Web アプリケーションにおけるデータ依存グラフ

A Data Dependency Graph for Web Applications

黒川 翔*　桑原 寛明†　山本 晋一郎‡　阿草 清滋§

Summary. In this paper, we propose a data dependency graph for web applications. Since web applications consist of many components, data are delivered among these components. Data flow analysis is useful for debugging when errors occurred. Traditional methods are, however, insufficient for data flow analysis in web applications. We define data dependence relations between components and inside components and integrate these relations as data dependency for web applications. Our target web applications are constructed using Servlet and JSP. We show a data dependency graph for a simple web application as example.

1　はじめに

Web アプリケーションはオンライン取引や業務システムなどを提供する手段として広く用いられている．Web を用いたサービスは，サービスの提供者と利用者を直接結びつけるビジネススタイルを急速に広め，インフラストラクチャの一部を形成しつつある．Web アプリケーションの重要度が大きくなると同時にその規模も大きくなっているため，Web アプリケーションの開発や保守は容易ではない．

一般に Web アプリケーションは複数のコンポーネントから構成される．すなわち，各コンポーネント内でデータを加工し，コンポーネント間でデータの送受信を行うことで Web アプリケーションは機能する．コンポーネント間におけるデータの送受信は，データを格納したリクエストと呼ばれるオブジェクトを送受信することで行われる．データが加工されながら複数のコンポーネント間を移動するため，エラーが発生した場合にその位置と原因を特定することは困難であり，生産性向上の足枷となっている．

本稿では，上記の問題を解決するために，サーブレット [2] 及び JSP [3] を用いて作成された Web アプリケーションを対象とするデータ依存グラフを提案する．我々の定式化は，コンポーネント内の依存関係とコンポーネント間をまたがる依存関係の両方を結び付けることができる．具体的には，送受信されるデータを格納しているリクエストに基づいてコンポーネント間のデータ依存関係を定式化した．本論文で扱うコンポーネントは Java プログラムであるため，コンポーネント内の依存関係としては，言語処理系の最適化に用いられている標準的な依存解析手法をそのまま用いることができる．

提案した定式化に基づいて，Web アプリケーションを構成するソースコード群からコンポーネント間の依存関係を抽出する手法とツールの実装について説明する．簡単な Web アプリケーションのデータ依存グラフを構築できることを確認した．

2　Web アプリケーション

Web アプリケーションの利用者は Web ブラウザを利用し HTTP を用いてリクエストを送り，同様に HTTP を用いて結果を受け取る．Web サーバにはページを記

*Sho Kurokawa, 名古屋大学大学院情報科学研究科

†Hiroaki Kuwabara, 名古屋大学大学院情報科学研究科

‡Shinichiro Yamamoto, 愛知県立大学情報科学部

§Kiyoshi Agusa, 名古屋大学大学院情報科学研究科

述するためのリソースが配置される．リソースは静的なリソースと動的なリソースに分けられる．静的なリソースは主に HTML で記述され，内容がそのまま HTTP を用いて利用者に送信される．動的なリソースは ASP，JSP，サーブレット，PHP などを利用して作成されたプログラムである．利用者からのリクエストに応じて実行され，利用者に送信するページを動的に生成する．

動的なリソースからページが生成される際に利用者から入力されたデータが利用される場合が多い．また，ある動的リソースにおいて生成されたデータを次のリクエストに含まれるようにし，利用者に見えない形で送信しておくことで，複数の動的リソース間でデータの送受信が行われている．Web アプリケーションでは様々なデータが利用者のリクエストを介して動的リソースというプログラム間を流れている．そのため，Web アプリケーションにおけるデータフローを捉えるには，リクエストを介するデータフローと動的リソース内のデータフローを結合する必要がある．

本稿では，サーブレット及び JSP で記述された動的リソースから構成された，MVC モデルに基づく Web アプリケーションを対象とする．静的リソースはリソース内でデータの流れを断ち切るため，対象とする Web アプリケーションには含まれないとする．本稿ではサーブレット及び JSP をコンポーネントとする Web アプリケーションだけを扱う．

