情報処理学会論文誌 Vol. 52 No. 12 3121–3134 (Dec. 2011)

コース管理システムと授業固有の課題チェック機能のWebサービスによる連携

伊藤 恵[†1] 美馬義亮[†1] 大西昭夫[†2]

本稿では，汎用的なコース管理システムを用いて授業の管理を行うことと，個々の授業ごとに特有の提出物自動チェック機構を導入することをWebサービスの仕組みを用いて同時に利用可能なようにする．そのことにより，コース管理システムの管理者の負担増大と個々の授業を担当する教師のシステム管理業務を少なく抑えながら，個々の授業にとって自由度の高い提出物チェック方法を利用可能とするシステム連携方法を提案する．具体例としてオープンソースのコース管理システムであるMoodleにこの連携方法を実装し，実際に教師が授業ごとに用意する提出物チェック用Webサービスと連携させて使用した利用事例を報告する．

A Linkage Mechanism between Course-management-system and Coursework-checking-functions over Web Services

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

A linkage method for enabling generic course-management-system to work with coursework-checking-functions, which provide course specific assignment checking functions, by using web service mechanism, is proposed in this paper. As the result of the application of above linkage method, course-management-system is extended to serve more flexible checking functions than original system, while it keeps the amount of maintenance additional workloads for both the system management and instructors' operations for extension as small as possible. As a practical application of this linkage method, an extension of the open-source course management system: Moodle is made. The detail of the implementation and the results from real educational practices over web service, are also described.

1. はじめに

教育現場において複数の授業の管理に利用可能なさまざまな汎用的コース管理システムが提供されている．これらのシステムでは，多くの授業で共通して必要となるような汎用的な機能は充実しているが，個々の授業に特有の部分は別途用意する必要に迫られることが多い．Moodle[1),2)]のようなオープンソースのコース管理システムでは，システム自体の機能を独自に拡張することが可能であるが，授業独自の機能を用意した教師がコース管理システム自体の管理を負うことになったり，それを避けるために外部委託したことによって授業特有の部分に関する自由度や対応の柔軟性が損なわれたりすることもある．

汎用的なコース管理システムと個別の授業のための支援システムを連携させる試みは多くある[3)]．疎な連携方法の場合には双方のシステム管理は独立しているため，教師が担当する授業のための個別授業支援システムを管理することはあっても，連携先のコース管理システムの管理を負わなくて済み，また，個別授業支援システムの機能拡張等に際しても柔軟な対応が可能である．しかし，連携が疎であるがゆえに，たとえばユーザ認証が別々になったり，ユーザインタフェースに一貫性がなかったりすることにより，利用者からは「連携しているが別のシステム」として認識され，使用性が下がる傾向がある．逆に密な連携方法の場合は，利用者に「1つのシステム」と認識されて使用性が下がらない可能性が高いが，システム管理の負担や拡張時の対応の柔軟性に影響が出ることがある．

CAS統合認証したシステムどうしをuPortalのようなポータル環境によって統合する方法もある[4)]．この場合はシステムどうしの内部的な連携は疎な特徴を保ったまま，ユーザインタフェースはシームレスに連携されるため，使用性が下がりにくい．しかし，ユーザインタフェース上でシームレスになっているだけであるため，学習履歴等をシステム間で連携させるにはどちらか一方あるいは両方のシステムに相応の対策を施す必要がある．

汎用的なコース管理システムにはなく，個別の授業支援システムにある，独自の機能がどういった性質を持つものであるかによって，どう連携すべきか，あるいは，連携せずに統合すべきか等適した対策が異なってくると考えられる．本稿では提出課題のチェック方法のみを独自に用いたい場合に着目する．つまり，授業支援のほとんどすべての機能はコース管理

