Vol.2010-CLE-3 No.6
2010/12/9

Webサービスを利用したMoodleの課題チェック機能の外部拡張とその実践

伊藤　恵[†1]　美馬義亮[†1]　大西昭夫[†2]

本稿では，オープンソースの教育支援システムである Moodle に対し，学生提出物のチェック時に Web サービスを利用する機構を導入し，教師が独自に用意するチェック用システムと Moodle とを連携させて使用することで，Moodle 管理者と Moodle を利用する教師の双方にとって有用となる外部拡張について述べる．また，この拡張の利用事例として，実際の授業での Web サービス実装および実践結果を報告する．

External Development of Assignment Check Function for Moodle Using Web Service And Its Use

Kei Itou,[†1] Yoshiaki Mima[†1] and Akio Ohnishi[†2]

In this paper, we report an external development method of Moodle, an open-source course management system. The method is usable for both administrator and teachers and brings in an assignment checking function based on Web services. This function is implemented by the linkage between a feature using Web services in Moodle and a teacher prepared system which checks deliverables by students.

As a practice of this method, we also report the implementation of Web service and the use in the actual course.

1. はじめに

社会的なコンピュータ利用の拡大に伴い，教育現場でもコンピュータを用いた講義資料の作成や提示，学習者の演習活動などに使われるだけでなく，コンピュータを利用した様々な教育支援システム[1)–3)]が導入されるケースが増えてきている．

Moodle のようなオープンソースの教育支援システムでは，教師自身が管理者であって，かつ，相応のスキルがあれば，自ら拡張して，自分の授業に特化した機能を実装することも可能であり，Moodle の既存機能に対して独自の機能を追加できる．実際に教師自身の教育経験に基づいて作られた追加モジュールや機能拡張パッチなどが多数公開されている．しかし，授業ごとや課題ごとなどで異なった拡張が必要となる場合や，公開されている既存の拡張モジュール等と多数組み合わせたい場合などを考慮すると，Moodle 自体を都度拡張していくことはシステムとしての保守性低下の問題があるほか，拡張したい機能の処理内容や実装方法によっては Moodle システム自体の負荷増大や処理能力低下をもたらしてしまうこともある．また，教師自身が管理者でない場合には教師にとって担当授業科目に合わせた柔軟な拡張を Moodle に施すことは困難であり，特に教師が学習者に課した個々の課題ごとに Moodle の既存機能の範囲を超えた別々の採点機構を導入したい場合には，教師がコース上に作成した個々の小テストページごとに複雑な採点機能拡張をするのは難しい．

一方，教育支援に関わらずさまざまな Web システムの構築に際して，単一のシステムに多数の機能を盛り込むのではなく，個別の機能を持つ Web サービスと呼ばれるソフトウェア部品を組み合わせて Web システムを構築する方法が注目されており，Google/Amazon/Yahoo!Japan なども無料で利用できる Web サービスを公開している．Web サービスを利用して構成されたシステムは，別々に開発されたシステム部品がシームレスに統合され，複数のシステム部品が組み合わさっていることによる利用者への追加の負担なく利用できる利点をもたらすことが可能である．

そこで本稿では，学生提出物のチェック時にシステム内部で Web サービスを利用する機構を Moodle に導入し，教師が別に用意する提出物チェック用システムと Moodle とを連携させて使用することで，Moodle 管理者と利用する教師の双方にとって有用な外部拡張方法とその実践について報告する．

2. 関連研究

2.1 Moodle

日本の大学でも使われ始めているオープンソースの教育支援システムの一つに Moodle[1),2)] がある．Moodle には「コース」単位での教育コンテンツ管理，教師や学生などのユーザ管理など授業支援に関わる包括的な機能が揃っており，導入する教育機関や授業の状況に合わ

†1 公立はこだて未来大学
Future University Hakodate
†2 株式会社 VERSION2
VERSION2, Inc.