定義 1 (コンポーネント) サーブレット及び *JSP* は *Web* アプリケーションを構成するコンポーネントである． □

コンポーネント内は通常の Java プログラムであるため従来のデータ依存関係 [6] が利用できる．コンポーネント間におけるデータ依存関係を新たに定義し，2 つのデータ依存関係を統合することで Web アプリケーション全体のデータ依存グラフを定義する．

3 コンポーネント間におけるデータフロー

コンポーネント A から生成されたページを見ている利用者が，次にコンポーネント B を要求しコンポーネント B からページが生成されるとき，コンポーネント A からコンポーネント B へ遷移するという．コンポーネント間におけるデータの送受信はリクエストを介して行われる．コンポーネント A からコンポーネント B への遷移に伴って，データ C が送られる様子を図 1 に図示する．

1. コンポーネント A がデータ C をリクエストへ格納する
2. コンポーネント A からコンポーネント B へリクエストが伝達される
3. コンポーネント B がリクエストからデータ C を取り出す

以上の 3 ステップによりコンポーネント間でデータの送受信が行われる．ここで，リクエストに格納されるデータをリクエストデータと呼ぶ．図 1 ではデータ C がリクエストデータである．

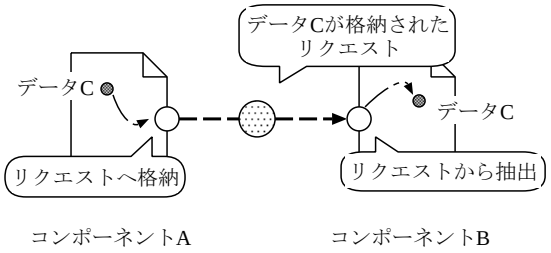

図 1 リクエストによるコンポーネント間のデータフロー

コンポーネント間におけるデータフローは，直観的にはどのコンポーネントからどのコンポーネントへどんなデータが送られるかということである．ここで送られるデータはリクエストデータであるので，コンポーネント間のデータフローは次の 2 つの情報から決定できる．

1. コンポーネント間の遷移に関する情報
2. リクエストデータに関する情報

1. は Web アプリケーションを作成する際に利用したフレームワークに依存する要素である．例えば Struts フレームワーク [1] では，コンポーネントの制御を独自の設定ファイルに記述し，Web アプリケーションの遷移情報を一元的に管理する．我々のグループでは Struts フレームワークに基づく Web アプリケーションの遷移情報を解析するツールを作成した．コンポーネント間の遷移情報の取得方法は，対象とする Web アプリケーションの構成方法に依存するため本稿では特に触れない．コンポーネント間におけるデータ依存関係を定義する前提として，コンポーネント間の遷移情報は既に得られているとする．以下ではリクエストデータについて述べる．

3.1 リクエストデータ

リクエストデータは次の 2 つがある．

1. HTTP リクエストに含まれるパラメータ
2. 任意のオブジェクト

1. は URI クエリ文字列や POST データであり，“?name0=value0&name1=value1&...” のように名前と値を対とした文字列のシーケンスである．すなわち，コンポーネントから HTML のアンカーやフォームを用いて送信されたデータである．サーブレットコンテナはシーケンス内の各対に対し，名前をリクエストデータ名，値をリクエストデータの値としてリクエストにカプセル化する．以下では 1. から生成するリクエストデータをリクエストパラメータと呼ぶ．

2. は ServletRequest インタフェースの以下のメソッドを用いてリクエストに格納あるいは取得するオブジェクトである．

- setAttribute(“name”,obj)
- getAttribute(“name”)

setAttribute メソッドは，任意のオブジェクト obj に対し “name” という名前をつけてリクエストに格納する．このときリクエストデータ名は “name”，その値は obj である．ここで同一リクエストに，それぞれ同名のリクエストパラメータと 2. から生成されるリクエストデータを格納することができるため，名前だけでリクエストデータを特定することはできない．