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

システムに任せ，提出課題のチェック方法だけを授業独自の機能で実現する場合の連携方法について考える．

そこで本稿では，授業自体の管理は汎用的なコース管理システムを利用しながら，Web サービスを使って個々の授業ごとに特有の提出物自動チェック機構を容易に組み込めるようにする．コース管理システムの管理者の負担増大と個々の授業を担当する教師のシステム管理業務量を少なく抑えながら，個々の授業にとって自由度の高い提出物チェック方法を利用可能とすることを目指し，この条件を満たす連携方式の設計およびコース管理システム Moodle への実装を行った．いくつかの授業科目での実利用を行ったのでここに報告する．

2. 関 連 研 究

2.1 コース管理システム

教育現場において複数の授業の管理に利用可能なさまざまな汎用的コース管理システムが提供されている．コース管理システムにはユーザアカウント管理，教師によるコースの作成や管理，学生のコースへの登録，小テスト/課題提出/掲示板等のさまざまなモジュールをコースごとに選択して利用する機能等が提供されていることが多い．汎用的なコース管理システムでは，多くの授業で共通して必要となるような機能は充実しているが，個々の授業に特有の部分は別途用意する必要に迫られることが多い．

2.1.1 Moodle

オープンソースのコース管理システムの 1 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，教師や学生等のユーザ管理等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せてカスタマイズによる機能調整ができる．また，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等が多数公開されている．

しかし，授業独自の機能を用意した教師がコース管理システム自体のシステム管理を負うことになったり，それを避けるために外部委託したことによって授業特有の部分に関する自由度や対応の柔軟性が損なわれたりすることもある．

2.2 汎用的コース管理システムと個別授業支援システムの連携方法

汎用的なコース管理システムと個別の授業のための支援システムを連携させる試みは多くある[3)]．連携方法によってシステムの保守コストや負荷分散，操作等の統一性，機能の連携度合等が異なってくる．個別の支援システムの機能をコース管理システムに組み込む集中型の場合，支援システムの側の機能拡張の際にもコース管理システム全体を考慮する必要があり保守コストが高くなりやすいうえ，コース管理システムに機能が集中するため負荷分散させにくくなるが，操作等の統一性は高く，支援システムの機能とコース管理システムの機能を連携させやすい．個々の支援システムをコース管理システムとは別に稼働させ，システム間での連携を図る分散型の場合，支援システム単独での機能拡張が容易であってコース管理システムと分けて保守を行うことができると考えられるほか，システムが別に稼働することで負荷分散しやすくなると考えられるが，操作等の統一性は保ちにくく，システム間の機能連携には相応の対策を施す必要がある．

1 章で触れたように，CAS 統合認証したシステムどうしをポータル環境によって統合する方法もある[4)]．この場合，システム間の連携は分散型のままでありながら，CAS サーバを介したシングルサインオンによる認証を行い，ポータル環境下で複数の対応システムがシームレスに連携されるため，使用性が下がりにくい．しかし，ユーザインタフェース上でシームレスになっているだけであるため，学習履歴等をシステム間で連携させるためにはどちらか一方あるいは両方のシステムに相応の対策を施す必要がある．

2.3 提出課題のチェック方法

教育現場において学生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．

多くの汎用的なコース管理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，提出締切の設定や提出の有無の確認は可能であるものの，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な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かかることとなる[5)]．

特定の教育内容/教育環境に特化した専用システムでは，学生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[6),7)]，さまざまな教育内容/教育環境への応用性は高くない．

3. アプローチと設計

本稿では，授業自体の管理は汎用的なコース管理システムを利用しながら，Web サービスの枠組みを利用し，個々の授業ごとに特有の提出物自動チェック機構を容易に組み込めるようにすることで，コース管理システムの管理者の負担増大と個々の授業を担当する教師のシステム管理業務を少なく抑えながら，個々の授業にとって自由度の高い提出物チェック方法を利用可能とすることを目指す．

3.1　アーキテクチャ

本稿では，汎用的なコース管理システムと授業特有の機能の連携方法として，授業特有の機能を Web サービスとして用意し，コース管理システムからその Web サービスを利用する方式をとることとした．