せてカスタマイズによる機能調整ができるほか，オープンソースソフトウェアであるため，使用者に相応のスキルがあれば細部にわたる機能変更/拡張も可能である．実際に教師自身の教育経験に基づいて作られた追加モジュールや機能拡張パッチなどが多数公開されている．

2.2 Moodle と他システムとの連携事例

Moodle 内部での機能拡張やカスタマイズの他，他システムと連携することで Moodle を機能拡張することも試みられている．

Moodle を Google Apps や Facebook，Google Search，Twitter 等と連携させるようなモジュールが公開されている[4)–6)]．しかし，これらは基本的には Moodle を他システムへのユーザインタフェースとして用いるものが多く，Moodle 上のコースでの通常の「活動」や「ブロック」との本質的な連携になっているものは少ない．

2.3 提出課題のチェック方法

教育現場において学習者がファイルで提出した課題レポート等をシステム的にチェックする方法は必ずしも一般的になっていない．

多くの教育支援システムにおけるファイルによる課題提出受付機能では，提出〆切の設定や提出の有無の確認はシステム的に行われて，教師による確認を支援できていても，ファイル形式のチェックやファイルの中身のチェックはシステムで為されないものが多く，これらに関しては教師側の人的コストが掛かることとなる．

特定の教育内容/教育環境に特化した専用システムでは，学習者の提出ファイルを詳細にチェックし，自動採点するものもあるが，様々な教育内容/教育環境への応用性は高くない．

3. 設計と実装

本稿では，Web サービスを利用することで Moodle と他システムの連携機構を実装し，Moodle への提出課題チェックを他システム上に実装することで，Moodle の採点機構の外部拡張を行った．本節では，全体の設計方針と拡張後の利用の流れ，連携部分の API 設計，および，Moodle 側の実装内容について紹介する．Web サービス側の実装については利用事例ごとに異なるため，その具体例を 4 章で紹介する．

3.1 設 計 方 針

課題チェック処理自体を Moodle とは独立したサーバ (以下チェックサーバ) 上で動作させ，チェック処理は Moodle 以外からでも利用できるよう Web サービスとして提供する．Moodle 上では，学生から課題が提出される度にチェックサーバ上の所定の Web サービスを利用して，提出物のチェックを行い評点やフィードバックを得て，それを Moodle 上に保存

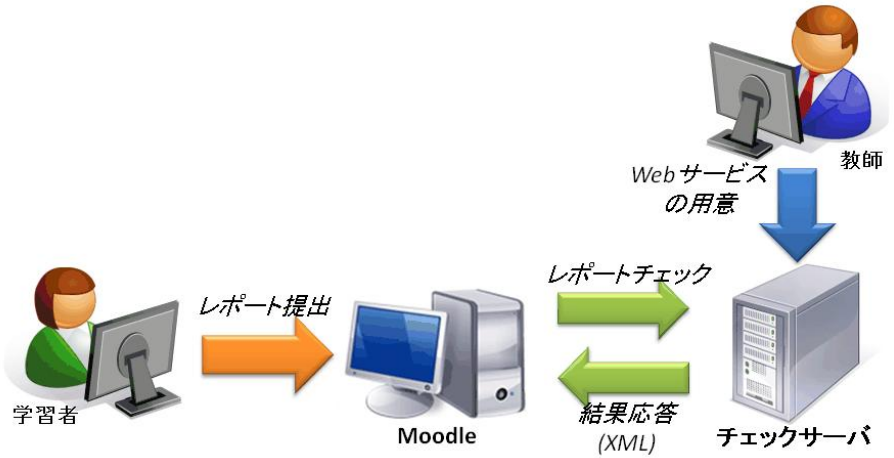

図 1 システム構成

および表示させる．この機構を使わない従来の Moodle の利用については従来通り Moodle システム単独で動作し，課題チェックが必要となった場合のみ，ネットワーク経由でチェックサーバと連携する (図 1)．

このような構成にすることで，以下のような利点が得られる．

(1) 教師は Moodle 自体の管理権限を持たなくても，チェックサーバ上でさまざまな課題チェックを柔軟に実現でき，かつ，そのチェック結果を Moodle 上で利用できる．