3.2 コンポーネント変数

コンポーネント間におけるデータフローを特定するために，リクエストデータに関する情報が必要である．3.1 節で述べたように名前だけでリクエストデータがパラメータかオブジェクトかを特定することは不可能である．そのため，名前の有効範囲がパラメータかオブジェクトかを指定する必要がある．そこでリクエストデータ名とリクエストデータ名の有効範囲を組にしたコンポーネント変数を定義する．

定義 2 (コンポーネント変数) コンポーネント変数は，リクエストデータ名 $name$ とリクエストデータ名の有効範囲 $namespace$ の組

$$(name, namespace)$$

と定義する．リクエストデータ名の有効範囲がリクエストパラメータの場合 $namespace = \textit{“Parameter”}$，オブジェクトの場合 $namespace$ の値はオブジェクトのクラス名とする．また，コンポーネント変数 $c = (name, namespace)$，$c' = (name', namespace')$ に対し，

$$c = c' \quad \Leftrightarrow \quad name = name' \wedge namespace = namespace'$$

とする． □

コンポーネントに入力されるリクエストデータに対応するコンポーネント変数をコンポーネント入力変数，コンポーネントから出力されるリクエストデータに対応するコンポーネント変数をコンポーネント出力変数と呼ぶ．

コンポーネント変数に対応するソースコードを表 1 に示す． 表 1 において，re-

表 1　コンポーネント変数

種類	コード	コンポーネント変数
入力	request.getParameter(“name”)	(“name”, “Parameter”)
	request.getParameterValues(“name”)	
	request.getAttribute(“name”)	(“name”, “HttpServletRequest”)
	⟨jsp:useBean id=“name” scope=“request” class=“BeanClass” /⟩	
	sampleBean.getName()	(“name”, “SampleBean”)
	⟨jsp:getProperty name=“SampleBean” property=“name” /⟩	
出力	⟨form action=“url” method=“post”⟩ ⟨input type=“text” name=“name” /⟩ ⟨/form⟩	(“name”, “Parameter”)
	⟨a href=“url?name=value”⟩	
	request.setAttribute(“name”, expr)	(“name”, “HttpServletRequest”)
	sampleBean.setName(expr)	(“name”, “SampleBean”)
	⟨jsp:setProperty name=“SampleBean” property=“name” value=“⟨%= expr %⟩″/⟩	

quest は HttpServletRequest 型のオブジェクト，sampleBean は SampleBean クラスの Bean である．リクエストを用いたコンポーネントにおけるデータの送受信は，表 1 に示したメソッドと HTML タグによって行われる．リクエストに Bean を格納したデータの送受信を扱うため，表 1 には Bean のアクセサメソッドと対応する JSP アクションが含まれる．表 1 に基づきコンポーネントのソースコードからコンポーネント変数を抽出する．

4　データ依存グラフ

4.1　コンポーネント間におけるデータ依存関係

コンポーネント間を伝達するデータはリクエストデータである．リクエストデータを特定するための情報をコンポーネント変数として定義した．そのため，ある 2 つのコンポーネントの間に遷移が存在し，それぞれのコンポーネントのコンポーネント変数の中に等しいものが存在する時，等しいコンポーネント変数に対応するリクエストデータに関して，コンポーネント間のデータ依存関係がある．以下にコンポーネント変数に基づきコンポーネント間におけるデータ依存関係を定義する．

定義 3 (コンポーネント間におけるデータ依存関係) 次の条件を満たすとき，コンポーネント A と B にデータ依存関係があるという．

- コンポーネント A から B への遷移がある
- コンポーネント A におけるコンポーネント出力変数の集合 C_A^{out} とコンポーネント B におけるコンポーネント入力変数の集合 C_B^{in} に対し，$c_a = c_b$ なる C_A^{out} の要素 c_a と C_B^{in} の要素 c_b が存在する □

