3L-5

動的モデルに着目した Rich Internet Application の Round-Trip Engineering 支援

林千博† 名倉正剛‡ 高田眞吾†

†慶應義塾大学理工学部 ‡株式会社日立製作所

1 はじめに

現在 Web アプリケーション (WA) が普及している．一般的に WA 開発では，モデリングとコーディングを繰り返しながら開発する反復型開発が用いられている．しかし，小規模な更新ではモデルの更新を行わずコードを直接更新する場合が多い．そのため，WA 開発では設計段階で作成したモデルと実装されたコードの間で頻繁に不整合が生じ，開発効率が低下するという問題点がある．

この問題点を解決するために，モデルとコードの整合性を常に保ちながら開発を行う Round-Trip Engineering (RTE) 支援ツールが提案されている [4]．このツールを利用することで常にモデルとコードの整合性を保ったまま開発できるため，開発者はモデリングとコーディングを自由に反復することができる．

しかし，このツールを，近年普及している RIA の開発に適用した場合，非同期通信を扱うことができないという問題がある．これは，このツールがサーバ側のコードに対応しており，RIA の非同期通信やクライアントロジックを扱うことができないからである．また RIA を対象にしたその他の既存のモデル変換手法 [1] では，一般的にモデルとコードのどちらかへの変換しかできない．

そこで本研究では，非同期通信を含むソフトウェアの振る舞いを表すモデル (動的モデル) を利用することによって RIA を対象に RTE を実現するための開発支援ツールを提案する．このツールを利用することにより，RIA を利用した WA 開発において，モデルとコード間の整合性を保ったまま，開発を行うことを可能にする．

2 関連研究

Amalfitano らは Ajax アプリケーションを動的解析し，DOM の変化から有限状態オートマトンを作成する手法を提案した [1]．しかしこの手法は非同期通信を表現できるモデルを使用していない．また，この手法はコードからモデルへの変換のみにしか対応していないため，RIA の反復型開発に用いるのは難しい．

動的モデルを用いて WA の RTE を行う手法に Imazeki らの研究がある [4]．しかし Imazeki らの研究はサーバ側のロジックのみに対応し，RIA の特徴である非同期通信やクライアントロジックを扱うことができない．

RTE 支援ツールに EclipseUML[7] や Together[2] などがある．しかしこれらのツール は対応するモデルがクラス図のみなので，非同期通信を含む制御フローを表現することはできない．

Round-Trip Engineering of Rich Internet Applications Focusing on Dynamic Models

Chihiro HAYASHI†, Masataka NAGURA‡ and Shingo TAKADA†

†Faculty of Science and Technology, Keio University

‡Hitachi, Ltd.

3 提案: RIA を対象にした RTE 支援

本論文では動的モデルに着目した Rich Internet Application の Round-Trip Engineering 手法及び支援ツールを提案する．本提案は MVC パターンに基づき Google Web Toolkit (GWT) により作成された Ajax アプリケーションを対象とし，シーケンス図，ステートマシン，ソースコード間の RTE を支援する．

RIA の RTE を行うモデルとしてシーケンス図とステートマシンに注目した理由は，本提案はオブジェクト指向言語である GWT に基づいている為，モデルには UML が適しているからである．さらに第 1 章でも述べたが，RIA は振る舞いが複雑であり，RIA のモデリングには動的モデルを用いるべきである．以上の理由から UML の動的モデルであるシーケンス図とステートマシンを用いて RIA を対象とした RTE 手法を提案する．

本提案の特徴として非同期通信を含む RIA 全体の RTE を行える点が挙げられる．本提案の RTE の手法は Imazeki らの提案 [4] を RIA に適用できるように拡張したものである．具体的な拡張点としては Imazeki らの提案ではサーバのみを扱っていたが，本提案ではクライアントと非同期通信を記述できるようモデルを拡張し，拡張したモデル間のモデル変換を行えるようにした．これにより開発者はモデリング段階とコーディング段階を自由に反復することができる．

