C11-5

再構築・再運用を考慮したLMS

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）は、教師あるいは学習者の要望に応じて適宜機能を拡張できる枠組みが望まれている。しかしながら、頻繁な拡張がシステム操作の複雑さや、不要になったプログラムの蓄積をもたらし、運用を困難にさせる原因にもなる。そこで本研究では、筆者らが 2001 年から独自開発を続けてきたLMSを例として、拡張したシステム全体を複数のサブシステムに分割し、それらを用途に応じて再構築・再運用するための方法について検討した。

1　はじめに

LMSに代表される多くのWebベースシステムは、基本的に HTML、CGI、SQL の３つの言語系を駆使して構築されるが、システムが大規模になればなるほど、これらの言語系が広範に絡み合い、たとえ１箇所の修正であっても、システム全体の挙動に影響を及ぼすことも少なくない。したがって、LMS の多くはメンテナンスやサーバ移行が非常に煩雑であり、システム開発者以外の者が継承して運用を続けていくためには、システム全体を隅々まで網羅的に調べるしかない。

そこで、LMS を継承者が円滑に運用できるための方法として、システム全体を単独運用可能なサブシステムに分割し、本質的な（枝葉末節な表現がない）部品として安全に扱うことができる技術が必要であると我々は考える。これにより、継承者は全体を網羅しなくとも、部品の修正や構成だけを考えればよいので、結果的にシステム運用は容易になる。また、サブシステムが本質的な部品として提供されるので、システムの新規導入を希望する者に対しては、部品の選択・統合のみの操作で、より洗練されたシステムを構築できることが期待される。

本研究の目的は、LMSの再構築・再運用のためのフレームワークおよび方法を検討することである。具体的には、筆者らが 2001 年から独自開発を続けていた LMS[1][2]を用いて、システム全体を複数のサブシステムに分割するための方法と、サブシステムを用途に応じて再構築・再運用できるための方法について議論する。

2　本 LMS の枠組み

2.1 一般的な LMS との違い

例えばMoodleの場合、学習機能の部品となるモジュールを拡充してシステムを構成していく。そのモジュールは、新たな学習機能が欲しいという要求に応じて開発が進められるように、「機能追加」という枠組みが主体となる。それに対し本LMSは、Moodleのような「機能追加」という枠組みを持ちつつ、プロトタイプとしてのシステム全体を拡張・修正していくなかで、ある段階で部品化できるところを切り出して、別途サブシステム化していくという「機能分割」の枠組みも持つ。そして、機能分割によって作られたサブシステムと、他のサブシステムを追加してシステム全体を再構築すると、そのシステムもまたプロトタイプとなって、新たな機能分割の可能性を与える。このようにして、本 LMS は機能分割と機能追加のサイクルを経て、システムそのものとサブシステムを洗練させていく（図１）。

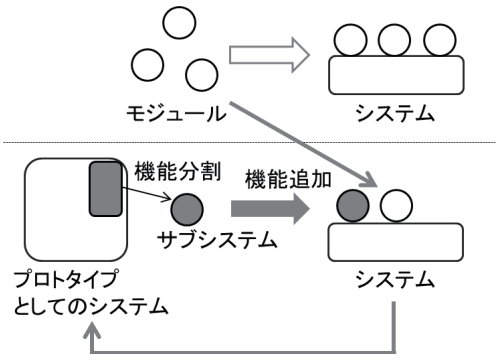

図１　一般的な LMS（上段）と本 LMS（下段）の枠組みの違い

2.2 メニュー

部品の概念を明確にするため、本 LMS では、「メニュー」という３層構造によってシステム環境を構成する。メニューとは、サブシステムの機能群を１つにまとめたもの、例えば学習者に提供する１つの学習環境を表す。このメニューは、サブシステム単位あるいはユーザ定義でグループ化でき、そのグループを「メニューカテゴリ」と呼ぶ。そして、メニューカテゴリに登録する１機能（１つの CGI ファイル）、例えば、動画配信サブシステム中の「動画の視聴」などを単位としたものを「メニューアイテム」と呼ぶ。したがって、メニューアイテムの集合はメニューカテゴリとなり、メニューカテゴリの集合はメニューとなる。なおユーザに表示する際は、メニューアイテムやカテゴリの並び順は変更可能である（図３）。