この方式では，システムを利用する学生からはコース管理システムだけが見え，その裏で利用される Web サービスは直接見えないため，ユーザ認証が別になったり，ユーザインタフェースが異なったりするような使用性の低下が防げる．また，教師の側では個々の提出物チェックを 1 つ 1 つ単機能のサービスとして提供すればよく，教師が責任を負う必要のあるシステム管理に相当する業務を少なく抑えることができる．コース管理システム自体とは管理が別になるため，コース管理システムのシステム管理者の手を煩わせることなく，Web サービス側の提供内容を柔軟に変更することが可能である．

Web サービスは，利用する側から利用される側へ Web アクセスができさえすれば基本的に通信可能であり，コース管理システムを稼働させているサーバとは別のサーバで Web サービスを提供できるため，汎用コース管理部分と授業特有部分との負荷分散が可能である．コース管理システム側で Web サービスからの応答待ちに適度なタイムアウト時間を設定するなどして，所定時間内に応答がなくてもコース管理システムとしての動作に支障がないように実装すれば，Web サービス側で問題が生じてもコース管理自体にはほとんど影響を与えずに運用を続けることができる．たとえば，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．コース管理システムと同一サーバでこれを行おうとすると，コンパイルや実行にさまざまな制約がかかったり，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，コース管理システムの他の利用者にも大きな影響が出てしまうが，Web サービスにより連携している別サーバでこういったチェックを行うことにすれば，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 Web サービス内で自然言語処理やインターネット検索を利用することで，レポートの不正コピーチェックを行う等といった利用も考えられる．また，提供するサービスや利用する側のシステムとの連携方法にもよるが，いわゆるユーザインタフェースやユーザ認証等の仕組みを持つ必要がない．

教師が誰でも Web サービスを用意できるというわけではないが，必要な API さえ満たせば実装言語等や動作環境は問わないという特長もある．Web サービスを用意する際には Web サーバ等の準備も必要になるが，内容や利用状況に応じて他の教師と共用することも可能である．Web サービスは，もしも必要があれば，連携することを想定したコース管理システム以外からも利用することが可能であり，複数のコース管理システムからの並行利用も可能である．

3.2　利用方法の設計

一般に教育の場での提出物チェックにおいては，自動的にチェック可能な部分と教師によるチェックが必要な部分がある．本稿で提案する方法で直接扱えるのは前者だけになるため，後者をどのように扱うかを考慮しておく必要があるが，コース管理システムの持つ機能の範囲内で，以下のような対応が可能であって，かつ，どの対応を採用するかを教師が選択可能であることが望ましい．

(1)　自動チェックの結果を教師が上書きできるようにする．
(2)　自動チェックの結果とあわせて教師によるチェックの結果を提示できるようにする．
(3)　自動チェックの結果は学生に提示せず教師が参照するだけとして教師によるチェックを合わせた結果のみを学生に提示する．

授業内での実際の利用例としては，たとえば，学期の途中で理解度確認等のために実施し，最終成績への影響が少ない小テスト等では，自動チェックの結果をそのまま学生に通知するようにしておき，期末試験等では自動チェック結果は学生に直接は提示せずに教師のみが参照したうえで，教師による採点結果を学生に提示する等の利用方法が考えられる．

3.3　連携 API の設計

コース管理システムの提出物受付機能の中に Web サービスを呼び出す処理を実装する必要があるが，Web サービスそのものは個々の授業独自の提出物チェック機構を用意する段階になるまで設計できない．この時点で設計する必要があるのは情報の受け渡し方，コース管理システムから提供可能あるいは提供すべき情報の範囲，コース管理システムが Web サービスから受け取る応答である．

3.3.1　情報の受け渡し方

Web サービスを用いる前提では情報の受け渡し方として SOAP や REST 等が考えられるが，それらの場合に必要となるタグ名やパラメータ名を最終的には決定する必要がある．ただし，コース管理システム上の教師用のインタフェースの範囲内で可能であれば，これらのタグ名やパラメータ名も教師が設定できる方が，個々の授業用に Web サービスとして提出物チェック機構を用意する際の自由度が高く，また，用意される Web サービスによってコース管理システムの実装変更が必要となる可能性が低くなる．以下の説明では REST を採用し，タグ名/パラメータ名をコース管理システム側の機能実装時に固定する必要があるとの想定で行う．

表 1　コース管理システムから Web サービスに送る REST データ例
Table 1　Example REST data sent from CMS to Web service.

