4M-4

要求仕様と設計の機能要件のトレーサビリティを保持する為のWebアプリケーション設計手法の評価

奥田 博隆† 小形 真平‡ 松浦 佐江子‡‡

芝浦工業大学†,‡‡ 信州大学‡

1. はじめに

近年、実行環境に依存しない技術としてWebアプリケーションの重要度が高まっている。

我々は、ユースケース分析に基づく業務系 Web システム開発を対象に開発者の定義した要求分析モデルから顧客がその妥当性を確認出来るWeb UI(User Interface)のプロトタイプ生成を特徴としたUML(Unified Modeling Language)を用いたモデル駆動要求分析手法[1]を提案してきた。

しかし、要求仕様から系統的に漏れなく正確に設計仕様へ適用する事(トレーサビリティ)と複数ユーザを想定した Web の特徴を設計モデルへ追加定義の方法が確立されていなかった。そこで、我々は、効率・拡張性の向上の為、開発環境に応じて MVC パターンに基づく様々な特徴を持つ Web フレームワークを用いる事が多い事に注目し、フレームワーク固有の処理とロジックの分離を特徴とする要求分析モデルからのWeb フレームワーク設計モデルを提案した[2]。本稿では、我々が提案するモデル駆動要求分析手法を用いて、授業で用いるグループワーク支援システム(GWSS)の開発を行った結果から、そのトレーサビリティをソースコードとモデルの関係から議論する。

2. 関連研究

Cadavid 等は、Domain Specific Language の立場からWeb ページの入出力構成を入出力項目の Form、要素の繰り返し表示を List などのステレオタイプを付記したクラスとして表現し、そこに表示したい要素を属性に記述し、その振る舞いを CRUD(Create/Read/Update/Delete)観点からタグ付き値として付記したモデルからの Web アプリケーションの生成手法を提案している[3]。生成されるアプリケーションは、エンティティの CRUD とその表示機能を持つ。一方、本手法では、顧客の要求に基づいて表示や入力方法とその振る舞いを決定する要求分析モデルを用いて設計モデルへ適用する事から、単に CRUD 機能だけでなく、要求から来る振る舞いに対してもトレーサビリティが保てる。

3. モデル駆動要求分析手法と設計手法

3.1. 要求分析手法

要求分析では、システムが提供するユースケースを、その実施主体者とシステムの相互作用として定義することが目的で、数種の UML を用いて要求分析モデルを定義する。特に、図 1 上段のように 3 つの観点の振る舞いをユーザの動作起点となるトリガを含む入出力項目(ユーザ)、入力項目検査やユーザへの通知(インタラクション)、トリガが起点となり起動する動作系列であるビジネスロジックとその例外フロー(システム)をアクティビティ図のパーティションに分けて明確に定義する。データ構造は UML クラス図で定義し、その具体値はオブジェクト図で定義する。

Web Application Design Method for Maintaining Traceability of Functional Requirement Between Design and Requirements Specifications

† Hirotaka Okuda, Shibaura Institute of Technology

‡ Shinpei Ogata, Shinshu University

‡‡ Saeko Matsuura, Shibaura Institute of Technology

3.2. Web フレームワーク設計モデル化手法

要求分析モデルの User, Interaction, System パーティションがフレームワークの View, Controller, Model に対応づく。しかし、Controller にはフレームワーク固有の仕組による実装方法の違いからフレームワーク固有の View 依存の処理とビジネスロジックの分離が必要である。