2.3 CGI の記述

本 LMS はルール型言語 ET[3]で CGI が記述されていることも特徴として挙げられる。例えばコメント記入という設定画面を追加する場合、図２のように、元の CGI ファイルに ET のアトム列（S 式）を追記し、参照するデータベースのカラムを増やすだけで概ねよい。このように本 LMS は、HTML、CGI、データベースアクセスをすべて S 式表現で統一し、様々なプログラミング言語を使い分けるという負担を減らして、プロトタイプとしてのシステム全体の拡張を、容易かつ迅速に行えるよう配慮している。

3 システムの再構築

3.1 CGI の分割

2.2 節で述べたメニューの構成が、システムの分割を見定める情報源となる。すなわち、メニューカテゴリがサブシステムを構成する CGI のフォルダに対応し、メニューアイテムがそのフォルダに格納される CGI ファイルとなる。例えば、図３はメニューの構成を表し、左図は機能単位でメニューを構成した状態である。これは、相互評価フォルダの中に６つの CGI ファイルが格納され、それとは別に、動画配信フォルダに２つの CGI ファイルが格納されていることを意味している。つまり、相互評価サブシステムと、動画配信サブシステムが個々の部品になっていることを示しており、メニュー構成をキーにして、システム分割のための情報を与えている。

右図は、相互評価の機能と動画配信の機能のうち学習者に提供する機能だけを抜粋し、１つに統合したメニュー構成である。これは機能単位でのシステム分割以外に、学習単位での分割が可能であることを示唆している。このメニュー構成は、サブシステム同士の機能の統合から、新たなサブシステムを生み出す可能性が期待できる。ただし、この分割はサブシステムの機能を包括できない（例えば、動画視聴機能は含まれるが、動画管理機能は含まれないことから、視聴専用のサブシステムになる）ケースが多いため、運用上ではメニュー構成の方法に課題が残る。

3.2 データベース

データベースはテーブルのほか、外部キー、SQL 実行権限（ロール）、ビュー、ストアドプロシージャなどの情報が LMS の運用にとって不可欠である。本研究では、これらの情報を ET のアトム列で記述し、ET の処理系で実行すると、データベースを再構築できる仕組みになっている。アトム列は、稼働しているプロトタイプとしてのシステムのデータベースにアクセスし、データベース情報生成プログラムの実行によって、全テーブ