4.2 コンポーネント内におけるデータ依存関係

本稿で対象としているコンポーネントはJavaプログラムである[1]．Javaプログラムにおける変数の出現単位を式とする．コンポーネント内におけるデータ依存解析は，次の2つのステップで行う．

ステップ1 コンポーネントのソースコードに対して，式におけるデータ依存関係を解析する．

ステップ2 コンポーネント変数と依存関係にある式を特定する．

ステップ2により，コンポーネント内のデータ依存関係にコンポーネント間のデータ依存関係を加えることができる．

式におけるデータ依存関係はOSDG [6] に基づく．OSDGはJavaプログラムを式の粒度で解析する細粒度システム依存グラフである．OSDGでは，変数xに関する2つの式e_1, e_2が参照関係あるいは構成関係にある場合にデータ依存関係があるという．参照関係は次の3つの条件を満たすDef-Useの関係である．

- e_1を評価してxの値が決定される
- e_2でxを使用する
- e_1を含む文からe_2を含む文への制御フローがあり，途中でxの値が変更されない

構成関係はe_1を右辺，e_2を左辺とした代入文を構成することである．

式とコンポーネント変数の依存関係は，コンポーネント変数の抽出元となったステートメントにおいて，リクエストに格納されるデータを表す式，あるいはリクエストから取り出されるデータが代入される式と，抽出されたコンポーネント変数の依存を表す関係である．例えば，ステートメント

```
String expr = request.getParameter("name");
```

からは表1に示されるようにコンポーネント変数 (“name”,“Parameter”) が抽出される．ここで式 `expr` はリクエストから取り出されたデータが代入される式なので，`expr` と (“name”,“Parameter”) の間にデータ依存関係が存在する．

コンポーネント内におけるデータ依存関係の定義を以下に示す．

定義 4 (コンポーネント内におけるデータ依存関係) 式におけるデータ依存関係を$\mathcal{S}_1$，式とコンポーネント変数の依存関係を$\mathcal{S}_2$とする．コンポーネント内におけるデータ依存関係を$\mathcal{S}$とすると，$\mathcal{S} = \mathcal{S}_1 \bigcup \mathcal{S}_2$と定義する． □

4.3 Webアプリケーションにおけるデータ依存関係

コンポーネント間におけるデータ依存関係とコンポーネント内におけるデータ依存関係からWebアプリケーションにおけるデータ依存関係を定義する．

定義 5 (Webアプリケーションにおけるデータ依存関係) *Web*アプリケーションにおけるデータ依存関係$\mathcal{W}$を

$$\mathcal{W} = \bigcup_i \mathcal{S}(\mathcal{RS})^i$$

と定義する．ここで$\mathcal{R}$はコンポーネント間におけるデータ依存関係，$\mathcal{S}$はコンポーネント内におけるデータ依存関係であるとする． □

コンポーネントAの式e_aからコンポーネント内におけるデータ依存関係とコンポーネント間におけるデータ依存関係を推移的にたどることでコンポーネントBの式e_bへ到達できるとき，e_aとe_bにはデータ依存があるという．

定義5に基づきWebアプリケーションにおけるデータ依存グラフを定義する．

定義 6 (Webアプリケーションにおけるデータ依存グラフ) データ依存グラフGを以下のように定義する．

$$\begin{aligned} G &= (V, E) \\ V &= C \cup E_x \end{aligned}$$

[1] コンポーネントがJSPであるとき，JSPソースコードから変換されたJavaソースコードを対象とする．

$$
\begin{aligned}
E &= \{(e_a, e_b) \mid e_a, e_b \in E_x.\ (e_a, e_b) \in \mathcal{S}_1\} \\
&\cup \{(c, e) \mid c \in C.\ e \in E_x.\ (c, e) \in \mathcal{S}_2\} \\
&\cup \{(e, c) \mid e \in E_x.\ c \in C.\ (e, c) \in \mathcal{S}_2\} \\
&\cup \{(c_a, c_b) \mid c_a, c_b \in C.\ (c_a, c_b) \in \mathcal{R}\}
\end{aligned}
$$

