LMS の拡張を補助する動作検証機能

石川　貴彦, 赤間　清

名寄市立大学 保健福祉学部

北海道大学 情報基盤センター

ishikawata@nayoro.ac.jp

概要：LMS（学習管理システム）のモジュールを開発したり、拡張したりする場合、既存のソースファイルを複製し、インターフェースやデータベースアクセスを修正していくことが、開発者が思いつく手間のかからない方法になる。しかしながら、開発の難しさと教育実践の同時進行との兼ね合いから、動作検証が疎かになりやすい傾向にある。そこで本研究では、開発・実践の進行を妨げずに、CGI の遷移とデータベーステーブルの参照を簡便に検証できる仕組みを考えた。そして、この検証機能を実システムに導入し、検証結果を用いた開発者補助の具体例を示した。

1　はじめに

一般的な LMS は、基盤となるシステムに、学習機能を単位として構成されたサブシステム（モジュール）を複数追加して、運用するシステムが形成される。システムそのものは HTML、CGI、SQL といった主に３つの言語系が駆使されたプログラムの集合であり、システム全体を把握するには、追加した個々のモジュールを網羅的に調べるしか方法がないので、開発や拡張は困難な作業となる。そして、モジュール更新の頻度が高いことも LMS の特徴と言えよう。新たな教育方法のアイデアが浮かび、それを実現するモジュールを作成・追加したとき、実践途中でさらに拡張していく場合もあれば、教育効果が思うように上がらず、以降の使用を断念し別のモジュールに鞍替えするという不安定さもある。

以上のことから、モジュール開発の難しさと教育実践での運用が交錯し、同時進行で開発が行われがちになるため、動作検証が疎かになりやすい。したがって、こうした状況にも対応できる検証方法を与えることが、LMS 開発を容易にする手段になるかもしれないと考えた。具体的には、モジュール内の CGI ファイルを適宜解析し、その解析情報を用いて動作を検証する方法である。これは、テストデータの入力やモジュールの直接実行がないので、簡便性と安全性を期待できる。

本研究では、LMS 開発と教育実践がオンタイムで行われる状況下で、モジュール構成を逐次的に診断し、簡便に動作検証できる機能を構築した。そして、この機能を独自開発 LMS[1]に導入して、開発者補助の具体例を示した。

2　動作検証のための準備

2.1 S 式による CGI プログラムの記述

本研究で用いる独自開発 LMS は、ルール型言語である ET 言語[2]で CGI が記述され、述語と引数の列からなる S 式（アトム）で表される。

（述語　引数$_1$，引数$_2$，…）

述語はユーザが独自に定義できる D アトムと、ET 言語のビルトインである B アトムの２種類を併用する。引数のうち、＊で始まるものは変数である。アトムを実行するには、アトムの置き換え関係を表す ET ルールを記述し、これらルールの適用によって CGI を制御する。つまり本 LMS においては、ET ルールの集合が CGI のプログラムとなり、以下の *R1*～*3* のように記述される。

```
R1 (main), {(read *id *pass)}
      → (getDBdata *id *pass *data),
        (displayWeb *id *pass *data).

R2 (getDBdata *id *pass *data)
      → (sprintf *query "select score from DBtable
                                  where id = '%s';" (*id)),
        (sql:send *query *id *pass *data).

R3 (displayWeb *id *pass *data)
      → (cgi:header), (title "テスト結果"),
        (printf "点数は%d 点です" (*data)), (br),
        (form:begin "nextcgi.eti"),
        (form:submit "次のページ"),
        (form:end), (cgi:footer).
```

R1 は CGI を呼び出す最初のルールであり、中カッコ{ }で囲まれた条件部を満たした場合（ここでは、ID とパスワードを読み込んで認証に成功した場合）、矢印以降のアトムに置き換えるという記述である。*R1* で置き換えられた２つのアトムは、次に *R2* と *R3* にそれぞれ適用し、各ルールにて次の実行が進む。*R2* は代入で SQL 文を生成し、それを発行してデータを取得するルールであり、*R3* は取得したデータを HTML に代入して Web ページを表示するルールである。

このように本 LMS は、HTML、CGI、SQL の３つを S 式表現でなるべく統一している。S 式で記述するとプログラムそのものをデータとして扱いやすくなるので、プログラムを解析する上では都合が良い。また、開発者にとっては、様々なプログラミング言語を使い分けるといった負担軽減の側面もある。