(2) チェックサーバ上で高負荷な処理を行っても，Moodle 自体の動作に影響を与えない．特にプログラミング科目においては，学生の提出したプログラムをコンパイルし，実際に実行して，実行結果が正しいかどうかを確認したい場合がある．Moodle 内部でこれを行うよう拡張してしまうと，コンパイルや実行にさまざまな制約が掛かったり，無限ループを含むようなプログラムが提出された場合にサーバが高負荷になり，他の Moodle 利用者にも大きな影響が出てしまうが，Moodle 自体と分離してチェックサーバを設けることで，これらの問題が軽減される．また，プログラミング以外でも自然言語処理やインターネット検索を用いてレポートの不正コピーチェックを行うなどといった利用も考えられる．

3.2 利用の流れ

実際に使用するにあたって，教師は (1) 学生が提出するファイルをチェックするためのス

クリプトを Web サービスとして用意し，(2)Web API を通じてそのスクリプトを利用するように Moodle 上の「問題バンク」にプログラミング問題を作成し，(3) その問題を含む小テストを作成する (図 2).

チェック用スクリプトを Web サービスとして用意するためにはスクリプト自体の作成のほか，Web サーバの用意などが必要となるが，内容や利用状況に応じて他の教師と共有することも可能である．プログラミング問題の作成やその問題を含む小テストの作成の流れは，Moodle 上の既存機能を用いて小テストを作成する場合と基本的に同様であるが，プログラミング問題の作成内容については 3.4 節で述べるような独自の内容が含まれることとなる．

学生は Moodle 上の小テストページに指定された提出ファイルをアップロードするだけで，チェック結果を見ることができ，Moodle と Web サービスが裏で連携していることを知る必要はない．ただし，適切に連携できていない場合にはアップロードされたファイルが単に Moodle 内に提出物として保存されるだけで，期待される結果表示は得られないことになる．

3.3 Web API の設計

Moodle とレポートチェック用の Web サービスとの間で情報交換するための Web API を設計した．Moodle で小テストの中でファイルアップロードを行った際に保持している情報のうち，レポートチェックに使用する可能性のあるものと，レポートチェックの結果として Web サービスから Moodle が受け取る必要があると考えるものを挙げて，設計を行った．

Moodle から送るデータはアップロードされたファイルデータそのもののほか，提出者の Moodle 上のユーザ ID，提出日時などが考えられ，Moodle が受け取るべきものとしては採点結果の評点やフィードバックとして提出者に提示するコメントが考えられる．

Web API として設計するため，Moodle から送るデータは HTTP の POST メソッドを用いて渡し，Moodle が受け取るデータは XML 形式で受け取る．表 1,2 は Moodle からチェックサーバに送られるデータ例とチェックサーバから Moodle に返される応答 XML の例である．

3.4 Moodle への実装

Moodle の小テスト上で利用できる新しい種類の問題として「プログラミング問題」を実装した．

Moodle には学生がファイル提出できるモジュールとして課題モジュールがあるが，ファイル提出以外の項目を併用したい場合や，複数ファイルを所定の順番に提出させて，それぞれ別のチェックを行いたい場合，また Moodle XML 等からの一括問題登録を行いたい場合