POST 変数	値
file	提出ファイル
id	学籍番号等教師が学生を特定できる ID
date	提出日時

3.3.2　コース管理システムから提供する情報

提出物チェックという目的をふまえるとコース管理システムから提供可能な情報としては，提出されたファイルそのもの以外に，提出日時や提出元の情報，提出者自身の情報等がありうる．提出者自身の情報としては氏名や学籍番号等の単純なデータのほかに，コース管理システムが保持している範囲内での当該学生の学習履歴等もありうる．また提出物に付随する情報として提出物自体とは別にファイルサイズやファイル形式等の情報を提供するということも考えられる．

また，受付可能なファイルサイズの上限やファイル形式をコース管理システム側で限定しておき，合致するものだけを Web サービスに送るという設計が考えられるが，将来的にどの程度のファイルサイズ，どのようなファイル形式を受理すべきかは変わっていくと想像されるため，コース管理システム側ではこれらに関するチェックやフィルタリングを新たには行わず，すべて Web サービスに送り，Web サービス側でのチェックに任せることとした．もちろん，コース管理システム側の本来の設定で学生がアップロード可能なファイルの上限サイズは別途定められているのが通例であり，それを超えるものは本来送信されない．

提出物チェック機構側の利便性やセキュリティ面をふまえると，提供可能な情報を多く用意して，実際にどれを渡すかは利用する教師がつど選択できることが望ましいが，コース管理システム側の実装によっては教師による選択が困難な場合には，必要度の高い情報に絞り，提出ファイルそのもの，提出者に関する最小限の情報（学籍番号等），提出日時程度に限定するという選択肢もありうる（**表 1**）．

3.3.3　コース管理システムが受け取る応答

コース管理システムが Web サービスから受け取るべき応答は，コース管理システム自体が保持可能な情報の種類にもよるが，（自動採点可能な場合の）点数の数値と，提出物に対して自動的に生成可能な範囲のフィードバックコメントのテキストがあれば，さまざまな利用状況に対応できるものと考えられる（**図 1**）．

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

図 1　Web サービスからコース管理システムへの応答例
Fig. 1　Example responce from Web service to CMS.

4.　Moodle への実装

3 章の設計に基づき，オープンソースのコース管理システム Moodle の「小テスト」モジュールを拡張し，提出物チェックに Web サービスを利用可能とした拡張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 5 章で紹介する．

4.1　実 装 方 針

Moodle の「小テスト」モジュールに新たな種類の問題を作成し，アップロードされたファイルを Web サービスに送って，得られた結果を採点結果と学生へのフィードバックとして Moodle に保存することとした．原理的にはさまざまな使途が考えられるが，主としてプログラミング科目での利用を想定して，この問題の種類を「プログラミング問題」と命名した．

プログラミング問題を含む小テストを学生が「受験」し，当該問題にファイルを提出するたびに Web サービスへのデータ送信が行われ，Web サービスから応答として返される提出物の点数とフィードバックコメントを取得して，それにより小テストの受験結果が更新される．点数とフィードバックは Moodle の従来の小テストと同様に受験後すぐに学生が閲覧できるかどうかを教師が設定可能である．Web サービスからの応答が一定時間以内に得られなければ，アップロードされたファイルを保存するだけで採点結果やフィードバックの更新は行わない．また，小テスト以外のモジュールやプログラミング問題が使われていない場合の小テストは，従来の Moodle と同様に動作する．

