B-006

代数的手法によるWebアプリケーションフレームワークの評価
Evaluation Method for Web Application Frameworks based on Algebraic Approach

加賀 周 † 酒井 三四郎 † 富樫 敦 ‡
Shu Kaga Sanshiro Sakai Atsushi Togashi

1. はじめに

近年、Webアプリケーションの開発が盛んに行われている。その開発はWeb特有の制限が多く、通常のアプリケーションに比べ開発効率が良いとはいえない。そのため、Javaにおける開発では、MVCアーキテクチャ[1]やそれを元にしたWebアプリケーション用のフレームワークが多種開発されている。

しかし、数多く存在するWebアプリケーションフレームワークを比較検討する際の評価方法は広く知られていない。本研究では代数的な形式化手法に基づいた評価方法を提案する。形式化の方法とテストケースを示した上で、Java用のWebアプリケーションフレームワークとしては有名なStruts[2]と自作のPlumo[3]を評価する。

2. Webアプリケーションの形式化

本研究の提案する評価方法は、処理の流れを代数的な形式化手法で表現する。典型的なWebアプリケーションは、

$$\mathrm{A} = \mathrm{A}(0)$$
$$\mathrm{A}(u) = \sum_{x \in \mathrm{D}(u)} \mathrm{req}(x).\mathrm{exec}(x).\mathrm{generate}(y).\overline{\mathrm{d}(z)}.\mathrm{A}(z)$$

と表現される。ここで、$y = \mathrm{exec}(x)$ であり、$z = \mathrm{generate}(y)$ である。なお、代数式中の exec は実行アクションであるが、$y = \mathrm{exec}(x)$ での exec は、x に依存する関数である。これは他の式も同様である。また $\mathrm{D}(u)$ をパラメタ u に依存した、リクエスト等の取りうる値の集合とし、$\mathrm{D} = \mathrm{D}(0)$ とする。この式を図に示すと図1のようになる。

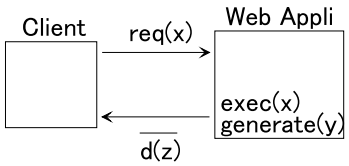

図 1: Webアプリケーション

まず、クライアントからリクエスト x が送られ、それに対応した何らかの処理（$\mathrm{exec}(x)$）が行われる。そこから表示すべきページ y が決まるので、そのページを生成（$\mathrm{generate}(y)$）する。生成されたページ z はクライアントに返る。次に発生するリクエストの集合 $\mathrm{D}(u)$ はクライアントに返ったページから予想可能である。この動作を繰り返すことでアプリケーションが動作する。

†静岡大学
‡宮城大学

なお、ここでは簡単のためブラウザの戻るボタンやリロードは考えない事とする。

このようにフレームワークを表現することで、フレームワークの処理構造についての評価が可能になる。ここでは例としてアプリケーションの再利用に関する評価を扱う。

3. 評価

実際に既存のWebアプリケーションフレームワークであるStrutsと研究過程で制作したPlumoについて、アプリケーションの再利用評価を行う。

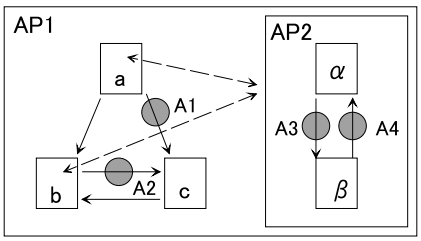

図 2: テストケース

テストケースとして、図2のように3ページからなるアプリケーション AP1 を考える。AP1 では2箇所で何らかの処理が行われる。AP1 を開発するにあたって、同じフレームワークで作成された既存のアプリケーション AP2 をページ1と2において組み込む。AP2 で行われる処理結果は AP1 から自由に読みとれるものとして、できる限り AP2 を修正することなく AP1 の一部として動作できるか評価を行う。

3.1 Struts

Struts は Apache Jakarta プロジェクトの成果物の一つである。Webアプリケーションフレームワーク自体が持つ機能は限られているが、学習が容易である。

分析の結果、Struts の基本式は次のようになった。

$$\mathrm{A} = \mathrm{A}(0)$$
$$\mathrm{A}(u) = \left(\sum_{x \in \mathrm{D}(u)} \mathrm{req}(x).\mathrm{action}(f(x)).\mathrm{JSP}(y).\overline{\mathrm{d}(z)}.\mathrm{A}(z) \right) + \left(\sum_{x \in \mathrm{D}(u)} \mathrm{req}(x).\mathrm{JSP}(x).\overline{\mathrm{d}(z')}.\mathrm{A}(z') \right)$$