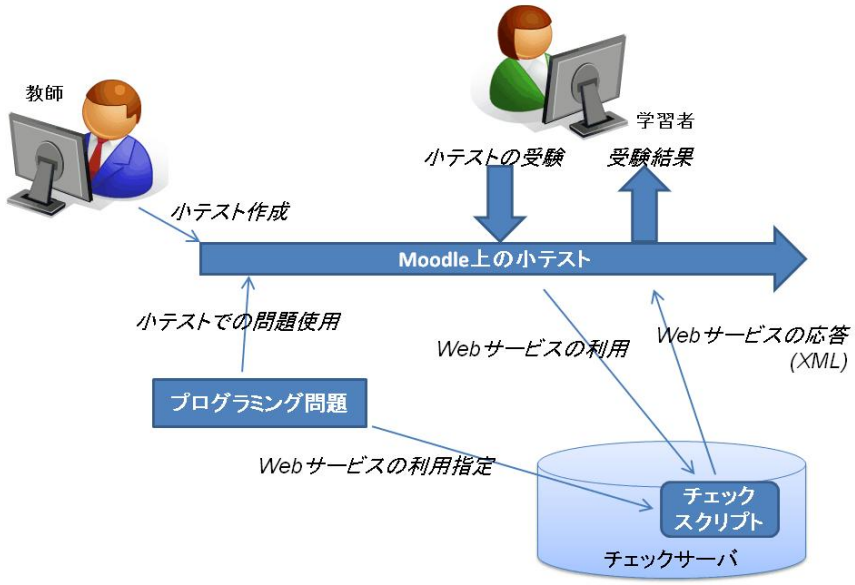

図 2 利用の流れ

表 1 Moodle からチェックサーバに送られる POST データ例

POST 変数	値
file	提出ファイル
id	Moodle ログイン ID
date	提出日時
email	提出者メールアドレス

表 2 チェックサーバから Moodle への応答例

```
<?xml version="1.0" encoding="utf-8"?>
<result>
<comment>チェック結果</comment>
<score>10.0</score>
</result>
```

表 3 プログラミング問題の設定項目

一般	カテゴリ
	問題名
	問題テキスト
	表示イメージ
	評点のデフォルト値
	ペナルティ要素
	全般に対するフィードバック
	タグ
アップロード設定	注意書き
	実行例
	ヒント
	締切日時
	締切後も提出可能
	締切後の最高得点
	チェック URL
	チェック POST 変数名
	その他の POST 変数名
評価設定	採点方法
	得点 XML タグ
	フィードバック XML タグ

などを想定し，課題モジュールの拡張ではなく，小テストモジュールに組み込める問題の種類の一つとして「プログラミング問題」を実装することとした．

「プログラミング問題」では，従来の種類の問題と同様の問題文の設定の他に，提出されたファイルのチェック用に利用できる外部の Web サービスの URL や，Web サービスから得られた応答の Moodle 内での利用方法等を問題ごとに個別に設定できることとした．プログラミング問題の設定項目は表 3 の通りである．カテゴリ，問題名等は既存の問題タイプと同様のものであり，プログラミング問題固有の部分はアップロード設定の部分と評価設定の部分である．以降，3.4.1 から 3.4.3 節でプログラミング問題固有の設定内容等について概説する．

3.4.1 問題提示

問題の提示は既存の小テスト問題と同様の問題テキスト表示以外に，著者らの所属大学におけるプログラミング演習科目の実際の出題形式を参考にし，注意書き，実行例，および，ヒントを表示できるようにした．

具体的には図 3 のような設定を行うことで，図 5 のような受験画面が表示される．この

表 4 採点設定の種別

Web サービスによる提出物チェック	なし	あり			
応答形式	-	plaintext		XML	
応答への採点結果含有	-	なし	あり	なし	あり

使用例では実行例やヒントを記述していないため，小テスト受験画面では注意書きのみが表示されている．

3.4.2 採点設定

採点に関しては，Web サービスによって自動採点結果を得る XML 採点，応答が XML ではなく単なるテキスト形式で点数のみ返される場合のプレーンテキスト採点，そして自動採点が困難な場合のための手動採点を選択できる (図 4)．つまり，提出されたファイルのチェックに Web サービスを利用するか否か，利用する場合の応答形式がプレーンテキストか XML か，そのどちらの場合も応答の中に点数自体が含まれるか否かによって，表 4 の 5 種類の設定方法が選択可能である．

3.4.3 締切設定

締切後も受け付ける場合の最高得点設定が可能であり (図 4)，例えば，締切前の提出であれば 100%で得点させ，締切後の提出であれば締切前の場合の 60%の得点になるように自動的な採点調整を行うことが可能である．

3.5 提出物のダウンロード機能

プログラミング問題を利用して学生が提出したファイルを Web サービス経由のチェックスクリプトによって自動処理するのとは別に，提出されたファイルを教師が別途確認したい場合があることを想定し，プログラミング問題で提出されたファイルに関する一括ダウンロード機能も実装した．

小テストの「受験結果」ページ群の中に「プログラミング問題」というページを新たに用意し，そこで利用したい条件を指定して，zip 形式による一括ダウンロードができるようにした．