ここで C をコンポーネント変数の集合，E_x を式の集合，$\mathcal{R}$ をコンポーネント間におけるデータ依存関係，$\mathcal{S}_1$ を式におけるデータ依存関係，$\mathcal{S}_2$ を式とコンポーネント変数の依存関係であるとする．□

5 データ依存グラフの適用例

フォームから半径を入力し，円周や面積を計算して表示する簡単な Web アプリケーションを考える．この Web アプリケーションを構成するコンポーネントは index.jsp, CalCircle.java，result.jsp である．この Web アプリケーションに対してデータ依存グラフを求める．

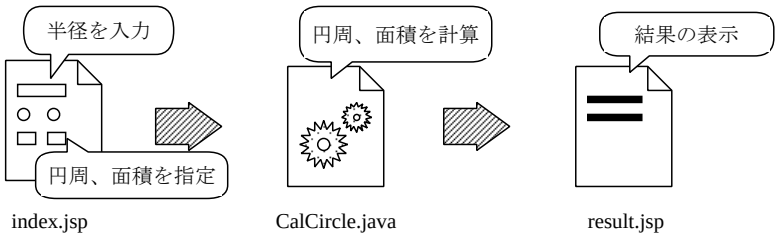

図 2　半径から円周，面積を求める Web アプリケーション

5.1 コンポーネント間におけるデータ依存関係

コンポーネント間のデータ依存関係を求める．この Web アプリケーションには index.jsp から CalCircle.java，CalCircle.java から result.jsp への遷移が存在する．それぞれのコンポーネントからコンポーネント変数を抽出し，定義 3 に基づき index.jsp と CalCircle.java，CalCircle.java と result.jsp の間においてコンポーネント変数を比較する．

```
<form action="CalCircle" method="post">
  <input type="text" name="radius" /><br />                    (1)
  <input type="radio" name="option" value="round" checked />   (2)
  <input type="radio" name="option" value="area" /><br /><br />
  <input type="submit" value="submit" />
</form>
```

図 3　index.jsp ソースコードの一部

図 3，4，5 は図 2 の Web アプリケーションを構成するコンポーネントのソースコードの一部である．図中の (1)〜(12) はコンポーネント変数に対応するコードを表す．それぞれから得られるコンポーネント変数を表 2 に示す．(1)〜(12) のコンポーネント変数を c_1〜c_{12} とすると，表 2 と定義 3 から，コンポーネント間におけるデータ依存関係として $\mathcal{R} = \{(c_1, c_3), (c_2, c_4), (c_8, c_9), (c_5, c_{10}), (c_6, c_{11}), (c_7, c_{12})\}$ が得られる．

```
01:    public void doPost(HttpServletRequest request,
02:        HttpServletResponse response)
03:        throws IOException, ServletException {
04:        RadiusBeans bean = new RadiusBeans();
05:        String radius = request.getParameter("radius");      (3)
06:        double r = Integer.parseInt(radius);
07:        String option = request.getParameter("option");      (4)
08:        double result = 0;
09:        if (option.equals("round")) {
10:            result = 2 * r * Math.PI;
11:        } else {
12:            result = r * r * Math.PI;
13:        }
14:        bean.setRadius(r);                                   (5)
15:        bean.setOption(option);                              (6)
16:        bean.setResult(result);                              (7)
17:        request.setAttribute("beanObj",bean);                (8)
18:    }
```

図 4　CalCircle.java ソースコードの一部

```
<jsp:useBean id="beanObj"
             scope="request" class="RadiusBeans" />          (9)
                                  ⋮
半径 <jsp:getProperty name="beanObj" property="radius" />    (10)
<jsp:getProperty name="beanObj" property="option" />         (11)
: <jsp:getProperty name="beanObj" property="result" />       (12)
```