つまり，Moodle はフロントエンドとして学生に問題を提示し，解答を受け取るという機能を果たし，Web サービス側は提出された解答を引き継ぎ，評価して Moodle に返すバックエンドサービスを行うことになる（**図 2**）．

4.2　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

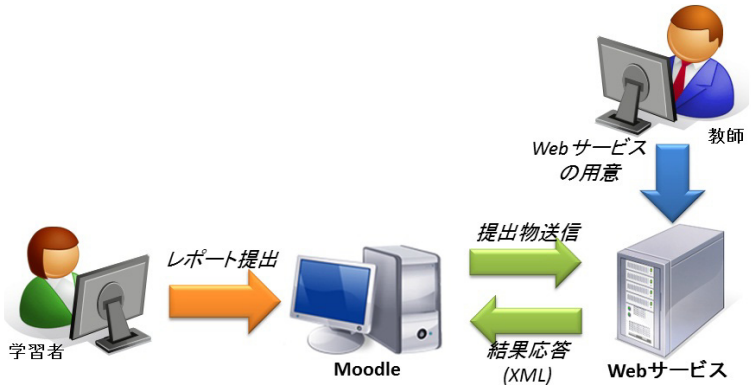

図 2　システム構成
Fig. 2　System architecture.

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 3**）．

チェック用スクリプトは学生の提出したファイルに授業で必要となるチェックを施し，結果として点数やフィードバックとしての評価コメントを返すように作成する．スクリプトの実装言語は基本的にどのようなプログラミング言語でもかまわないが，最終的に Web サービスとして提供するため，スクリプトを最初から開発するのであれば Web アプリケーション開発に向いている言語の方が望ましい．Web サービスとして実装する際には学生提出物を HTTP POST で受け取り，結果を XML で返すように実装する必要がある．

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（以下チェックサーバ）の用意等が必要となる．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 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 4.3 節で述べるような独自の内容が含まれることとなる．

学生が 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と正誤チェック等がバックエンドにあるチェックサーバによってなされる．（教師が結果をすぐに見られるよう小テストの設定をしていれば）学生は，自動的に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，Web

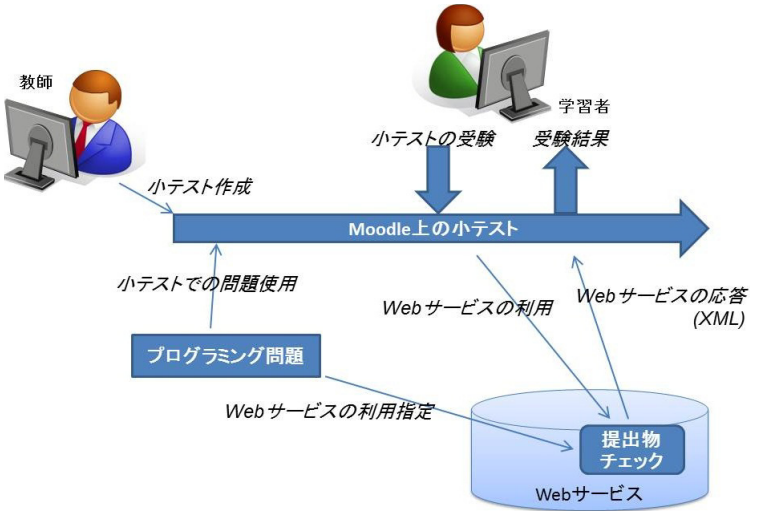

図 3　利用の流れ
Fig. 3　Service flow.

サービスが停止している場合や一定の時間内に応答が得られない等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．

4.3　プログラミング問題の実装

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等を想定し，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種類の問題の 1 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「記述問題」と同様の問題文等の設定のほかに，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可能とした．プログラミング問題ごとの設定項目は **表 2** のとおりである．一般部分の設定項目は既存の「記

表 2　プログラミング問題の設定項目
Table 2　Items for Programing Question configuration.