同じ学生が同じ小テストを複数回受験することでファイルをいくつ提出した場合でも，すべての提出ファイルを Moodle 上に保存しておき，教師がそれらを一括ダウンロードできるようにした．すべての提出物をダウンロードしたい場合や締切前の最後のものだけダウンロードしたい場合など，小テストの運用方法によってさまざまな可能性があるため，一括ダウンロードの対象として (a) 締切前で最後，(b) 最後，(c) 最初，(d) その他の提出物を含め

るか否かを個別に設定できるようにした．また，一括ダウンロードした zip ファイルを展開した際に，その後のファイルの利用方法によって，学生ごとのフォルダに分かれて展開された方が望ましい場合と，受験ごとのフォルダに分かれて展開された方が望ましい場合があると考え，どちらにするかも一括ダウンロード時に指定できるようにした．

4. 利用事例

本節で紹介する事例ではプログラミング科目ではなく，特定形式のレポート提出を学期中に複数行う授業科目において，提出レポートの形式的なチェックに本稿の拡張機構を使用した．

この科目での利用では，レポートの記載内容の確認や評価・採点は教師が行うこととし，提出されたファイルが指定された形式になっているかどうかをチェックするスクリプトを Web サービスとして事前に用意した．これを Moodle のプログラミング問題から呼び出すように設定し，実際に授業学期中のレポート提出に使用した (図 5)．

4.1 Web サービスの実装例

今回の事例では提出されるレポートの形式自体も XML を用いており，その形式等を機械的にチェックするスクリプトを PHP 言語を用いて作成し，スクリプトで自動チェック可能な範囲のチェックを行って，チェック結果を Web サービスの応答として HTML 文字列で返すよう実装した．実装したスクリプトでは，具体的には以下のような項目をチェックした．

(1) 全く関係ないファイルを誤って提出していないか (XML 以外のファイル形式など)
(2) XML として文法上間違ったファイルを提出していないか (タグが正しく閉じられていないなど)
(3) 指定した DTD に従った XML 文書になっているかどうか

この事例では提出レポートの自動採点は行わず，スクリプトによるチェック結果を，課題提出に対するフィードバックとして返すのみとした．チェック結果は，見つかった問題点を箇条書きとして列挙して返す形式とした．返されたフィードバックは Moodle 上の小テストの個々の受験に対するフィードバックとして保存されるため，Moodle 上の小テストレビュー画面でいつでも閲覧することができる (図 6)．

4.2 授業での利用

公立はこだて未来大学 3 年前期選択科目「システム管理方法論」の 2010 年度に開講された 2 つのクラスの各 10 回のレポート提出に使用した．2 つのクラスは学生の所属コースに応じて分けられているものであり，開講時期や内容は同様である．2 クラスで Moodle 上の