3.1 モデルの拡張

本節では，RIA に適合するよう拡張したモデルについて述べる．なお，ステートマシンの拡張に関しては，コントローラと類似しているので省略する．

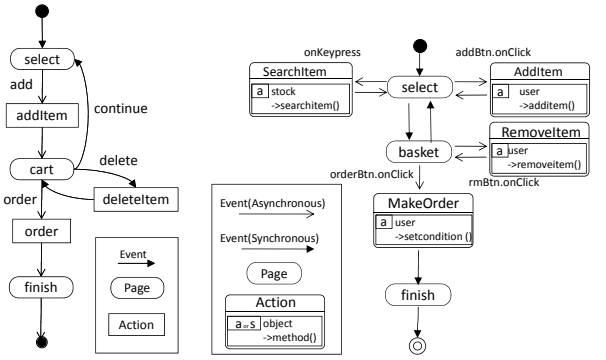
3.1.1 コントローラモデル

コントローラモデルとは Imazeki らが定義したモデルであり，MVC パターンにおいてビューとモデル間の制御を行うコントローラの振る舞いを状態遷移図で表したモデルである．コントローラモデルからは，ビューのイベントとモデルのアクションの関係がわかる．

本提案におけるコントローラモデルの例を図 1 に示す．これは RIA の非同期通信に関しては，矢尻の形を変更することで対応している．また，状態内に記述されている通信の非同期と同期の区別については，状態内のアクション言語 [6] を box で囲み，a (Asynchronous) と s (Synchronous) を振ることで対応している．

3.1.2 シーケンス図

シーケンス図において RIA を表現できるように拡張した点について述べる．これは非同期メッセージに関しては，コントローラモデルと同様に矢尻の形を変更することで対応している．また，ライフラインがクライアント側とサーバ側のどちらに属すのかを区別するために，ライフラインに c (Client) と s (Server) を振ることで対応している．さらに，非同期通信中にクライアントの処理や他の非同期通信を実行することも可能である．これに関しては par により同時に起こる可能性のあるアクションを表記する．



(a) Imazeki らの提案　　(b) 本提案

図 1: コントローラモデル

3.2 Round-Trip Engineering

本提案の特徴は，非同期通信を含む RIA 全体の RTE を行う点にある．本提案は以下のモデル変換により RTE を実現する．モデル変換 1 から 3 は Imazeki らの手法を使用しており，モデル間の変換の手法に関しては大きな変更点はない．よって本節では変換 4 及び変換 5 についてのみ述べる．

1. シーケンス図からステートマシンの生成
2. ステートマシンからシーケンス図の生成
3. ソースコードからシーケンス図の生成
4. ステートマシンからソースコードの生成
5. コントローラモデルから非同期通信コードの生成

3.2.1 ステートマシン → ソースコード

本提案で使用する動的モデルは，プログラムのクラスの構造に関する情報を保持していない．Imazeki らはこの問題に対して，ステートマシンとソースコードの整合性の点検を行い，不整合が生じていた場合には警告を出すに止めているため，開発者がソースコードを直接修正する必要がある．

しかし RIA を対象とする場合，非同期通信に関する振る舞いが複雑であるため，開発者が直接コードの修正を行うのは難しい．そこで Hettel らの提案した Abductive Approach[3] を用いてステートマシンの変更点をソースコードへ反映する．

まずは Abductive Approach によりプログラムのクラスの構造の候補を作成する手順を示す．最初にソースコードを解析し，クラス図の作成を行う．次に Tefkat[5] によりステートマシンとクラス図の対応関係を把握し，ステートマシンの変更箇所を基にクラス図の変更する可能性のある箇所を求める．このとき，変更の副作用として必要なクラスやメソッドが消えてしまうことがある．クラス図の変更箇所ごとに，この副作用をステートマシンと比較しながら明確にし，副作用により消えた箇所を復元することのできる変更箇所を求める．この手順によりクラス図の候補が複数個作成される．