一般（既存の記述問題と同様）	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

表 3　プログラミング問題の Web サービス利用設定例
Table 3　Example Web service configuration for Programing Question.

設定項目	値
チェック URL	http://...../autocheck/check.php
チェック POST 変数名	file
その他の POST 変数名	id={username}&num=11
採点方法	XML 評価
得点 XML タグ	score
フィードバック XML タグ	comment

表 4　その他の POST 変数名で設定可能なもの
Table 4　Available tags for other POST variables.

{username}	ログイン ID
{kanjiname}	漢字氏名
{latinname}	ローマ字氏名
{email}	メールアドレス
{lang}	ユーザのデフォルト言語
{schoolname}	学校名
{affiliation}	所属学部・学科
{studentnumber}	学籍番号

述問題」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．ただし，アップロード設定の設定項目のうち，「注意書き」から「締切後の最高得点」までは本稿で述べている Web サービスとの連携には直接関係なく，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状況を調査した結果，多くのプログラミング問題で必要であると判断して追加した項目である．

Web サービスとの連携に直接関係する設定項目について，次節で概説する．

4.4　Web サービス利用設定

表 2 のアップロード設定のうち，直接 Web サービスの利用に関係するものは，チェック URL，チェック POST 変数名，その他の POST 変数名の 3 つである．これらは**表 3** の設定例のように教師が設定可能である．チェック URL，チェック POST 変数名，その他の POST 変数名はそれぞれ利用する Web サービスの URL，提出ファイルを送信する際の POST パラメータ名，および，その他 Moodle から Web サービスに送る POST パラメータ名と値の組である．その他の POST 変数名の部分には**表 4** で示されているデータの範囲内で，Moodle システム自身が保持しているデータを Web サービスに送信することが可能である．

表 2 の設定項目のうち評価設定の部分が，Web サービスからの応答を Moodle の小テストモジュールがどう処理するかを設定する部分であり，これらは表 3 の設定例のように教師が設定可能である．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**）．また，採点自体は行わずに XML またはプレーンテキストの応答の中にフィードバックコメントのみを含めることも可能であり，これらを合わせて**表 5** の 5 種類の設定方法が選択可能である．なお，応答形式がプレーンテキストの場合で採点結果が含まれる場合，1 行目に点数が書かれており，2 行目以降にフィードバックが書かれているものとして扱うこととした．

このように Web サービス利用時のパラメータを教師が自由に設定できるようにしたことで，Web サービス側の実装や通信時のセキュリティ方針に応じた柔軟な対応を可能とした．

4.5　手動採点への対応

Moodle の小テストモジュールは，学生の個々の「受験」に対して点数とフィードバックコメントを保持することができ，プログラミング問題での Web サービス利用では，Web

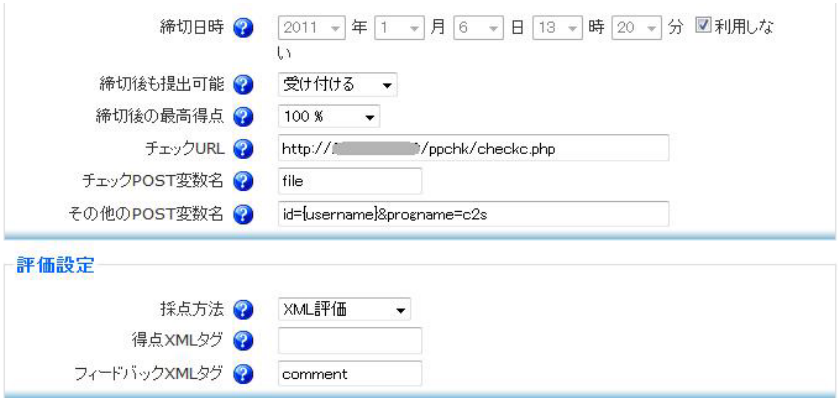

図 4　プログラミング問題の設定画面（締切および評価設定部分）
Fig. 4　Screenshot of configuration for Programing Question (Deadline and scoring).