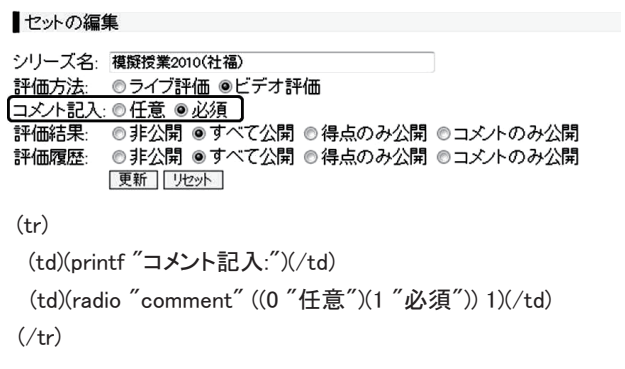

図２　相互評価の設定画面とアトム列との対照

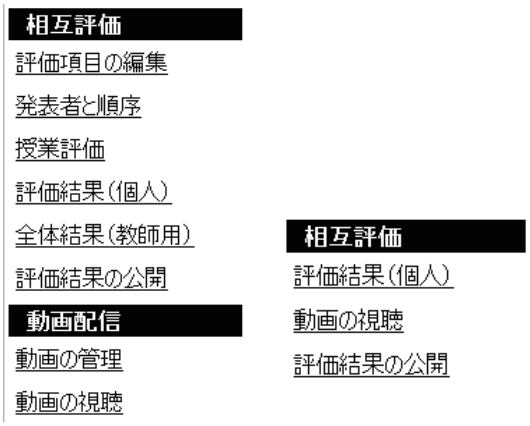

図３　メニューの構成（左：機能別　右：学習別）

ルに対応するアトム列を自動生成する。アトム列は、以下のように記述され、上から順に Color という名前のテーブルに対するデザイン(table)、アクセス権限(power)、Articlesテーブルとの外部キー関係(foreign-key)を表している。

```
?- (table "Color"
     (("colorID" "tinyint identity" 3 0 ( ) PK)
      ("name" "nvarchar" 10 0 ( ) ( )))).
?- (power "Color"
     (("public_user" ("Select")))).
?- (foreign-key "Color"
     (("colorID" ("Articles" "colorID")))).
```

3.3 格納データ

データベースの再構築は 3.2 節で示したデータベースの構造だけでなく、格納データの移行も重要であり、その移行方法についても考えなければならない。格納データは 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ケースが多い。そして、レコードを削除する際、連番で付与した ID は、削除されるとその ID が欠番になって、データベース構造の劣化が進む。劣化したテーブルは、管理の煩わしさやパフォーマンスの低下をもたらし、これをサーバ移行や他者に提供するなどの場合には、欠番のない状態で保持しておくことが望ましい。この欠番のない状態を本研究では標準形と定義する。

そこで、データベースの標準形化の方法として、図４の例をもとに考える。

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４　演習問題のデータベース例

これらのテーブルは、「国語」というセットの中に「漢字問題」と「文法問題」が含まれていることと、「算数」というセットの中に「計算問題」が含まれていることを表現している。そして、親テーブルの SetID2 と QuizID3 が、レコード削除によって欠番になっている状況を示している。

このデータベースを再構築する場合、①バックアップファイルをリストアする、②格納データを新たなテーブルに挿入していくといった方法が挙げられる。①の方法では欠番がそのまま残る状態で復元され、構造劣化は解消されない。②の方法ではオートインクリメントによって親テーブルの欠番はなくなるが、子テーブルは更新されないためリンクが一致しない。つまり、図４の子テーブルのように、SetID3 や QuizID4 が含まれていたレコードは、親テーブルの ID の変更によって挿入に失敗し、図５のような状態になる。したがって、「国語」のセットには「漢字問題」のみが含まれ、「算数」のセットは何も含まれないという状況が生じる。

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

SetID	SetName
1	国語
2	算数

SetID	QuizID
1	1

QuizID	Question
1	漢字問題
2	計算問題
3	文法問題

図５　テーブル間のリンクの不一致

この問題を解決するために、親テーブルの ID 書き換えに連動して、子テーブルの ID も同時に書き換えていく方法を考える必要がある。子テーブル ID の書き換え方法として、各親テーブルの新旧 ID 対照表を作成し、その対照表を基に子テーブル ID を書き換えていく。対照表の表現には配列を用いる。その理由としては、配列のインデックス番号と旧 ID を対応付けて、その配列に格納するデータを新 ID とすると、表１のような対照表を作成できるためである。例えば、図４の QuizSet テーブルでは、SetID2 が欠番となっているため、対照表の配列２は( ) (空データ) となり、新 ID では、その( )を除き連番で ID を割り振っていく（表１）。

表１　新旧 ID 対照表

QuizSet（親）

配列	0	1	2	3
旧ID	( )	1	( )	3
新ID	( )	1	( )	2

Quiz（親）

配列	0	1	2	3	4
旧ID	( )	1	( )	3	4
新ID	( )	1	( )	2	3

これらの対照表を ET のプログラムとなるルールでそれぞれ表現すると、

```
(array "QuizSet" "SetID"*X)
        → (= *X { ( ) 1 ( ) 2 }).
(array "Quiz" "QuizID" *X)
        → (= *X { ( ) 1 ( ) 2 3 }).
```

