Alloy Analyzer によるモデリング入門 演習課題

2013 年 1 月 18 日 小林健一

Step1: 以下の記述を元に、モデルを作成し、run でインスタンスを生成してください。

1.課題概要「トレーニングコース申込システム」

Formal Methods Education 社(FME)は、形式手法を中心としたシステム開発のコンサルティング会社です。FME では定期的に、トレーニングコースを開催しています。トレーニングコースの申込は、Web 上から申込システムを使用することによって行います。

2.モデル要素

トレーニングコース申込システムのデータモデルは、おおむね以下の要素から成り立ちます。

- コース:トレーニングコースのマスタデータです。

ex.
コース A(ID[1],コース名[Alloy Analyzer 入門],標準受講料[15,000 円])
コース B(ID[2],コース名[Coq で学ぶ定理証明],標準受講料[68,000 円])

- 開催日程:コースに対する、具体的な各開催回です。

ex.
開催日程 A(ID[11],コース ID[1],開催[4 月 27 日],会場[東京],受講料[15,000 円])」
開催日程 B(ID[12],コース ID[1],開催[5 月 31 日],会場[東京],受講料[15,000 円])」
開催日程 C(ID[13],コース ID[2],開催[9 月 10 日,11 日],会場[名古屋],受講料[68,000 円])」

- 日付:各開催日程と紐づく、年月日データです。

ex.
日付 A(ID[201],日付[2013/5/31])
日付 B(ID[202],日付[2013/9/10])
日付 C(ID[203],日付[2013/9/11])

- 会場:各開催日程と紐づく、会場データです。

ex.
東京会場(ID[501],会場名[東京],定員[20 名],会場住所[東京都新宿区 2-1-1])
名古屋会場(ID[502],会場名[名古屋],定員[16 名],会場住所[愛知県名古屋市 1-2-2])

- 申込:各開催日程に対して、申込者が申し込むデータです。

ex.
申込 1(ID[3],開催日程 ID[1],申込日時[2013/1/11 12:30])

- 受講者:各申込に紐づく、受講者データです。

ex.
受講者 1(ID[5],申込 ID[3],氏名[奥山登])
受講者 2(ID[6],申込 ID[3],氏名[氷川透])