ここで、$y = \mathrm{action}(f(x))$ であり、$z = \mathrm{JSP}(y)$ である。

これを元に、テストケースを形式化する。まず、組み

込まれるアプリケーションである AP2 を示す。

$$
\begin{aligned}
\mathrm{AP2} &= A(0)\\
\mathrm{A}(2) &= \mathrm{req}(A3).\mathrm{action}(f(A3)).\mathrm{JSP}(\beta).\overline{\mathrm{d}(1)}.\mathrm{A}(1)\\
\mathrm{A}(1) &= \mathrm{req}(A4).\mathrm{action}(f(A4)).\mathrm{JSP}(\alpha).\overline{\mathrm{d}(2)}.\mathrm{A}(2)
\end{aligned}
$$

AP2 を組み込んだアプリケーションである AP1 を示す。

$$
\begin{aligned}
\mathrm{AP1} &= A(0)\\
\mathrm{A}(1) &= \mathrm{req}(a).\mathrm{JSP}(a).(\mathrm{JSP}(\alpha)+\mathrm{JSP}(\beta)).\overline{\mathrm{d}(2)}.\mathrm{A}(2)\\
\mathrm{A}(2) &= \mathrm{req}(A1).\mathrm{action}(f(A1)).\mathrm{JSP}(c).\overline{\mathrm{d}(3)}.\mathrm{A}(3)\\
&\quad +\mathrm{A}(3)+\mathrm{A}(5)+\mathrm{A}(6)\\
\mathrm{A}(3) &= \mathrm{req}(b).\mathrm{JSP}(b).(\mathrm{JSP}(\alpha)+\mathrm{JSP}(\beta)).\overline{\mathrm{d}(4)}.\mathrm{A}(4)\\
\mathrm{A}(4) &= \mathrm{req}(A2).\mathrm{action}(f(A2)).\mathrm{JSP}(c).\overline{\mathrm{d}(3)}.\mathrm{A}(3)\\
&\quad +\mathrm{A}(5)+\mathrm{A}(6)\\
\mathrm{A}(5) &= \mathrm{req}(A4).\mathrm{action}(f(A4)).\mathrm{JSP}(\alpha).\overline{\mathrm{d}(6)}.\mathrm{A}(6)\\
\mathrm{A}(6) &= \mathrm{req}(A3).\mathrm{action}(f(A3)).\mathrm{JSP}(\beta).\overline{\mathrm{d}(5)}.\mathrm{A}(5)
\end{aligned}
$$

このアプリケーションには問題がある。AP2 に関するリクエストを発生させた時、ページ a、b に戻ってくることができない。Struts では AP2 を改変なしに再利用することができないといえる。

3.2 Plumo

Plumo は研究の過程で開発した Web アプリケーションフレームワークである。軽量であるが、機能は Struts にも及ばない。改善すべき点は多い。

Plumo は次の式で表現できる。

$$
\begin{aligned}
\mathrm{A} &= \mathrm{Get}(0)\\
\mathrm{Get}(x) &= \mathrm{get}(x).\mathrm{OL}(f(x)).\mathrm{Bind}(g(x)).\overline{\mathrm{d}(x)}.\\
&\qquad \left(\mathrm{Post}(x)+\left(\sum_{u\in c(x)}\mathrm{Post}(u)\right)\right)\\
\mathrm{Post}(x) &= \mathrm{post}(x).\mathrm{doPost}(f(x)).\\
&\qquad \left(\left(\sum_{y\in D(x)}\overline{get(m(y))}\mid \mathrm{Get}(m(y))\right)\backslash\mathrm{get}\right)\\
\mathrm{OL}(x) &= \mathrm{onLoad}(x).\left(\sum_{u\in c(x)}\mathrm{OL}(u)\right)\\
\mathrm{Bind}(x) &= \mathrm{bind}(x).\left(\sum_{u\in c(x)}\mathrm{Bind}(u)\right)
\end{aligned}
$$

ここで、$y=\mathrm{doPost(f(x))}$ である。関数 $c(x)$ は x に対する子孫アプリケーションの集合を返す関数であり、関数 $m(x)$ は x の最も親にあたるアプリケーションを返す関数である。

