LMS の学習機能の洗練を支援する LMS マネージャーの開発

石川　貴彦

名寄市立大学 保健福祉学部

ishikawata@nayoro.ac.jp

概要：LMS などの Web システムは、CGI、データベース、格納データの３つから機能を構成する。その開発過程では、CGI プログラミングやデータベースデザインなど、相互に絡み合って作業が進められるため、拡張や改善の複雑さが伴う。そこで本研究では LMS マネージャーを開発し、学習機能の指定に応じて、関連する CGI やデータベースを自動的に抽出・パッケージ化する枠組みを構築したので報告する。

1　はじめに

Web ベースシステムは、データの入出力やインターフェースをつかさどる CGI（プログラム）、データベース（DB）、ユーザ情報やログといった格納データの主に３つが組み合わさることで機能が成立する。LMS（Learning Management System）の場合、その機能はクイズ、資料配布、動画視聴など、学習機能毎に独立して運用するケースが多く、Moodle[1]ならばモジュールという形でパッケージ化されている。この学習機能の開発過程では、①DB をデザイン、②Web インターフェースおよびデータベースアクセスの CGI プログラムを作成、③データを管理という３つの作業を同時進行で行い、それらは複数の DB や CGI にまたがって実装しなければならない。そのため、新機能の開発や拡張、改善に対して相当の複雑さと労力が伴う。

このように、複雑な開発過程を経て出来上がった LMS を、サーバ移行や他者への提供といった今後を見越して、洗練化されたパッケージとして保存管理しておくことは、LMS 開発の複雑さを緩和する上で考慮すべき１つの事項となる。さらに、各教員の教育内容や方法に応じて、使用する学習機能は異なるため、LMS 全体の管理だけでなく、機能単位でのパッケージ管理が望まれる。

そこで本研究では、開発済の LMS に対して、学習機能毎に CGI、DB、データが１つになったパッケージを自動で構築し、それらを適正管理するための LMS マネージャーを開発した。そして、稼働中の LMS にマネージャーを適用し得られたパッケージを、新規サーバに移行する実験を行い、自動生成の精度について検証した。

2　LMS の概要

2.1 S 式による LMS 開発

筆者は、プログラミング言語 ET[2]を用いて独自 LMS[3]を開発し、これまで全ての機能を ET プログラムで作成した。その ET 言語の記法には S 式が用いられ、述語と引数の列からなるアトムが実行単位となる。引数のうち、* で始まるものは変数であり、述語のみの０引数という場合もアトムになる。また、引数にはリストを含めることができ、Lisp 言語ではリストの car 部と cdr 部をドット対表記で記述するが、ET 言語の場合はドットの代わりに縦棒で区切る。

```
(述語　引数１　引数２　…)
(述語　*A *B　…)
(述語　(*X | *Y) …)
```

そして、アトムの置き換え関係を表したものを ET ルールと呼ぶ。例えば、LMS の CGI プログラムは、以下のようにルールを記述する。

```
(main),
  { (get "id" *ID), (get "passwd" *PASS),
    (check *ID *PASS) }
      → (getDBData *ID *PASS *DATA),
          (displayWeb *ID *PASS *DATA).
```

このルールは、main アトム（ヘッド部）が、中カッコ内に記述された２つの get アトムと check アトム（条件部）を満たしたとき、矢印以降の getDBData アトムと displayWeb アトム（ボディ部）に置き換えるという処理を実行する。実際のルールの動きは、main 実行すると、直前の Web ページから送られたパラメータ、ここでは ID と

パスワードを取得し、それらが正しいものであると判定されると、DB から情報を取得し、Web 上で情報を表示するという流れになる。getDBData アトムと displayWeb アトムは、別のルールでさらに定義する。このようなルール群を１つのファイルで書き表して、CGI を実行するプログラム（ETI）が出来上がる。

CGI を S 式で記述するメリットは、ルールそのものを１つのデータとして扱えることである。そのため、CGI のソースコード（ここでは ETI）を解析するためのプログラムを構築する際には、ルールを以下のようなアトムと見て、S 式の処理を行えばよい。

```
(Rule (main) ((get “id” *ID)(get “passwd” *PASS)
(check *ID *PASS)): ((getDBData *ID *PASS *DATA)
(displayWeb *ID *PASS *DATA)))
```