図 5　result.jsp ソースコードの一部

表 2　コンポーネント変数一覧

コンポーネント	種類	番号	コンポーネント変数
index.jsp	出力	(1)	(“radius”, “Parameter”)
		(2)	(“option”, “Parameter”)
CalCircle.java	入力	(3)	(“radius”, “Parameter”)
		(4)	(“option”, “Parameter”)
	出力	(5)	(“radius”, “RadiusBeans”)
		(6)	(“option”, “RadiusBeans”)
		(7)	(“result”, “RadiusBeans”)
		(8)	(“beanObj”, “HttpServletRequest”)
result.jsp	入力	(9)	(“beanObj”, “HttpServletRequest”)
		(10)	(“radius”, “RadiusBeans”)
		(11)	(“option”, “RadiusBeans”)
		(12)	(“result”, “RadiusBeans”)

5.2　Web アプリケーションにおけるデータ依存グラフ

定義 6 に基づき構築した Web アプリケーションにおけるデータ依存グラフを図 6 に示す．式を表すノードに図 4 における “行番号:式名” を付した．コンポーネント内におけるデータ依存関係 $\mathcal{S}$ は式におけるデータ依存関係 $\mathcal{S}_1$ および式とコンポーネント変数におけるデータ依存関係 $\mathcal{S}_2$ である．関係 $\mathcal{S}_1$ に対応するエッジの張り方を 10

行目の式 r (以下 10:r と表す) を例に説明する．10:r は 06:r を参照しているため参照関係にある．よって 06:r から 10:r へエッジを張る．また 10:r は 10:result へ代入される値を構成しているため構成関係にある．よって 10:r から 10:result へエッジを張る．次に関係 $\mathcal{S}_2$ に対応するエッジの張り方を 17 行目の式 bean (以下 17:bean と表す) を例に説明する．17 行目はコンポーネント変数 (8) に対応するステートメントである．setAttribute メソッドは第二引数の式が表すデータをリクエストに格納するメソッドであるため，コンポーネント変数 (8) と 17:bean の間には依存関係がある．よって 17:bean からコンポーネント変数 (8) へエッジを張る．以上のことを CalCircle.java 内の式およびコンポーネント変数に対して行うことで，コンポーネント内におけるデータ依存グラフを構築できる．

データ依存グラフは，Java の式におけるデータ依存関係に加え，コンポーネント間におけるデータ依存関係も表している．例えば index.jsp でフォームから入力される半径を表すデータ (図 6 中の (1)) は，CalCircle.java に伝達し result の計算に用いられる．そして計算結果 result は result.jsp に遷移するときに CalCircle.java から result.jsp へ伝達される．つまり result.jsp で表示される計算結果は index.jsp で入力される値に依存している．このように，データ依存グラフから図 2 の Web アプリケーション全体のデータ依存関係を把握することができる． そのため，エラー発生時における原因とその位置を特定する等のデバッグに対する支援が期待できる．