表 5　採点設定の種別
Table 5　Kinds of score configuration.

Web サービスによる提出物チェック	なし	あり			
応答形式	-	プレーンテキスト		XML	
応答への採点結果含有	-	なし	あり	なし	あり

サービスが利用可能で，かつ，Web サービスの応答から点数やフィードバックを取得するように教師が設定している場合に限り，受験時に点数とフィードバックを自動書き換えするものである．小テストの点数やフィードバックは受験後にはいつでも教師が書き換え可能であるほか，それらを当該学生がいつから閲覧可能になるのかも教師が設定可能であるため，授業の性質や状況に応じてさまざまな利用パターンが可能である．

また，自動採点を利用せずに目視評価による手動採点をしたい場合や，自動採点したうえで教師による採点結果で書き換えたい場合等に，提出物を個別にダウンロードして，個別に点数を書き換えるだけでなく，一括ダウンロードや一括点数書き換えが必要となる．点数の一括書き換えについては Moodle の既存機能の CSV インポート等で可能であるが，提出物の一括ダウンロードは小テストモジュールに関しては既存機能がなかったため，プログラミング問題独自の提出物一括ダウンロード機能を実装した．ダウンロード機能の詳細は本稿では述べないが，小テストの設定で同じ学生が複数回受験できる設定もあるため，最初の提出物のみダウンロード，締切前の最後の提出物のみダウンロード，すべての提出物をダウンロード等ダウンロード対象を柔軟に設定してダウンロードできるようにした．

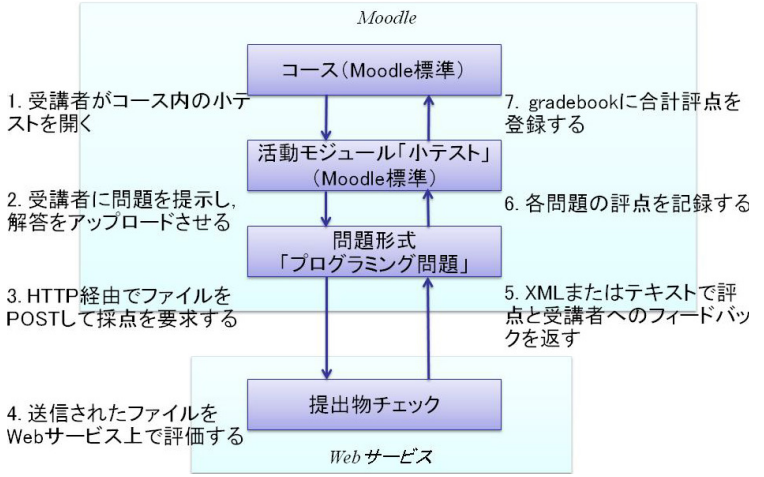

図 5　小テスト全体の処理の流れ
Fig. 5　Processing flow of whole quiz.

4.6　実装の詳細

プログラミング問題は Moodle における小テストの問題の種類の 1 つとして実装したため，以下のパスに実装しており，questiontype.php は `default_questiontype` クラスを継承して実装している．

`<Moodle ルートディレクトリ>/question/type/upchecker`

プログラミング問題を含む小テストが受験された場合の処理の流れは**図 5** のようになっており，受験者によって提出されたファイルおよび必要な情報を Web サービスに送り，得られた採点結果やフィードバックを Moodle 内で評点およびフィードバックとして記録する．評点を小テストモジュールに渡す処理は，`API grade_responses(&$question, &$state, $cmoptions)` をオーバライドし，引数`$state` のメンバ変数`$state->raw_grade`, `$state->grade` を設定することにより行っている．