ルールのヘッド部は必ず１つのみアトムを記述し、条件部とボディ部はそれぞれ０個以上のアトムを記述するので、適用パターンは以下の４つのみとなる。

① (Rule *H (*C1|*X) : (*B1|*Y))
② (Rule *H : (*B1|*Y))
③ (Rule *H (*C1|*X) :)
④ (Rule *H :)

CGI や DB テーブルのパスを含むアトムは、条件部かボディ部にしか書かないので、LMS のプログラム全てを①～④のパターンにマッチングさせ、そこから①③の(*C1|*X)と、①②の(*B1|*Y)に適合するアトム列の各要素をトークンとして解析する。アトムの解析は、筆者らの論文[4]で示した方法に従って CGI パスと DB テーブル名の一覧を取得することができ、これら一覧を元に CGI ファイルを複製し、DB テーブルの構築と移行データを挿入する SQL 文（スクリプト）を生成する。

2.2 LMS のフォルダ構成

本 LMS は、学習機能単位でフォルダ管理する構造であり、eti-bin フォルダ内に各機能を追加して運用する。機能は動画配信（movie フォルダ）、相互評価（checklist）、掲示板（bbs）などをこれまでに制作し、各フォルダ内には ETI ファイル、画像ファイル、スタイルシートなどが含まれる。例えば movie フォルダの中には、12 の ETI ファイルと、８つの画像ファイルが含まれている。dir コマンドで各機能のフォルダを指定すると、ファイル名の一覧を取得できるので、そこから各ファイルを開き 2.1 節に示した①～④のパターンに適用させれば、ETI ファイル間におけるパス関係と、使用 DB テーブルをフォルダ単位で求めることができる。画像ファイル等については、フォルダ一式をまるごと複製する。

3　LMS マネージャー

3.1 マネージャーの概要

前章で示した LMS の概要を元に、CGI、DB、格納データの３つを機能毎にパッケージ化する LMS マネージャーを開発した。このマネージャーも ET 言語で作成し、S 式やリストの処理を容易に行えるようにした。

マネージャーでは、LMS 全体の CGI と DB をパッケージ化するか、構築したいパッケージの機能をユーザが指定し、フォルダ単位でパッケージ化するかを選択できる。LMS 全体をパッケージ化するということは、システムのバックアップを取ることと同等であり、部分的なパッケージとなると、それは他者に提供するためのモジュールという形になる。

3.2 使用 CGI と DB の検出方法

CGI のパッケージ化は、使用中と認定された CGI だけを含める。図１は各 CGI のパス関係を表したものであり、CGI1 を解析すると CGI2 と DB1 にリンクしていることを検出し、CGI2 と DB1 が使用中と判定する。同様に、CGI2 からは CGI1、CGI3、DB2 が、CGI3 からは CGI2、DB3 が導かれ、各 CGI で検出されたリンクを集約し重複を除くと一覧が得られる。例えば、システムの改良のために CGI2 をコピー・温存し、作業にあたるケースがあるが、CGI2-copy は他のどの CGI からも参照されていないので使用外と判定し、パッケージには含まない。したがって、パッケージには CGI

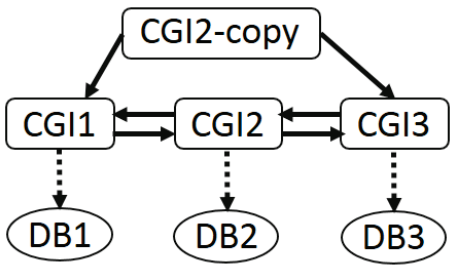

図１　CGI と DB のパス関係

QuizSet(親) ⟵ QuizSet_Quiz(子) ⟶ Quiz(親)

SetID	SetName
1	国語
3	算数

SetID	QuizID
1	1
1	4
3	2

QuizID	Question
1	漢字問題
2	計算問題
4	文法問題

図２　演習問題の格納データ例

QuizSet(親) ⟵ QuizSet_Quiz(子) ⟶ Quiz(親)