2.2 CGI パスの取得

前節において、本 LMS のプログラムは S 式で記述され、データとして扱えることを述べた。これを活用して解析情報の取得方法を考える。CGI の遷移は、HTML の form タグに記述されたボタンのパスを芋づる式に辿っていけば、その状態を確認することができる。したがって、CGI ファイルから form タグを検出し、そこに記述されているパスを取得すればよい。ただし本 LMS の場合においては、前節 *R3* のように、form:begin アトムの引数が、パスの記述であることが確定しているので、文字列検索をして抜き出すのではなく、アトムの引数を取得すればよい。

2.3 データベーステーブル名の取得

次に CGI が参照するデータベース（以下 DB と略記する）のテーブル情報の取得を考える。これも *R2* のように、sprintf アトムの第２引数に SQL 文が記述されるので、それを解析する（例えば from から where の間に書かれた文字列を抽出する）ことでテーブル名を取得できる。ただし、代入で一旦 SQL 文を生成せずに、直接記述できるものもあり、この場合は sql:send アトムの第１引数に記述されることになる。したがって、テーブル名は sprintf アトムと sql:send アトムを検出し、そこに書かれた SQL 文から文字列検索をして取得することになる。

2.4 パスおよびテーブル情報の取得プログラム

一般的な LMS のモジュールは、学習機能を単位として１つのフォルダで構成されているものが多く、フォルダを配置すればモジュールが追加されるという仕組みを取っている。本 LMS においても同様である。そこで、検証しようとする１つのフォルダを選択し、その中に含まれる全ての CGI ファイルから、リンク先のパスとテーブル名を取得するプログラムを作成した。取得したパス、テーブルは S 式のリスト列として出力し、これらの情報を用いて動作検証を行う。リスト列は以下のように表現する。

((フォルダ CGI$_1$ アクション$_1$ パス or テーブル$_1$)
(フォルダ CGI$_2$ アクション$_2$ パス or テーブル$_2$)…)

リスト列のアクション$_N$（第３引数）は、button か db のどちらかが要素となる。button の場合、第４引数には CGI のパスが記述され、db の場合はテーブル名が記述される。例えば、quiz というフォルダを検索したとき、リストは button と db の場合で以下のように記述される。

```
(("quiz" "question.cgi" button "answer.cgi")
 ("quiz" "question.cgi" db "QuizTable") … )
```

ここまでをまとめ、モジュールの選択からリスト列の生成までの流れを以下に示す。

(1) 検証対象となるモジュール（フォルダ）を選択する。
(2) フォルダ内に含まれる全ての CGI ファイルを検索する。
(3) 各 CGI ファイルから form:begin アトムを検索し、ボタンに書かれた CGI パスを取得する。
(4) 続けて sprintf アトムと sql:send アトムを検索し、DB テーブル名を取得する。
(5) 取得した情報をリスト列で書き出す。

3 動作検証機能

3.1 リンクの検証

CGI を検証する際、まず思いつくのがページ間のリンクであり、特にボタン先のリンク切れや、誤ったページへのリンクが挙げられる。また、前のページに戻るボタンなど細かい実装が疎かになりがちであり、ユーザビリティの点からもリンクの検証は重要である。

そこで、リンクを検証するにあたり、どのよう

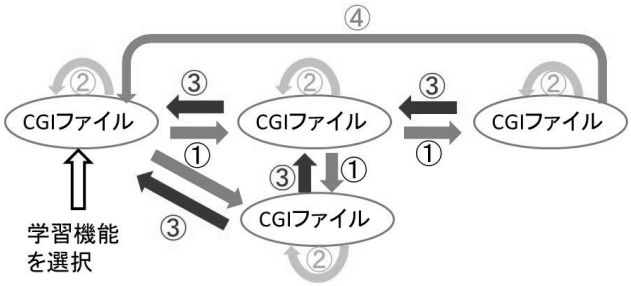


図１　リンク検証のためのパターン図

な状態が望ましいかを考えるため、パターン図（図１）を作成した。まずログイン後のページから学習機能を選択し、１つの CGI を呼び出す。この CGI がモジュールの起点となる。次にリンク先の CGI から１つを選択し、別の機能を操作する（①）か、あるいは DB を更新し自身に再帰する（②）。これらの操作を繰り返して、Web ページを遷移していく。ユーザビリティの面からは、直前の CGI に戻るか（③）、またはページが深くなったときに、起点にジャンプできればよい（④）。

このように４つの検証パターンを、CGI 毎に診断することで、①と②からは Web ページの遷移を、③と④からはユーザビリティを検証することができ、モジュール内のどの CGI でリンク切れが生じているのかを把握しやすくなる。

