回目の予約に失敗",
                    w システム.特急券を得る (mk_token (<予約 ID02>), w 変更後のクレジットカード
) = mk_token (<次の予約内容>));
                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("\ttest01 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>));
                def w2 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 変更前の予約会員証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、古いクレジットカードで特急券を得るこ
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 変更後
のクレジットカード);
                        w4 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("\ttest01 テスト意図に反し、変更前の予約会員証で特急券を
得ることに失敗 \n",
                            w3 予約内容 = mk_token (<最初の予約内容>));
                        assertTrue("\ttest01 テスト意図に反し、予約会員証で特急券を得ること
に成功したが、予約内容が正しくない \n",
                            w4 予約内容 = mk_token (<最初の予約内容>))
                    )
                )
            )
        )
    )
end
TestCaseT0004
```

Test Suite : vdm.tc

Class : TestCaseT0004

Name	#Calls	Coverage
TestCaseT0004‘test01	9	90%

Name	#Calls	Coverage
TestCaseT0004‘TestCaseT0004	9	√
Total Coverage		**90%**

16　まとめ

16.1　統計情報

注釈抜きの VDM++ ソース行数は、492 行、モデル作成工数は約 1 日、発表用の資料作成と VDM++ モデルの読みやすさのための整形清書に約 2 日かかった。

なお、上記モデルの作成工数は、筆者が問題を解決するために消費した工数より少ない。

17　参考文献等

VDM++[2] は、1970 年代中頃に IBM ウィーン研究所で開発された VDM-SL[1] を拡張し、さらにオブジェクト指向拡張したオープンソース[*2]の形式仕様記述言語である。

参考文献

[1] CSK システムズ. VDM-SL 言語マニュアル. CSK システムズ, 第 1.1 版, 2007. Revised for VDMTools V7.1.

[2] CSK システムズ. VDM++ 言語マニュアル. CSK システムズ, 第 1.1 版, 2007. Revised for VDMTools V7.1.

[*2] 使用に際しては、（株）CSK システムズとの契約締結が必要になる。

索引

クレジットカードを切り替える, 6
契約, 16
修正後のクラス図, 4
特急券予約システム, 5
特急券予約 Domain, 9
特急券予約 DomainData, 13
特急券予約, 14
特急券を得る, 6
予約集合を得る, 13
予約する, 5
予約表に追加する, 11
予約表を更新する, 10
予約を得る, 9, 10