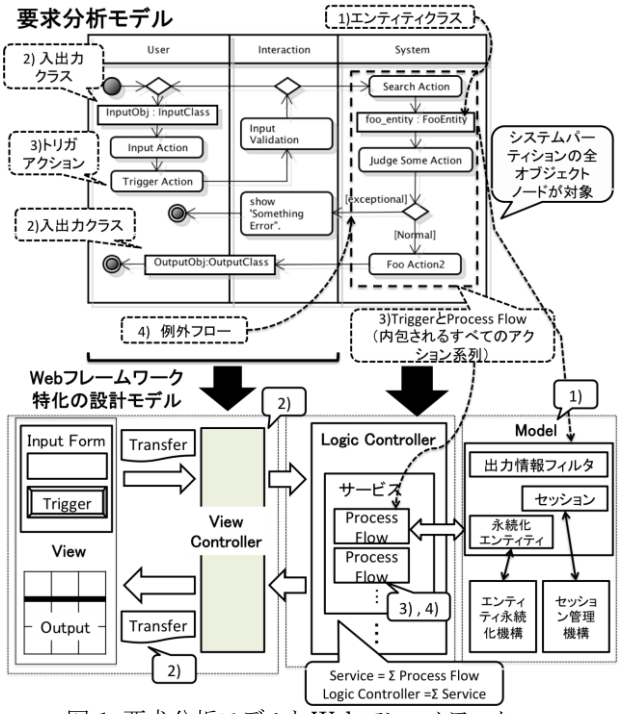

図 1:要求分析モデルと Web フレームワーク設計モデルの対応関係

本手法では、図 1 下段に示すように、Model に定義されるビジネスロジックを用いて、正常・例外・代替のロジックフローを記述した Logic Controller、フレームワークの View に依存する処理であるページ遷移処理・View へのデータ送受信・入力項目の検査を行う View Controller の 2 種の Controller を定義する。

Logic Controller は、トリガに対応する入出力処理単位を表す Process Flow をサービス単位に構成する。アクティビティ図のフローを踏襲し、この Process Flow としてコード化することで、機能要件に関する要求仕様と設計仕様を対応付けている。

View Controller は、Web UI ではページ読み込みとトリガ実行が動作起点となる事から、View Controller 及び View に動作起点に対応する記述を定義し、Logic Controller の実体である Process Flow を呼ぶことで要求仕様と設計仕様を対応付ける。

分離した Controller の機能を繋ぐ為、入出力クラスからトリガ部分を除く入出力項目に等しくなる定義されたデータ転送クラス(Transfer クラス)を用いて、View Controller と Logic Controller 間のデータ授受を行う。

Model に関しては、設計では複数ユーザのシステムとのイン

タラクションと、ネットワークを介した個々のユーザのデータの授受を考慮する必要がある為、システムパーティションにあるオブジェクトノードに(1)データベースへの永続化、(2)インスタンスの状態識別、(3)画面遷移間の値の保持、(4)出力情報のフィルタの 4 つの役割を決定する。

4. 適用事例

我々のモデル駆動要求分析手法を用いて、ソフトウェア開発演習用のグループワーク支援システム(GWSS)の開発を行った。規模はユースケース数 12 個、ソースコード行数 4946 行、永続化エンティティ数 16 個である。開発期間は約 25 日間、開発メンバは Web システム開発経験のある大学院生 2 名、使用した Web フレームワークは、Java 言語であり MVC 型フルスタックが特徴の Play! Framework[4]である。

4.1. ケース・スタディ

GWSS の機能である掲示板の閲覧に着目してモデルとソースコードのトレーサビリティを検証する。

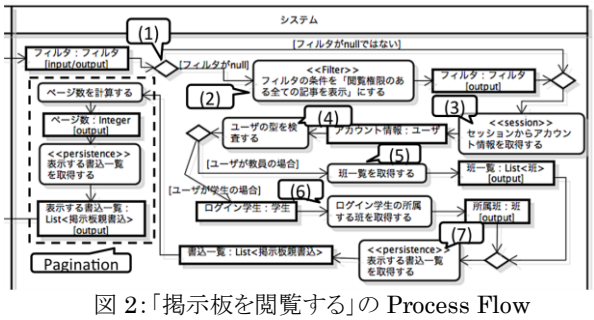

図 2:「掲示板を閲覧する」の Process Flow

表 1:「掲示板を閲覧する」Process Flow のコード

```
private static Map<String, Object> viewBBSProcess(Filter f) {
  Map<String, Object> rtnValue = new HashMap<String, Object>();
  if (f == null) //(1)
    f = Filter.getShowAllPosts();  //(2)
  User user = SessionManager.getUser(); //(3)
  if (user instanceof Student) { //(4)
    Group group = ((Student) user).getMyGroup(); //(5)
    rtnValue.put("group", group);
  } else if (user instanceof Teacher) { //(4)
    List<Group> allGroups = Group.findAll(); //(6)
    rtnValue.put("groups", allGroups);
  }
  List<ParentPost> postsByFilter = ParentPost.getPostsByFilter(f); //(7)
  rtnValue.put("posts", postsByFilter);
  rtnValue.put("filter", f);
  return rtnValue;
}
```