のように記述される。なお、*X のように*で始まるシンボルは変数を、{ }で表されたデータ構造は配列を表している。最初のルールは、「QuizSet テ

ーブルの SetID の対照表は、X[0]=( )、X[1]=1、X[2]=( )、X[3]=2 である」という意味を持つ。次に、これらの対照表ルールを用いて、新旧 ID 変換のルールを以下のように記述する。

```
(trans "QuizSet_Quiz" ((*OS *OQ) | *R) *NEWID)
        → (array "QuizSet" "SetID" *QS),
          (array "Quiz" "QuizID" *Q),
          (getArrayElement *QS *OS *NS),
          (getArrayElement *Q *OQ *NQ),
          (= *NEWID ((*NS *NQ) | *NR)),
          (trans "QuizSet_Quiz" *R *NR).
(trans "QuizSet_Quiz" ( ) *NEWID)
        → (= *NEWID ( ) ).
```

これは、QuizSet_Quiz テーブルの旧 ID である*OS(Old SetID)と*OQ(Old QuizID)を配列のインデックス番号として、対照表*QS(QuizSet)と*Q(Quiz)から*NS(New SetID)と*NQ(New QuizID)という新 ID を取得し、新旧 ID の書き換えをする。そして、この書き換えをレコードの数だけ繰り返すという処理を行う。これにより、子テーブルに対応する新 ID のリストが得られる。このリストを SQL 文のフォーマットに代入すれば、データベースアクセスできるアトム列がテキストファイルで生成され、その実行によって標準形のデータベースが再構築される（図６）。

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６　標準形のデータベース

4 システムの再運用

システムを再運用するとき、必要なサブシステムだけを選択する、過去のデータを新システムでも引き継ぐなどの要望が生じる。本 LMS を再運用する場合は、用いるサブシステム（CGI が入ったフォルダ）を選んで Web サーバに配置し、それに付随するデータベースを以下のようにアトム列で記述して、ET の処理系で実行し作成する。

```
?- (include "base.dat"), (include "base.etc").
?- (include "quiz.dat"), (include "quiz.etc").
?- (include "bbs.dat").
```

システム本体のデータベースである base.dat と base.etc は必須である。dat ファイルは、3.2 節で示したデータベースを再構築するプログラムであり、etc ファイルは、3.3 節で示した標準形となった格納データを、構築したデータベースに挿入するプログラムである。前述のアトム列の例を見ると、quiz（演習問題サブシステム）では過去問題のデータを用いるために dat と etc が記述されているが、bbs（掲示板サブシステム）では、過去のデータを用いず新規で運用するため、dat のみが記述されている状況であることがわかる。

5 まとめ

本研究では、LMS の再構築・再運用のフレームワークおよび方法を検討するため、独自開発した LMS を用いて、システム全体を複数のサブシステムに分割するための方法と、サブシステムを用途に応じて再構築・再運用できるための方法について議論した。

その結果、メニュー構成に基づく CGI の分割、データベース情報の自動生成、格納データの標準形化の３点から、再構築のための具体的方法を提案し、CGI の配置とアトム列の記述によってシステムを再運用するための方法を示した。

ただし、再構築を手動で操作する箇所があったり、ET のアトム列をファイルに記述してそれを実行したりと、システム継承者が容易に設定できるようになるまでには改善が必要である。今後は、再構築・再運用を簡便にする Web インターフェースの開発や自動化などを進めていく。

本研究は、平成 23 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] 三高康嗣、赤間清、石川貴彦、「Web ベースの自由学習支援システムの構築方法」、電子情報通信学会技術研究報告 Vol.103、pp.31-36、2004

[2] 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

[3] 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