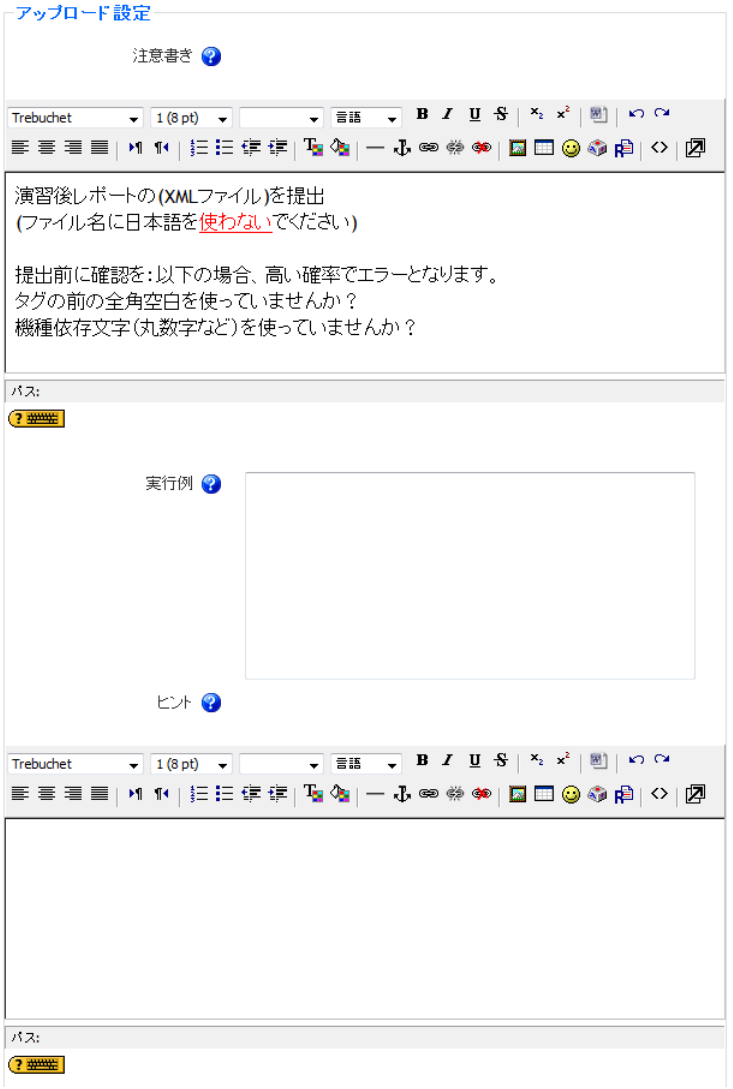

図 3 プログラミング問題の設定画面 (問題提示部分)

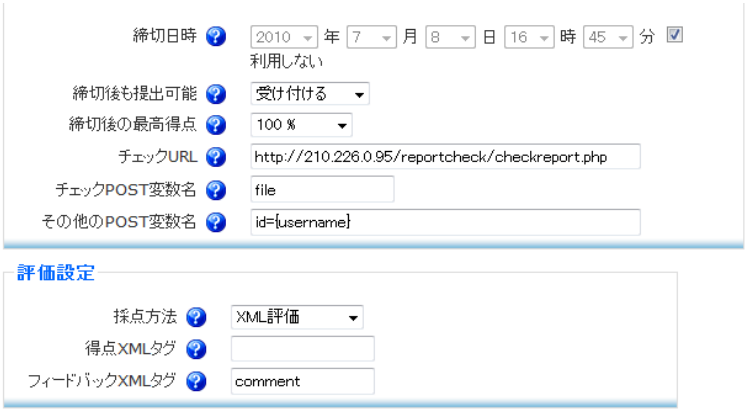

図 4 プログラミング問題の設定画面 (締切および評価設定部分)

1 コースを共用しており，コース上の小テストページ等もすべて共用である．Moodle コース内のグループ分け機能を用いて，受講学生の履修クラスを区別している．

学期中に実施した 10 回のレポート提出の 2 クラス合わせた提出者数，提出回数 (Moodle 上の提出用小テストの受験回数) は表 5 の通りである．レポートの内容チェックや採点は教師の手作業であるもののアップロードしただけで形式上のチェックは行われるため，1 回のレポート提出に対して個々の学生が 2~4 回程度の提出を試みており，特に最終成績への影響が大きいと事前アナウンスされた「まとめ 1」および「まとめ 2」のレポート提出に際しては提出回数が増え，学生 1 人当たり平均 3 回以上の提出が試みられている．なお，表中でレポート 2 の平均提出回数が 4.00 と多いのは，レポート 1 の提出時点では小テストページでのレポート自動チェックについて十分に周知されておらず，レポート 2 の提出時点では利用方法が把握できたために提出回数が増えたものと考えられる．

また，これらのレポート提出に関して，2 つのクラスを担当する教師 2 名は「まとめ 1」および「まとめ 2」で提出されたレポートを，一括ダウンロード機能を用いてダウンロードし，レポート内容についての詳細なチェックおよび採点を行った．採点結果は Moodle の既存機能により,「評定」への一括インポートを行った．提出されたレポートの形式的なチェックはチェックサーバで自動で行い，内容的なチェックは教師が手動で行うという運用形態に対しても，支障なく対応することができた．

