B-011

Web アプリケーションのための拡張画面遷移図から VDM++仕様への変換手法
A Conversion Technique from Extended Screen Transition Diagrams to VDM++ Specifications for Web Applications

石上 椋一† 松本 翔† 長尾 康生‡ 高木 智彦‡
Ryoichi Ishigami Sho Matsumoto Koki Nagao Tomohiko Takagi

1. はじめに

拡張画面遷移図は，Web アプリケーションの画面遷移を表す有向グラフに，制約条件（事前／事後／不変条件）や操作などの拡張情報を付加した仕様記述法である．筆者らは，更新頻度の高い Web アプリケーションの開発を支援するために，拡張画面遷移図に基づくテスト保守フレームワークを開発している[1]．本フレームワークでは，開発者はテンプレートを用いて MVC（Model，View，Controller）モデルに基づく Web アプリケーションを実装する．そして，支援ツールを用いてその Web アプリケーションを俯瞰する拡張画面遷移図を作成し，テストケース群（テストスイート）の生成や修正などに利用することを目指している．テンプレートおよび支援ツールを用いて，実装と拡張画面遷移図の間のトレーサビリティを確立する．これにより，保守工程において比較的少ない労力で，実装と拡張画面遷移図の対応関係を明らかにしてレビューを促進するとともに，保守による変更をテストスイートに反映させることができると考えられる．

本稿では本フレームワーク開発の一環として，拡張画面遷移図を，形式仕様記述言語 VDM++ [2]による仕様（VDM++仕様）に変換する手法について議論する．VDM++仕様の特長は曖昧さがないことであり，拡張画面遷移図を直接利用するよりも，精密なテストスイートを体系的に生成したり修正したりできると考えられる．この変換は，(1) 拡張画面遷移図から JSON への変換，(2) JSON から VDM++仕様への変換の 2 段階で行われる．JSON（JavaScript Object Notation）は，プログラムで処理しやすく，人が理解しやすい形式でもあるため導入した．本稿では，2 節と 3 節で(1) (2)それぞれの段階について手法を議論した上で，4 節で課題を考察する．

2. 拡張画面遷移図から JSON への変換

本研究の拡張画面遷移図は，MVC モデルの Model，View，Controller を包含している．Model はデータベース構成，View は画面構成，Controller はデータ処理方法に概ね対応している．これらのうち，Model については本フレームワークを用いた Web アプリケーション開発の最初のステップにおいて JSON で定義することになっている[1]．図 1 は，JSON で定義されたデータベース構成の簡単な例である．よって本節では，View と Controller を JSON に変換する手法について議論する．

拡張画面遷移図は，複数の画面遷移から構成される．さらに各画面遷移は，(a) 遷移元画面，(b) 遷移先画面，拡張

```
{
  "databases" : {
    "sql" : {
      "table" : {
        "id" : "INTEGER PRIMARY KEY",
        "message" : "TEXT"
      }
    }
  }
}
```

図 1　JSON で定義されたデータベース構成の例

```
"Index_to_Result" : {
    "pre"  : "cookie.session_id = true",
    "post" : "message in table.message",
    "operation" : {
              "SUBMIT" : "click",
              "message" : "send",
              "procedure" : <省略>
    }
}
```

図 2　JSON に変換された画面遷移の例

情報（すなわち，(c) 制約条件や(d) 操作）から構成される．View は(a) (b) (d)と，Controller は(c) (d)と密接に関連している．なお，制約条件は，Model に加えて，(d)で使用される変数や Cookie も対象になる．本手法では，まず拡張画面遷移図の各構成要素を抽出する．そして，構成要素の名称と内容の組を，構成要素間の包含関係に基づいて列挙することにより JSON を得る．

図 2 に，JSON に変換された画面遷移の簡略化された例を示す．先頭は「<遷移元画面>_to_<遷移先画面>」の形式で記述されており，Index 画面から Result 画面への遷移を表していることが分かる．この画面遷移の拡張情報は，pre（事前条件），post（事後条件），operation（画面遷移のトリガとなる操作）から構成される．そしてさらに operation は，SUBMIT（ユーザのボタン押下による提出操作）と message（当該操作の実行に必要な，ユーザから入力されるべきテキストデータ），procedure（当該操作の実行に伴うデータ処理手続き）から構成される．procedure の内容はここでは省略したが，自然言語あるいは何らかの形式言語による仕様記述が想定されている．それ以外は，VDM++や本フレームワークが連携する予定の外部ツール[3]などを参考にした独自表記が用いられる．