次に開発者がクラスの構造を決定する支援のために，候補のランキングを行う．このとき，元のクラスの構造に近いものを高いランクとする．

これにより動的モデルを用いて表現した非同期通信による複雑な振る舞いをコードへと反映することができる．

3.2.2 コントローラモデル → 非同期通信コード

ここでいう非同期通信コードとは，非同期通信を受信するサーバ側のコード，及び非同期通信を送信するクライアント側のコードのことである．

RIA は非同期通信により複雑な振る舞いをするので，非同期通信コードを直接コーディングすることは難しい．そこで本提案で使用するモデルを基に非同期通信コードを自動生成する手法を提案する．

まずは非同期通信を受信するサーバ側のコードの自動生成の方法を示す．サーバ側は Imazeki らの提案と同様に，コントローラの情報を基にコードを自動生成する (コントローラコード)．ただし，本提案は GWT に基づき実装された Ajax アプリケーションを対象とし，本提案のコントローラコードはサービスインタフェースとして，HttpServlet のサブクラスである RemoteServiceServlet を継承して実装する．

次に非同期通信を送信するクライアント側のコードの自動生成の方法を示す．これは非同期通信を送信するクラスのステートマシンを第 3.2.1 項で説明した手順を用いてプログラムのクラス構造を決定し，ステートマシンに記述された非同期通信の部分から，非同期通信の送信部と Callback 関数部のコードを作成する．次にコントローラのイベントの情報を読み取り，イベント駆動部のコードを作成する．これによりクライアントの非同期通信に関わる部分のコードを自動生成することができる．

上記の非同期通信コードの自動生成により，開発者は非同期通信による複雑な振る舞いを意識せず開発を行うことができる．

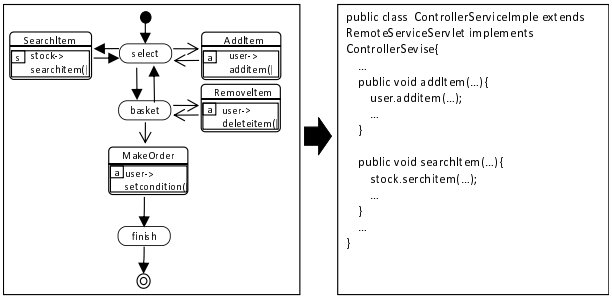


図 2: コントローラモデル→コントローラコード

4 結論

動的モデルを用いた RIA の RTE 手法を提案し，実装した．これにより RIA 開発の問題点を解決し，開発効率の向上を期待できる．

参考文献

[1] D. Amalfitano, A. R. Fasolino and P. Tramontana: “An Iterative Approach for the Reverse Engineering of Rich Internet Applications User Interfaces”, Proc. of ICIW, pp. 401-410, 2010.

[2] Borland: “Borland Together Technologies”, http://www.borland.com/us/products/together/, (参照 2011-1-14).

[3] T. Hettel, M. Lawley, and K. Raymond: “Towards Model RTE: An Abductive Approach”, Proc. of ICM, pp. 100-115, 2009.

[4] Y. Imazeki, S. Takada, and N. Doi: “Round-Trip Engineering of Web Application Focusing on Dynamic Models”, Proc. of ICEIS, pp. 228-233, 2008.

[5] M.J. Lawley and J. Steel: “Practical Declarative Model Transformation With Tefkat”, Model Transformations in Practice Workshop, LNCS Vol. 3844, pp. 139-150, 2006.

[6] S. J. Mellor and M. J. Balcer: “Executable UML - A Foundation for Model-Driven Architecture”, Addison Wesley, 2002.

[7] Omondo: “EclipseUML”, http://www.eclipseuml.com/, (参照 2011-1-14).