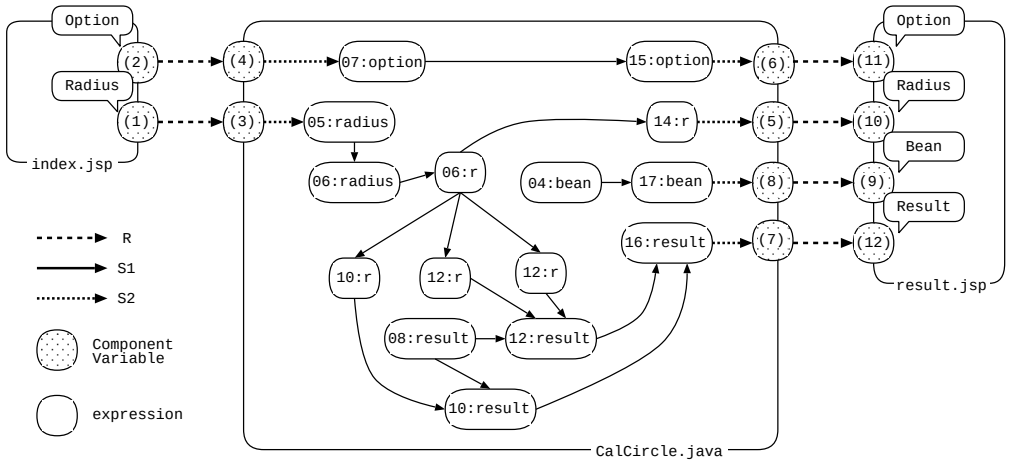

図 6　図 2 の Web アプリケーションにおけるデータ依存グラフ

6　ツールの実装

提案したデータ依存関係を解析するツールを細粒度のソフトウェア・リポジトリに基づいた CASE ツール・プラットフォーム Sapid [7] を用いて実装した．本ツールは Strus フレームワークに基づく Web アプリケーションを対象とする．

主な処理の流れは以下の通りである．

1. Web アプリケーションを構成するコンポーネントをそれぞれ対応するソフトウェアモデルに基づき解析し，リポジトリに解析結果を格納する．
2. コンポーネント間における遷移関係を解析する．具体的には，JSP に記述されるアンカータグとフォームタグからページの遷移先を取得し，struts-config.xml からサーブレットの遷移先を取得できる．
3. リポジトリに格納された解析結果からコンポーネント変数を各コンポーネント

について生成する．
4. 2,3 からコンポーネント間におけるデータ依存関係を解析する．
5. リポジトリに OSDG を適用しコンポーネント内におけるデータ依存関係を解析する．
6. 4,5 からコンポーネント間とコンポーネント内の依存関係を統合し，Web アプリケーションにおけるデータ依存関係を解析する．

本ツールは Web アプリケーションを構成するソースコード群を入力とし，各コンポーネントにおけるコンポーネント変数間の依存関係を出力する．図 7 の Web アプリケーションに対し，実装したツールを用いて解析した結果を表 8 に示す．

表 8 は result.jsp のコンポーネント変数 result と依存関係にある全てのコンポーネント変数を表している．この出力から Web アプリケーションを構成する全てのコンポーネントを範囲とした result の依存関係が分かる．そのため，例えば result が予期せぬ値を保持していた場合に，本ツールを用いて得られるデータ依存関係を参照することで，その原因を特定しやすくなる．本ツールでコンポーネントを跨いだデータ依存関係を自動的に取得することが可能になり，Web アプリケーションの開発や保守作業を支援できる．

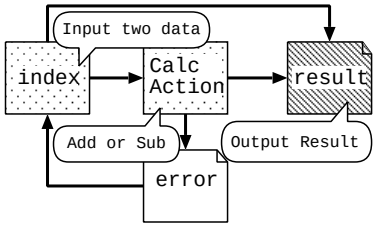

図 7　簡易計算システム

表 8　依存データ項目

ページ	種類	関連ページ	データ
result.jsp	入力	CalcAction	result
CalcAction	出力	result.jsp	result
CalcAction	入力	index.jsp	arg1
CalcAction	入力	index.jsp	arg2
index.jsp	出力	CalcAction	arg1
index.jsp	出力	CalcAction	arg2

7　おわりに

7.1　まとめ

本稿では，Web アプリケーションにおけるデータ依存グラフを定義した．Web アプリケーションにおけるデータ依存関係は，コンポーネント間における依存関係とコンポーネント内における依存関係からなる．2 種類の依存関係を結合するために，データに名前をつけて格納してコンポーネント間で送受信されるリクエストに着目しコンポーネント変数を定義した．コンポーネント変数はコンポーネントへ入力されるデータを表すコンポーネント入力変数と，コンポーネントから出力されるコンポーネント出力変数がある．遷移関係にあるコンポーネント間で等しいコンポーネント出力変数とコンポーネント入力変数が存在するとき，それらのコンポーネント間にはデータ依存関係があるとみなす．また，コンポーネント内におけるデータ依存関係として従来のデータ依存関係を利用し，コンポーネント変数との依存関係を定義した．