これを元に、テストケースの形式化を行う。組み込まれるアプリケーション AP2 は次のようになる。

$$
\begin{aligned}
\mathrm{AP2} &= \mathrm{Get}(0)\\
\mathrm{Get}(\alpha) &= \mathrm{get}(\alpha).\mathrm{OL}(\alpha').\mathrm{Bind}(\alpha'').\overline{\mathrm{d}(\alpha)}.\mathrm{Post}(\alpha)\\
\mathrm{Post}(\alpha) &= \mathrm{post}(\alpha).\mathrm{doPost}(\alpha').\\
&\qquad ((\mathrm{get}(m(\beta))\mid \mathrm{Get}(m(\beta)))\backslash\mathrm{get})\\
\mathrm{Get}(\beta) &= \mathrm{get}(\beta).\mathrm{OL}(\beta').\mathrm{Bind}(\beta'').\overline{\mathrm{d}(\beta)}.\mathrm{Post}(\beta)\\
\mathrm{Post}(\beta) &= \mathrm{post}(\beta).\mathrm{doPost}(\beta').\\
&\qquad ((\mathrm{get}(m(\alpha))\mid \mathrm{Get}(m(\alpha)))\backslash\mathrm{get})
\end{aligned}
$$

AP1 は AP2 を組み込むことを意識した上で、関数 m と c の定義を行えばよい。

$$
\begin{aligned}
\mathrm{AP1} &= \mathrm{Get}(0)\\
\mathrm{Get}(a) &= \mathrm{get}(a).\mathrm{OL}(a').\mathrm{Bind}(a'').\overline{\mathrm{d}(a)}.\\
&\qquad (\mathrm{Post}(a)+\mathrm{Post(AP2)})\\
\mathrm{Post}(a) &= \mathrm{post}(a).\mathrm{doPost}(a').(((\mathrm{get}(m(b))\mid \mathrm{Get}(m(b)))\\
&\qquad +(\mathrm{get}(m(c))\mid \mathrm{Get}(m(c))))\backslash\mathrm{get})\\
\mathrm{Get}(b) &= \mathrm{get}(b).\mathrm{OL}(b').\mathrm{Bind}(b'').\overline{\mathrm{d}(b)}.\\
&\qquad (\mathrm{Post}(b)+\mathrm{Post(AP2)})\\
\mathrm{Post}(b) &= \mathrm{post}(b).\mathrm{doPost}(b').((\mathrm{get}(m(c))\mid \mathrm{Get}(m(c)))\backslash\mathrm{get})\\
\mathrm{Get}(c) &= \mathrm{get}(c).\mathrm{OL}(c').\mathrm{Bind}(c'').\overline{\mathrm{d}(c)}.\mathrm{Post}(c)\\
\mathrm{Post}(c) &= \mathrm{post}(c).\mathrm{doPost}(c').(\mathrm{get}(m(b))\mid \mathrm{Get}(m(b)))\backslash\mathrm{get})
\end{aligned}
$$

ここで $m(\alpha)=$ AP1、$m(\beta)=$ AP1、$c(a)=$ AP2、$c(b)=$ AP2 である。

Post(AP2) は $\mathrm{Post}(\alpha)$ または $\mathrm{Post}(\beta)$ と同等である。もし、Post(AP2) が発行されても、関数 m によって AP1 のページの Get が発行され、AP1 に戻ってくる事ができる。

以上より、Plumo では AP2 を改変せずに AP1 に組み込むことが可能であることがわかる。

4. まとめと今後の課題

Web アプリケーションフレームワークの形式化手法を示し、これを用いたフレームワークの評価を行い、有効性を確認した。ただし、この評価手法ではフレームワークの処理構造に着眼して評価を行っているため、フレームワークの総合的な評価とはなり得ない。

今後はこの評価手法に用いた形式化を用いて、フレームワーク間移植等の発展を考えていく。

参考文献

[1] JavaServer Pages Model2 architecture
http://www.javaworld.com/javaworld/jw-12-1999/jw-12-ssj-jspmvc.html

[2] The Struts Web Application Framework
http://jakarta.apache.org/struts/

[3] Project Plumo
http://cam.cs.inf.shizuoka.ac.jp/~shu/plumo/

[4] "A Calculus of Communicating Systems", LNCS 92, Springer 1980