SetID	SetName
1	国語
[2]	算数

SetID	QuizID
1	1
1	[3]
[2]	2

QuizID	Question
1	漢字問題
2	計算問題
[3]	文法問題

図３　洗練化された格納データ

1～3 の各ファイルと画像ファイル一式、DB1～3 を構築するための SQL 文、アクセス権限などの情報および格納データが収められる。

3.3 格納データの洗練化

リレーショナル DB の格納データの多くは、ID を主キーとしオートインクリメント（連番）でレコードが追加される。レコードを削除するとその ID は欠番になり、新規データを追加すると、欠番を埋めずに新たな ID を付与する。図２は SetID＝2 のセットと、QuizID＝3 の問題が削除された後の歯抜けの状態を示した例である。このようにテーブルの構造劣化が進むと、徐々に管理の煩わしさやパフォーマンスの低下をもたらす。サーバ移行や他者へのモジュール提供等で格納データを使用する場合は、欠番のない状態で受け渡すことが望ましく、パッケージ化の際には、当該テーブルの ID を付与し直して連番を保持する。それが洗練という意味を表す。図３の四角で囲まれた箇所は、ID を再付与して連番に直した部分であり、この洗練化の方法については、筆者らの論文[5]に掲載しているので、そちらを参照されたい。

4 LMS マネージャーの検証実験

4.1 マネージャーの実行

マネージャーを起動すると、アトムを受け付けるプロンプトが表示され、all (LMS 全体) か、sub（フォルダ毎）のどちらかを入力する。(sub)を与えた場合は、LMS 内のフォルダを探索し、パッケージ化できる一覧を提示する。図４はその状態を表したものであり、ここでは movie フォルダのパッケージ化を指定している。フォルダ指定すると、

```
[DJ>(sub)
------------------------D execution ---------------------
"autocheck"
"bbs"
"checklist"
"column"
"enquete"
"job"
"lib"
"mail"
"masters"
"movie"
"personal"
"program"
"schedule"
"uploader"

target -->"movie"
```

図４　フォルダ選択画面

```
target -->"movie"
File not found.
File not found.
File not found.
("asxview.eti" "st_movieseries_order.eti" "movieview.eti" "st_movieset.eti" "asx
_gen.eti" "movie_item.eti" "movie_log.eti" "movie_itemorder.eti" "st_movie.eti"
"movieseries.eti" "movie_set.eti" "st_movieset_order.eti")
("MovieSet_MovieSeries" "MovieSeries" "MovieItem_MovieSet" "MovieItem" "MovieLog
" "MovieSet")
table
power
foreignkey
view
procedure
function
data
        1 個のファイルをコピーしました。
        1 個のファイルをコピーしました。
        1 個のファイルをコピーしました。
        1 個のファイルをコピーしました。
        1 個のファイルをコピーしました。
        1 個のファイルをコピーしました。
        1 個のファイルをコピーしました。
        1 個のファイルをコピーしました。
```

図５　パッケージ生成の実行画面

3.2 節で示した方法でファイルを検証し、使用中 CGI と DB テーブルが抽出される。図５の５～７行目のリストが、movie フォルダから取得された使用中 CGI ファイルの一覧であり、８～９行目のリストが、参照している DB テーブルの一覧である。10 行目以降は、その DB 一覧に伴うアクセス権限や外部キーなどの自動生成であり、以下のようなスクリプトとして、ファイルに書き出される。

```
// DB テーブル
?- (table "MovieSeries"
        (("id" "smallint identity" 5 0 () PK)
         ("name" "nvarchar" 50 0 () ())
         ("lectureID" "smallint" 5 0 () ())
         ("visible" "nchar" 1 0 "('T')" ()))).
// アクセス権限
?- (auth "MovieSeries" "students" ("SELECT")).
```

格納データについては、SQL 文を引数としたスクリプトが生成される。

```
?- (makeObjects "insert into MovieSeries (name, lectureID, visible) values ('動画 1', 4, 'T');").
```