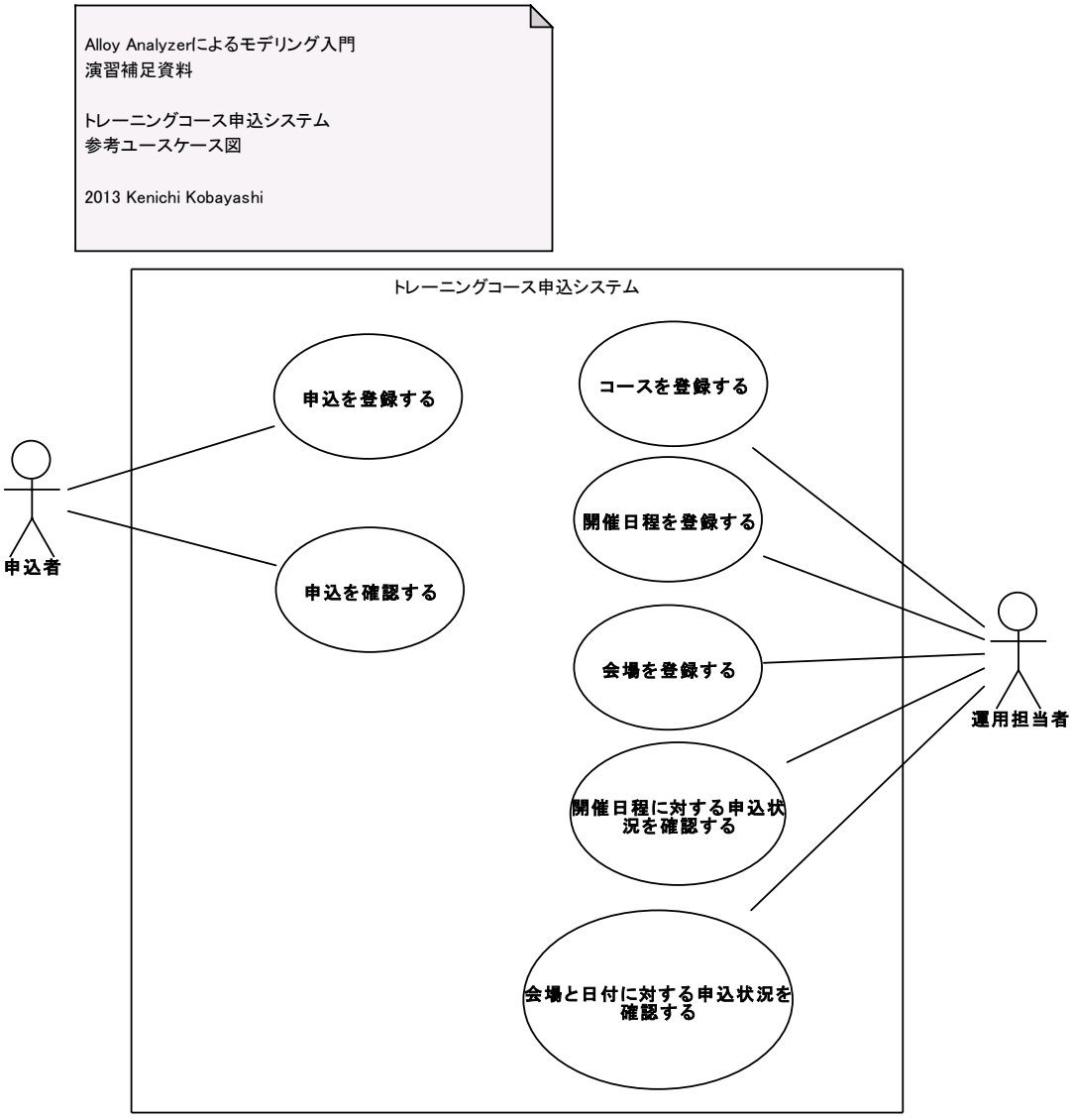
3.データ間の関係

1 件のコースには、0 件以上の開催日程が対応します。
1 件の開催日程には、必ず 1 件のコースが対応します。
1 件の開催日程には、1 件以上の日付が対応します。
1 件の開催日程には、必ず 1 件の開場が対応します。
1 件の開催日程には、0 件以上、会場定員以下の申込が対応します。
1 件の申込には、1 件以上 5 件以下の受講者が対応します。
同一会場同一日付に対応する開催日程は、0 または 1 件です。

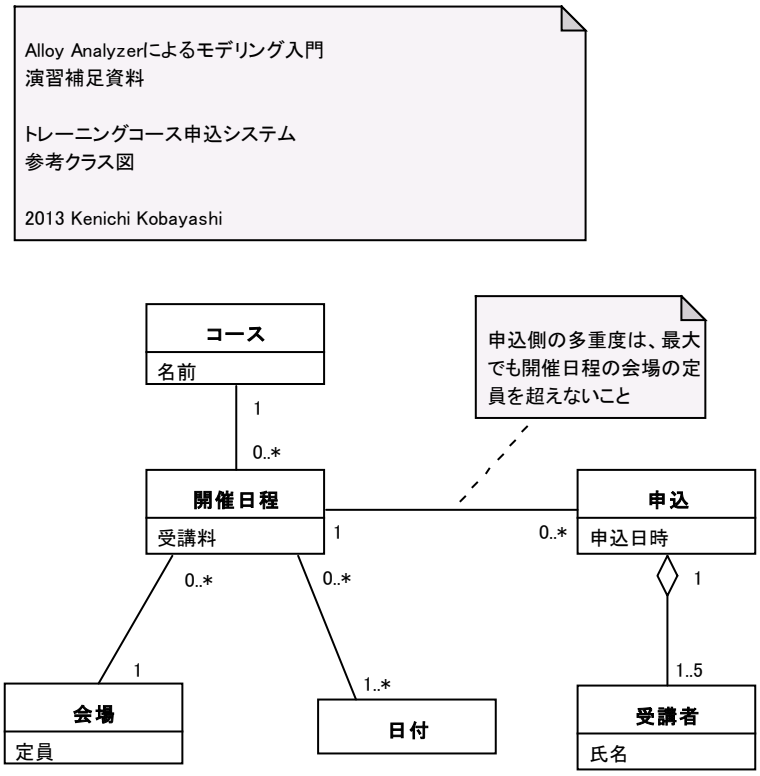
4.申込システム運用者からの基本的要求

1. コースを指定することにより、コースに対応する開催日程一覧を取得したい
2. 開催日程を指定することにより、開催日程に対する申込に対応する受講者一覧を取得したい
3. 会場と日付を指定することにより、指定された会場と日付に対応する開催日程を取得したい

参考:ユースケース図


Alloy Analyzerによるモデリング入門
演習補足資料
トレーニングコース申込システム
参考ユースケース図
2013 Kenichi Kobayashi
トレーニングコース申込システム
申込を登録する
申込を確認する
コースを登録する
開催日程を登録する
会場を登録する
開催日程に対する申込状況を確認する
会場と日付に対する申込状況を確認する
申込者
運用担当者