表 5 授業時の利用状況

提出者数	提出回数	レポート番号	平均提出回/学生
64 名	150 回	1	2.34
61 名	244 回	2	4.00
60 名	139 回	3	2.32
61 名	229 回	まとめ 1	3.75
61 名	136 回	4	2.23
58 名	116 回	5	2.00
57 名	111 回	6	1.95
55 名	140 回	7	2.55
55 名	109 回	8	1.98
54 名	175 回	まとめ 2	3.24

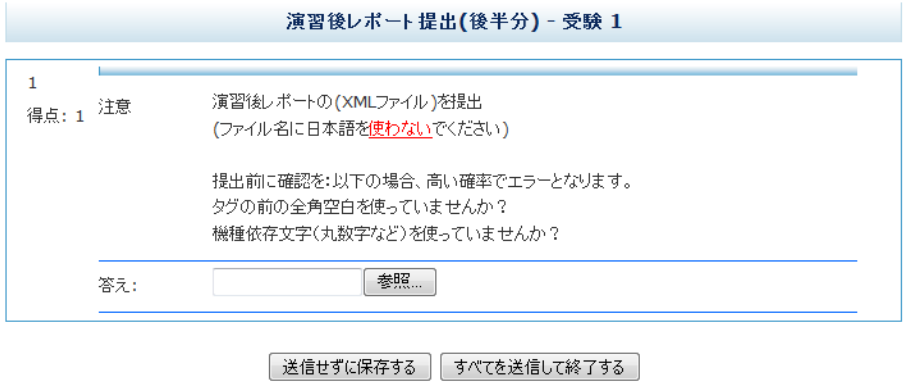

図 5 小テスト受験画面

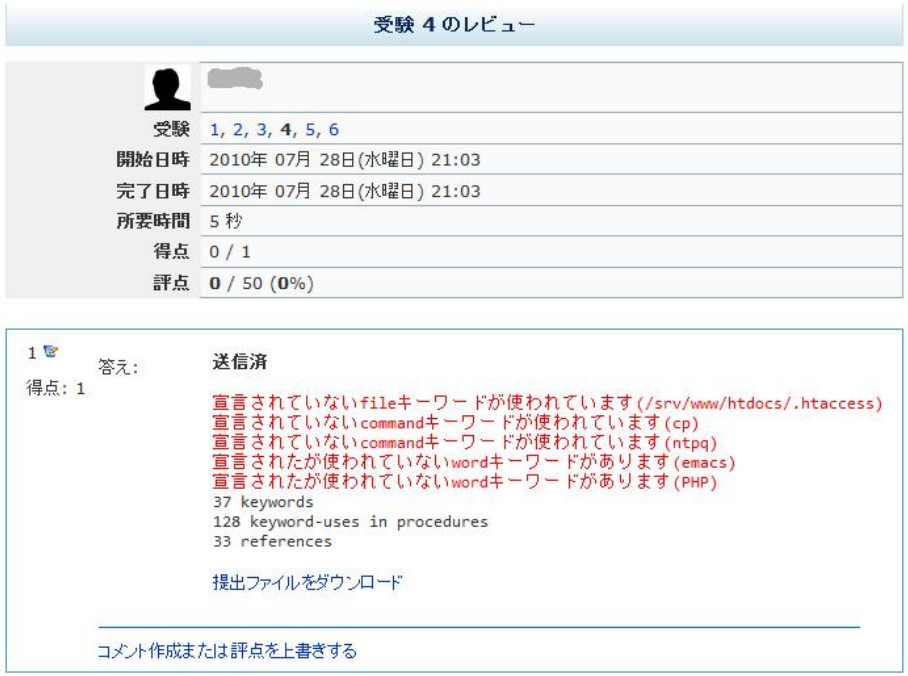

図 6 小テスト受験結果画面

4.3 利用結果

レポート提出時のレポート形式チェックにより，所定の形式を逸脱したレポート提出はほとんど為されなくなり，教師にとってレポートの内容に関する本質的な確認や採点に専念し易くなった．また，学生にとっても，形式が異なっていたために提出レポートが教師に採点して貰えないというリスクが減ったと考えられる．