4.2 生成されたパッケージの検証

マネージャーを既存の LMS に適用し、いくつかのフォルダを指定して、生成されたパッケージの精度について検証した。movie フォルダでは、12 の ETI ファイル中 11 ファイルがパッケージに取り込まれた。残った 1 つのファイルは使用中のファイルのコピーであり、3.2 節で示した通り、使用外として除外されたものである。次に、相互評価を行う checklist フォルダのパッケージ化を試み、35 ファイル中 31 ファイルが取り込まれた。そのうち 2 つのファイルは現在使用しておらず、どのファイルからも参照されなかったものだった。残り 2 つのファイルは使用中であるにも関わらず、取り込まれなかったファイルであった。本 LMS は、学年を選び、そこからクラスを選び、そして生徒を選ぶというように、[シリーズ]－[セット]－[アイテム]の 3 層構造で操作する枠組みになっている。その 2 つのファイルは、いずれも機能の起点となるシリーズを操作するものであり、それは図 6 のように、シリーズからセットへのリンクが一方通行になっていることが、取り込まれなかった原因であった。シリーズを参照するファイルがないことで、使用外と判定された。この場合はマネージャーの判定ミスによるものではなく、LMS の実装の仕方に問題があったといえる。セットを操作する CGI に、シリーズへ戻るといったリンクを配置して双方向になれば、セットの CGI からシリーズを使用していると判定するので、シリーズのファイルが取り込まれる（図 7）。

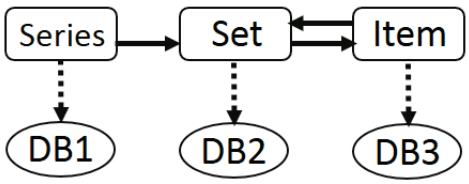

図 6　取り込まれなかったファイルのパス関係

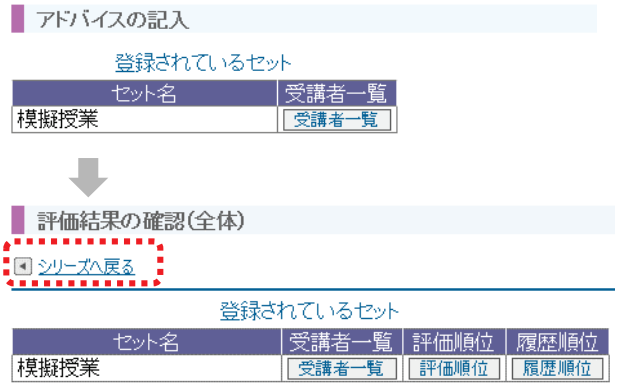

図 7　双方向にするためのモジュールの改善

このようにマネージャーは、使用中のファイルだけを取り込んで、モジュールの最新版を生成できただけでなく、リンクの不備も間接的に示して、システムの改善を促すことに寄与できた。

5 まとめ

本研究では、既存 LMS に対して、学習機能単位で CGI、DB、格納データの 3 つを洗練してパッケージ化する LMS マネージャーを開発し、いくつかのフォルダへの適用実験から、生成されたパッケージの精度について述べた。

その結果、ファイル間のパス関係を元に、使用中 CGI と DB の一覧を取得し、さらにその一覧から、アクセス権限や外部キーなどの DB 関連情報を構築したり、格納データを連番に直して整理したりして、洗練化したパッケージを自動生成することができた。また、出来上がったパッケージの構成を確認して、CGI 間のリンクの不備を発見し、LMS の改良を促すことにも活用できると示唆された。

開発した LMS マネージャーは、現在は ETI という限定的な言語のなかで稼動するものになっていることが課題であるが、新たに PHP や Perl を解析するパーサさえ作れば、様々な既存 LMS のパッケージ化にも対応できると予想される。

本研究は、科学研究費補助金若手研究 B（課題番号 24700903）の助成を受けたものである。

参考文献

[1] moodle、https://moodle.org/

[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

[3] 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

[4] 石川貴彦、赤間清、「LMS の拡張を補助する動作検証機能」、大学 ICT 推進協議会 2012 年度年次大会講演論文集、pp.1-4、2012

[5] 石川貴彦、赤間清、「再構築・再運用を考慮した LMS」、大学 ICT 推進協議会 2011 年度年次大会講演論文集、pp.301-304、2011