図 2 に「掲示板を閲覧する」の要求分析モデル中の Process Flow を示す。表 1 に「掲示板を閲覧する」の Process Flow の Java の実装コードを示す。図 2 中の吹き出しの番号と表 1 内の文末の番号が対応している。クラスの対応関係は、Filter クラスはフィルタ、User はユーザ、Group は班、ParentPost は、掲示板親書込である。要求分析モデルの(1)～(7)に対して、ソースコードの(1)～(7)が対応し、機能要件に関する要求仕様の振る舞いが定義できている。

図 3:モデルと実装が対応付かない記述例

設計での考慮点は、(A)開発言語やフレームワーク API の利用、(B)フレームワークが提供するモジュール利用がある。(A)に関しては、図 3 に示す様に時刻の取得は API を用いる為、コード中のステップとして明記する必要がなく、掲示板親書込生成時に取得した。(B)に関しては、図 2 の破線箇所はページング機構を表している。それには、フレームワークが提供するページング機構を利用し View Controller に適用した。

システムパーティションのオブジェクトノードの役割として図 2 中に示す様に、ステレオタイプとして記述した。4 つの役割を決定した根拠を以下に示す。

(1)データベースへの永続化:サービスの継続的運用の為に必要なデータを対象とする。(<<Persistence>>)

(2)インスタンスの状態識別:サービス利用の為に、あるエンティティの状態を識別する必要が有る時にセッション変数を利用する。(<<Session>>)

(3)画面遷移間の値の保持:Process Flow に跨るデータの受け渡しを目的としてセッション変数を利用する。

(4)出力情報のフィルタ:Process Flow の入力や出力となるオブジェクトノードを対象に画面出力情報を制御する為にセッション変数や GET メソッドを用いる。(<<Filter>>)

4.2. 使用性向上に関する考察

画面遷移に伴って発生する再読込が煩わしく、使用性が低下することがある。そこで、要求分析モデルにおいて、ユーザからの入力トリガに基づく入力の内、その出力が入力元の出力と同一である場合には、非同期処理を用いることで再読込を不要にすることができる。

今回、Web アプリケーションの使用性向上の為に、Web の汎用的な UI を統合的に扱える CSS フレームワークである Twitter Bootstrap[5]を用いて、要求分析モデルのアクション系列とオブジェクトノードの分析から下記の UI を容易に実装することができた。

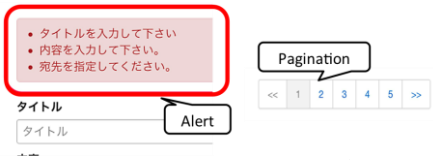

図 4:Alert と Pagination の例

図 4 右の Pagination は、システムからの出力がコレクション型の要素で、その要素数が単調増加する場合に適用すると、表示件数が膨大になるデータに対する有効な実装手段となった。また、要求分析モデルでは、例外を不変条件による例外とシステム実行上の例外の 2 つに明確に区分できる為、ユーザへの例外通知は、図 4 左の Alert が適用できた。

5. まとめ

要求分析モデルから Web フレームワーク設計モデルへのマッピングルールを提案し、その提案手法を用いて実システムに適用した。その結果をモデルとソースコードの観点からトレーサビリティを示し、UIの使用性向上箇所をモデルから発見できた。今後は、Web UI のフレームワークを参考に Web UI メタモデル構築し、UI モデル、要求分析モデル、Web フレームワーク設計モデルの 3 者間の関係を考察したい。

参考文献

[1]小形真平,松浦佐江子: UML 要求分析モデルからの段階的な Web UI プロトタイプ自動生成手法,コンピュータソフトウェア,Vol.27, No.2,pp.14-32(2010)

[2]Hirotaka OKUDA, Shinpei OGATA, Saeko MATSUURA : Mapping Rule Between Requirements Analysis Model and Web Framework Specific Design Model, Knowledge-Based Software Engineering 2012, Vol.240,pp.207-216(2012)

[3]JJ Cadavid, DE Lopez, JA Hincapié, JB Quintero: A Domain Specific Language to Generate Web Applications, http://www.academia.edu/916423/A_Domain_Specific_Language_to_Generate_Web_Applications

[4]Play! Framework, http://www.playframework.org/

[5]Twitter Bootstrap, http://twitter.github.com/bootstrap/