例として簡単な Web アプリケーションのデータ依存グラフを示した．また，Struts フレームワークを用いて作成された Web アプリケーションのソースコード群を入力し，その Web アプリケーションのデータ依存関係を解析するツールを実装した．

Web アプリケーションにおけるデータ依存グラフにより，コンポーネントを跨いだデータフローを表現できる．本稿で提案するデータ依存グラフを利用して Web アプリケーションに対しリファクタリングやスライシングを行うことが可能であり，Web アプリケーションの開発や保守に有用である．

7.2 関連研究

Ricca らは Web アプリケーションのスライシングを提案している [4]．スライシングを行うために Web アプリケーションにおける制御依存，データ依存，呼び出し依存を定義している．しかし，データ依存はステートメントレベルでの参照関係で定義されるため，どの変数の値がリクエストに格納されるかなどは分からないという点で本手法と異なる．また，ノードにデータフローと関係の無い HTML 要素を含めているため，生成されるデータ依存グラフは冗長なノードが多い．

Taguchi らは Web 遷移図から Web アプリケーションを生成する手法を提案している [5]．Web 遷移図はページの静的な関係を表すハイパーリンク図とプログラム間のデータフローを表すデータフロー図からなる．Web アプリケーションの遷移中に流れるデータ項目の名前のみに着目してデータフロー図を作成する．プログラム間のデータ依存関係は表現されるが，プログラム内のデータ依存関係は表現されないため，Web アプリケーション全体のデータ依存関係はわからない．

7.3 今後の課題

本稿で提案したデータ依存グラフはリクエストに基づいており，ユーザのアクセスに対してユーザ毎に値を保持するセッション変数に関するデータ依存関係を扱うことはできない．セッション変数はコンポーネント間の遷移経路によって値が異なるため，コンポーネント間の遷移経路を特定する手法が必要であり，セッション変数を考慮したデータ依存グラフについては今後の課題である．

データ依存グラフはサーブレット，JSP をコンポーネントとする Web 層から構築し，ビジネスロジックの実行を行う EJB 層や DBMS などのバックエンド層におけるデータ依存関係を考慮していない．ユーザから JSP へ入力されるデータが，データベースのどのデータと依存関係にあるかを解析するためには，Web 層と EJB 層，バックエンド層を跨いだ依存関係が必要である．データ依存グラフの適用範囲を EJB 層，バックエンド層へと広げることは今後の課題である．

謝辞 本研究を進めるにあたり，熱心に議論して頂いた阿草研究室の皆様に感謝致します．本研究の一部は文部科学省リーディングプロジェクト e-Society「高信頼 WebWare 生成技術」の助成による．

参考文献

[1] The Apache Software Foundation. The Apache Struts Web Application Framework. http://struts.apache.org/.
[2] Sun Microsystems. Java Servlet Technology. http://java.sun.com/products/servlet/.
[3] Sun Microsystems. JavaServer Pages Technology. http://java.sun.com/products/jsp/.
[4] Filippo Ricca and Paolo Tonella. Web Application Slicing. In *ICSM'01*, pages 148–157. IEEE Computer Society, 2001.
[5] Mitsuhisa Taguchi, Tetsuya Suzuki, and Takehiro Tokuda. A Visual Approach for Generating Server Page Type Web Applications Based on Template Method. In *Proceedings of the 2003 IEEE Symposium on Visual and Multimedia Software Engineering*, pages 248–250. IEEE Computer Society, 2003.
[6] 蜂巣 吉成, 山本 晋一郎, 阿草 清滋. オブジェクト指向言語のための細粒度システム依存グラフ. **情報処理学会論文誌**, 40(4):1851–1860, 1999.
[7] 福安 直樹, 山本 晋一郎, 阿草 清滋. 細粒度ソフトウェア・リポジトリに基づいた CASE ツール・プラットフォーム Sapid. **情報処理学会論文誌**, 39(6):1990–1998, 1998.