3.2 ２通りの検証方法

リスト列と検証パターンを用いて、動作検証の方法を考える。単に CGI や DB の関連を目視したい場合には、リスト列から以下の ET ルールに変換し、ET の処理系の中で実行する。これはプロトタイプという位置づけであり、実システムとは切り離された環境でモジュールの検証を行うことができる。

```
(cgi *f *cgi *act *path)
    ⇒ (= *f "quiz"), (= *cgi "question.cgi"),
      (= *act button), (= *path "answer.cgi");
    ⇒ (= *f "quiz"), (= *cgi "question.cgi"),
      (= *act db), (= *path "QuizTable").
```

例えば、上記の ET ルールに対して、(cgi "quiz" "question.cgi" button *path)という問い合わせを与えて実行すると、*path には answer.cgi という答が返る。これは、question.cgi 内にあるボタンを選択すると、answer.cgi にリンクしているということを端的に表している。

もう１つは、リスト列から SQL 文に変換し検証用の DB に登録する方法がある。この DB を用いて Web 上で操作すれば、実システムに近い環境での検証が可能になるので、開発者は実際に操作しながら Web ページの遷移を確認できる。

このように、リスト列を変換して CUI（ET）または Web サイドで検証できる枠組みを与え、開発者の目的に合わせた動作検証が行える。

3.3 Web ユーザインターフェース

Web 上での検証機能をモジュール化し、本 LMS に追加した。この Web インターフェースは、まず図２の画面から開始する。次に 2.4 節の取得プログラムの実行によって、解析情報を取得し終えたモジュール群は検証用 DB に登録され、一覧が表示される。開発者は検証するモジュールを選択すると、中に含まれる全ての CGI ファイルが表示される（図３）。一覧の左端は、CGI が学習機能として紐づけられている状況を示しており、これがモジュールの起点となる。つまり CGI の遷移を検証するには、左端に記載された学習機能から選ぶことになる。そして、起点の１つを選択すると図４が表示される。画面上部のウインドウには、選択した CGI の履歴が書き込まれ、開発者はどのような CGI の辿り方をしたのかを、随時ウインドウで確認する。一覧表の２列目には、3.1 節で示した


CGI・DB情報

検証するLMS内のフォルダを以下から選択してください。

フォルダ	選択
column	選択
job	選択
masters	選択
movie	選択

図２　モジュールの選択画面

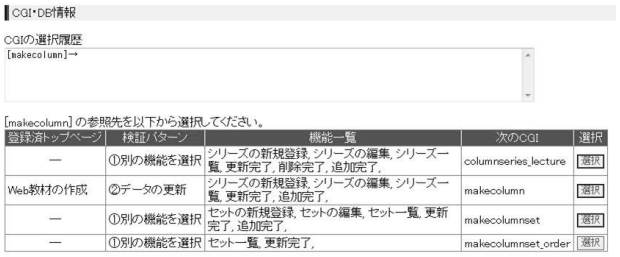

CGI・DB情報

CGIの選択履歴

[column] の参照先を以下から選択してください。

登録済トップページ	検証パターン	機能一覧	次のCGI	選択
—	①別の機能を選択	セット一覧, 課題セットの編集,	addcolumn	選択
—	①別の機能を選択	シリーズの新規登録, シリーズの編集, シリーズ一覧, 更新完了, 削除完了, 追加完了,	columnseries_lecture	選択
—	①別の機能を選択	シリーズ一覧, 追加完了,	linkcolumnset	選択
Web教材の作成	①別の機能を選択	シリーズの新規登録, シリーズの編集, シリーズ一覧, 更新完了, 追加完了,	makecolumn	選択
—	①別の機能を選択	コピー完了, コラムの新規登録, コラムの編集, セット一覧, 完了, 更新完了, 追加完了, 例題の新規登録,	makecolumn_one	選択
—	①別の機能を選択	セット一覧, 更新完了,	makecolumn_order	選択
—	①別の機能を選択	セットの新規登録, セットの編集, セット一覧, 更新完了, 追加完了,	makecolumnset	選択
—	①別の機能を選択	セット一覧, 更新完了,	makecolumnset_order	選択
Web教材の管理	①別の機能を選択	例題集シリーズ, 例題集セット,	setcolumn	選択
Web教材	①別の機能を選択	シリーズ一覧,	viewcolumn	選択
—	①別の機能を選択	例題集の設定,	visible_grp	選択