参考:クラス図


Alloy Analyzerによるモデリング入門
演習補足資料
トレーニングコース申込システム
参考クラス図
2013 Kenichi Kobayashi
コース
名前
1
0..*
申込側の多重度は、最大でも開催日程の会場の定員を超えないこと
開催日程
受講料
1
0..*
申込
申込日時
0..*
0..*
1
1
1..*
1..5
会場
定員
日付
受講者
氏名

Step2: インスタンス数を制限する fact を追加してください。

2.1 fact 各々の申込には最低一人最大五名の受講者が対応する

サンプル:

```
fact 各々の申込には最低一人最大五名の受講者が対応する {
        all e:申込 | #(受講者_申込.e) > 0 and #(受講者_申込.e) < 6
}
```

2.2 fact 特定開催日程における申込数の合計は会場のキャパシティ以内であること

2.3 fact 同一会場同一日付での複数開催日程は存在しない

Step3: 下記の assert を追加し、check をしてください。

3.1 assert 一つの開催日程には一つのコースが対応する

サンプル:

```
assert 一つの開催日程には一つのコースが対応する {
        all s:開催日程 | one s.開催日程_コース
}
check 一つの開催日程には一つのコースが対応する
```

3.2 assert 開催日程に紐づかない申込は存在しないこと

3.3 assert 申込に紐づかない受講者は存在しないこと

3.4 assert 開催日程に紐づかない日付は存在しないこと

Step4: データの取得操作を fun 定義により追加してください。

4.1 fun 開催日程取得[c:コース]: set 開催日程

サンプル:

```
fun 開催日程取得[c:コース]:set 開催日程 {
        開催日程_コース.c
}
```

4.2 fun コース取得[s:開催日程]:set コース

4.3 fun 開催日程取得[e:申込]: set 開催日程

4.4 fun 申込取得[s:開催日程]: set 申込

4.5 fun 日付取得[s:開催日程]: set 日付

4.6 fun 会場取得[s:開催日程]: set 会場

4.7 fun 開催日程取得[p:会場,d:日付]: set 開催日程

4.8 fun 受講者取得[s:開催日程]: set 受講者

4.9 fun 受講者取得[p:会場,d:日付]:set 受講者

Step5 操作前後のチェックを pred により追加してください。

5.1 pred 開催日程登録[c,c':コース,s:開催日程]

条件:同一会場、同一日付に別の開催日程が存在しなければ登録できる

サンプル:

```
pred 開催日程登録[c,c':コース,s:開催日程] {
    no t:開催日程 | t != s and t.開催日程_会場 = s.開催日程_会場 and t.開催日程_日付 = s.開催日程_日付
    開催日程_コース.c' = 開催日程_コース.c + s
}
```

5.2 pred 申込登録[s,s':開催日程,e:申込,t:受講者]

条件:申込最大数未満であれば登録できる

5.3 pred 開催日程削除[c,c':コース,s:開催日程]

申込が存在しない開催日程のみ削除できる

5.4 pred 会場削除[p:会場]

開催日程が存在しない会場のみ削除できる

オプション課題:追加仕様の検討

以降は、自由に仕様を追加変更して、結果を試してください。参考までに、いくつか仕様の追加変更例です。

Step6 セットコース(セット受講割引サービス)を追加する

コースをまとめて新たに一つのコースとする。

ex

モデリングコースは{やさしい形式手法、Alloy Analyzer 入門、Alloy Analyzer 基礎}を一セットにしたコースとする

モデリングの方針 1: コースの階層構造を作る

モデリングの方針 2: プロジェクトとコースの 2 つに分ける(プロジェクト:コース= 1:n)

注意点: セットコースの目的は、まとめて受講した際の割引サービスの提供であるため、セットコースに含まれる各開催日程は、独立した日程としても受講可能とする

Step7 初日のみ受講コースを追加する

ex: 「B-Method 演習(2 日間)」のうち、1 日目は「B-Method 入門(1 日目)」コースとして販売する

Step8 特定開催日程割引キャンペーンを実施する

ex: 2 月 8 日の「やさしい形式手法」のみ、50%引きとする

Step9 優待 ID を追加する

ex: 優待 ID:abcdef を付与して申し込んだ場合は 50%引きとする

注意点: 優待 ID は、特定対象コースに対してのみ有効とする

Step10 担当講師を追加する

ex: 担当講師{小森}は、開催日程{Scala プログラミング入門,2013/2/5}を担当する