ネットワークを介した課題チェック機構の利用が，ネットワーク帯域やサーバ負荷に影響を与える可能性があったが，提出レポートのファイルサイズが数 KB から 50KB 程度と小さいこと，チェック処理もファイルの形式的なチェック程度であって軽微な処理であったこと，また，提出する学生数がそれほど多くなく，かつ，2 クラスに分かれているため提出が短時間に集中することもあまりなかったことから，目立った影響は見られなかった．

5. 考　　察

有用な授業運用支援機能を持つ Moodle に対して，Moodle そのものを都度改造しなくても教師の自作スクリプトによるレポートチェックを組み合わせることができるようになったことで，教師が Moodle の管理者を兼ねていない場合でも，管理者に多大な業務負荷を掛けることなく，独自のチェックを組み込んだ課題提出ページを作成することが容易になった．また，Web サービスとしてさえ提供されていれば，チェックスクリプトの開発言語も限定されないし，チェックスクリプト自体は Moodle と別のサーバで稼働させてよいため，チェックスクリプトの稼働環境についても自由度が高い．

ただし，Moodle の中核的な機能の一つである「採点」を外部化することによるネットワーク帯域の問題やチェックサーバが高負荷になった場合に，この機構や Moodle 自体の利用に問題が起こることが予想される．現状では，通信タイムアウトによるチェック処理中断や，チェック結果の Moodle 内でのキャッシュなどを導入することで，サーバ間のネットワークやチェックサーバ側に問題が生じた場合でも，Moodle 自体の稼働にはできるだけ影響が出ないような対策を取っているが，実際にアクセスが集中した場合やチェックサーバの負荷が高くなった場合などについての評価はまだ行っていない．

また，一つの課題が Moodle 上の問題記述とチェックサーバ上のチェックスクリプトに分離されるため，長期的な授業科目運用を考えると本来一つであるべき教材を分離して管理することになり，一貫性を保ちながら管理し続けることによる教材管理負荷の増加が懸念される．

Vol.2010-CLE-3 No.6
2010/12/9

6. 今後の予定

本稿で紹介した利用事例はあくまでもレポートの形式的なチェックを行うものであり，プログラミング問題の本来の開発意図にあったプログラミング科目におけるレポートチェックとはやや異なるものである．今後，2010 年度後期や 2011 年度に著者らの所属大学内で比較的履修者数の多い C や Java のプログラミング科目において，課題提出への利用を予定している．これらの科目では学生の提出したプログラムをチェックサーバ上で実際に実行させてチェックすることを予定しているほか，受講者数も多いため，ネットワーク帯域やサーバに対する負荷も含めて，この機構に関する本格的な評価を行うことができると考えている．

7. ま と め

教師がレポートチェック用のスクリプトを自由に用意でき，学生が Moodle にレポートを提出するだけで，教師の用意したスクリプトによってチェックされ，その結果が他の評定と同様に Moodle 上で利用できるようにすることを目的として，Web サービスを利用した課題チェック機能を Moodle に実装した．

当初想定したプログラミング科目での実践は未実施だが，所定の形式でレポート提出をさせる授業科目において，提出されたレポートの形式的なチェックを自動実施し，限定的な条件下ではあるものの，実際の授業内で本機能を問題なく利用することができた．

今後は，プログラミング科目において，より本格的な実践を行っていく予定である．

参 考 文 献

1) Dougiamas, M.: Moodle, http://moodle.org.
2) 井上博樹，奥村晴彦，中田　平：Moodle 入門 – オープンソースで構築する e ラーニングシステム，海文堂出版 (2006).
3) 日本システム技術株式会社：GAKUEN/UNIVERSAL PASSPORT, http://www.jast-gakuen.com/.
4) Inc., M.: Integration: Moodle-Google Apps, http://moodle.org/mod/data/view.php?d=13&rid=2164&filter=1.
5) Fulton, A.: Integration: Facebook, http://moodle.org/mod/data/view.php?d=13&rid=3316&filter=1.
6) Ridden, J.: Block: Ajax Google Search Block, http://moodle.org/mod/data/view.php?d=13&rid=1337.