† 香川大学大学院創発科学研究科　Graduate School of Science for Creative Emergence, Kagawa University
‡ 香川大学創造工学部　Faculty of Engineering and Design, Kagawa University

```
class IndexScreen is subclass of Screen              -- Index 画面のクラス定義

   operations                                        -- 操作定義（Index 画面を起点とする遷移）
     public click_SUBMIT: Table * Cookie * seq of char ==> ()
     click_SUBMIT(table, cookie, message) == (
         <省略>                                        -- データ処理手続き（procedure に対応）
         cookie.current_screen := new ResultScreen();    -- 画面を更新（Result 画面に遷移）
     )
     pre cookie.session_id = true                                                    -- 事前条件
     post exists record in set elems table.table & message = record.message; -- 事後条件

     -- 以下同様に，Index 画面を起点とする他の遷移の操作定義が続く
                                  ⋮
end IndexScreen
```

図 3　VDM++仕様に変換された画面遷移の例

3. JSON から VDM++仕様への変換

次に，データベース構成の JSON，および画面遷移の JSON から VDM++仕様に変換する手法について議論する．VDM++仕様は，複数のクラス定義から構成され，各クラスは，型，インスタンス変数，操作などの定義から構成される．

本手法では，データベース構成の JSON に基づいて，データベースの各テーブルのクラスをそれぞれ定義する．テーブルクラスにおいては，テーブルの各レコードを扱うためのレコード型を定義する．レコード型は，当該テーブルのカラムを要素としてもつ．そして，そのレコード型を用いて，テーブルのデータを格納するためのインスタンス変数を定義する．テーブルにアクセスするための操作や，レコード，カラムに対する不変条件なども必要に応じて定義する．

一方，画面遷移の JSON からは，画面，Cookie，全体管理のクラスをそれぞれ定義する．

画面クラスについては，まず，すべての画面に共通するものを備えた親クラス（親画面クラス）を定義する．そして，各画面に対応する画面クラスを，親画面クラスを継承するクラス（子画面クラス）としてそれぞれ定義する．各子画面クラスでは，当該画面においてデータ入力を受け付けて処理し，画面遷移を実行する操作を記述する．各操作には，対応する画面遷移の事前条件や事後条件を付与する．

Cookie クラスでは，セッション管理のための主要な情報を保持するための型やインスタンス変数，初期化の操作などを定義する．ユーザが現在滞在している画面に関する情報も，子画面クラスのオブジェクトとしてここで保持される．

全体管理クラスにおいては，テーブルクラスや Cookie クラスのオブジェクトを扱うためのインスタンス変数や，指定したセッションの画面遷移を実行するための操作などを定義する．

画面クラスの操作の簡単な例として，図 2 から変換される VDM++仕様を図 3 に示す．これは Index 画面に対応する子画面クラスであり，Index 画面から Result 画面への遷移を実行する操作が定義されている．この操作の引数は，当該画面遷移においてアクセスされるテーブルクラスのオブジェクト，当該セッションの Cookie クラスのオブジェクト，当該画面において入力されるテキストデータ（文字列）の 3 つである．操作の内部では，JSON の procedure に対応するデータ処理手続きが記述される（ただし，この例では省略されている）．そして操作の最後に，遷移先の Result 画面に対応する子画面クラスのオブジェクトが生成され，画面が更新される．また，JSON にしたがって，事前条件と事後条件が付与されている．事前条件を満たさなければ当該操作は実行できないし，事後条件を満たさない場合は当該操作の実行において問題が発生したと判断できる．これらは，テストスイートの実行可能性の確保や，仕様上の誤りの検出などに役立つ．

4. おわりに

本稿では，テスト保守フレームワークの開発の一環として，拡張画面遷移図を VDM++仕様に変換する手法について議論した．この変換は，(1) 拡張画面遷移図から JSON への変換，(2) JSON から VDM++仕様への変換の 2 段階で行われる．VDM++仕様に基づく効果的なテストスイートの生成や修正を可能にするには，VDM++仕様に充分な情報が含まれなければならない．したがって，VDM++仕様の基となる拡張画面遷移図や JSON の表記法について検討する余地があると考えられる．また，本手法はツールによって支援されることが望ましいので，変換ルールやツールの設計について検討することを計画している．ただし，変換を完全に自動化することは困難である．たとえば，JSON において自然言語で記述された procedure の内容は，手作業で VDM++仕様に変換される必要がある．このような作業に要する労力についても今後の課題である．

謝辞

本研究は JSPS 科研費 JP17K00103 の助成を受けた．

参考文献

[1] 石上椋一, 高木智彦, "拡張画面遷移図に基づく Web アプリケーションのためのテスト保守手法とそのフレームワーク", 情報処理学会全国大会講演論文集, pp.269-270, Mar. 2023.

[2] J. Fitzgerald, P.G. Larsen, P. Mukherjee, N. Plat, M. Verhoef, "Validated Designs for Object-Oriented Systems", Springer-Verlag London, 2005.

[3] 伊藤望, 戸田広, 沖田邦夫, 宮田淳平, 長谷川淳, 清水直樹, Vishal Banthia, "Selenium 実践入門-自動化による継続的なブラウザテスト", 技術評論社, 2016.