図３　モジュール内の全 CGI の表示画面


CGI・DB情報

CGIの選択履歴

[makecolumn]→

[makecolumn] の参照先を以下から選択してください。

登録済トップページ	検証パターン	機能一覧	次のCGI	選択
—	①別の機能を選択	シリーズの新規登録, シリーズの編集, シリーズ一覧, 更新完了, 削除完了, 追加完了,	columnseries_lecture	選択
Web教材の作成	②データの更新	シリーズの新規登録, シリーズの編集, シリーズ一覧, 更新完了, 追加完了,	makecolumn	選択
—	①別の機能を選択	セットの新規登録, セットの編集, セット一覧, 更新完了, 追加完了,	makecolumnset	選択
—	①別の機能を選択	セット一覧, 更新完了,	makecolumnset_order	選択

図４　選択した CGI の検証画面


CGI・DB情報

CGIの選択履歴

[addcolumn]→

[addcolumn] の参照先を以下から選択してください。

登録済トップページ	検証パターン	機能一覧	次のCGI	選択
—	①別の機能を選択	コピー完了, コラムの新規登録, コラムの編集, セット一覧, 完了, 更新完了, 追加完了, 例題の新規登録,	makecolumn_one	選択
—	②データの更新	セット一覧, 課題セットの編集,	addcolumn	選択
—	①別の機能を選択	セットの新規登録, セットの編集, セット一覧, 更新完了, 追加完了,	makecolumnset	選択
—	①別の機能を選択	リンク切れの 可能性があります	ProblemSet	選択

※　③前のページまたは④先頭に戻るボタンが実装されていません

[addcolumn] の使用データベース

Columns
Columns_ColumnSet
ColumnSet
problem
Problem_ProblemSet
ProblemSet

図５　パターンの不足を示唆する画面

検証パターンが表示され、個々の CGI においてパターンが複数満たされているかどうかをチェックしている。パターンの不足が見つかった場合には、図５のようにコメントが提示される。例えば、③が満たされない場合は、戻るボタンが実装されていないと提示される。リンク切れについては、この検証の時点で、次に選択できる CGI の機能一覧を先読みしているので、その状況を基にして調査できる。つまり、次の CGI ファイルが見つからず、機能一覧を示せないときには、リンク切れと判断できるので、機能一覧を示す代わりに、図５のようなリンク切れを示唆するコメントを提示する。

4 開発者補助の具体例

Web インターフェースを用いて、稼働中の LMS から得られる検証結果を調査し、さらに、その結果から求まる不備の原因について検討した。

図５では、戻るボタンが実装されていないことに加え、リンク切れも提示されている。注目すべきは、参照している DB テーブルについて、本来用いるデータとは関連がないと思われるテーブル（図５内の下３つ：problem 等）が提示されている点である。このようになった原因は、元が演習問題を作成・編集するための CGI で、それを流用して Web テキストを作成・編集する CGI を開発者が作ったのだが、不要なソースを消し忘れて、元のリンクやテーブルが残存したためである。他にもリンク切れの CGI が見つかったが、この場合は、機能を拡張する予定で開発者はボタンを設置したが、途中でモジュールの開発が滞り、そのボタンがコメントアウトされたまま残存したことが原因であった。

このように、本機能で検証を逐次行うことで、開発者は不備を発見しやすくなり、検証と修正を繰り返してモジュール開発を進めるとよい。これが、開発者補助の一つの方法である。

5 まとめ

本研究では、LMS 開発と教育実践がオンタイムで行われる状況にも対応できるような動作検証の方法を検討するため、モジュール単位で CGI を解析し、解析情報を S 式で表すプログラムを作成した。そして、その情報を用いて検証を行うモジュールを作成、稼働中の LMS に導入し、検証結果を例示した。

LMS のプログラムそのものを、S 式表現の ET ルールで記述したことで、プログラムをデータとして扱いやすくなり、CGI パスや DB テーブルの情報を容易に取得できた。また、これらの情報を用いて動作を検証するため、４つのパターンを導入し、Web 上で逐次診断できるようにした。これにより、参照する DB テーブルの誤りやリンク切れ、戻るボタンの未実装を開発者に提示し、その結果を基にして該当する CGI に潜む不備を特定することができた。

今後も実システムでの検証を重ねていき、他にどのような検証結果が得られるのか、結果からどのような不備の原因が求まるのか等、事例をさらに増やしていき検証の精度を高める。本研究は、平成 24 年度北海道大学情報基盤センター共同研究採択課題として行われたものである。

参考文献

[1] 石川貴彦、赤間清、三浦克宜、「自由学習環境を実現する学習管理システムの構築」、教育システム情報学会 30 周年記念全国大会講演論文集、pp.433-434、2005

[2] 赤間清、 繁田良則、宮本衛市、「論理プログラムの等価変換による問題解決の枠組」、人工知能学会誌、Vol.12、No.2、pp.266-275、1997