Web サービスの利用にあたっては PEAR::HTTP_Client ライブラリを使って HTTP 通信を行っており，Web サービスからの応答が一定時間内に得られなかった際のタイムアウト処理のため，HTTP_Client コンストラクタにサーバへの接続確立までの待ち時間 `timeout` とサーバ接続後に応答を待つ時間 `readTimeout` を指定しており，所定時間内に応答が得ら

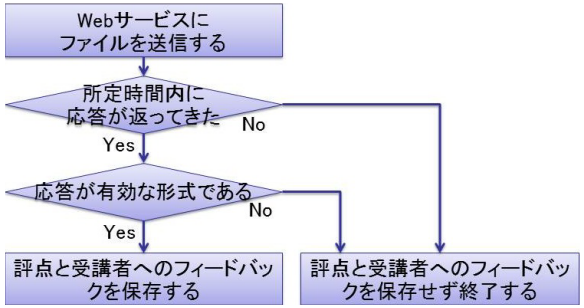

図 6　提出時の応答確認
Fig. 6　Flow of checking response on submission.

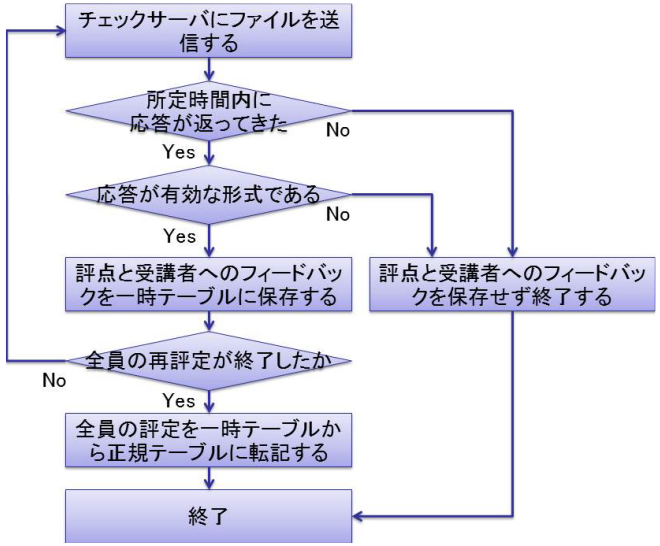

図 7　再評定時の応答確認
Fig. 7　Flow of checking response on regrading.

れない場合や得られた応答が有効な形式でない場合には，評点やフィードバックとして保存せずに終了することとした（**図 6**）．現時点では `timeout` を 10 秒，`readTimeout` を 60 秒と設定しているが，調整の必要はあるかもしれない．また，個々の受験時ではなく教師が

表 6　JSON 形式へのエンコード例
Table 6　Example encoding into JSON format.

{“filename”:“⟨提出ファイル名⟩”, “origname”:“⟨Moodle サーバ上でのファイル名⟩”,“serverresult”:“⟨サーバからの応答⟩”,“feedback”:“⟨受講者へのフィードバック⟩”}

図 8　教師への結果表示画面例
Fig. 8　Screenshot of quiz results for teacher.

評価基準を変更するなどして「再評定」を行う際には，すべての受験データに対して再度 Web サービスを使って提出物チェックをやり直すことになり，Web サービスが通常の受験時よりも高負荷になり正常な応答が得られなくなる可能性がある．途中までが正常で，それ以降が正常でないような再評定結果を保存しないよう，提出物チェックの結果をバッファリングして，正常な応答が揃った段階で正規の再評定結果として保存するように実装している（**図 7**）．

Moodle の小テストの解答テーブルには評点と受験者へのフィードバックの 2 つのフィールドしかないため，サーバからの応答（XML またはプレーンテキスト），提出されたファイル名，Moodle 内に保存したファイル名，受験者へのフィードバックを JSON 形式でエンコード（**表 6**）して，フィードバックのフィールドに保存しておき，教師への結果表示画面では，`API response_summary` をオーバライドして，上記の JSON データをそのまま表示する代わりに，提出されたファイルのダウンロードリンクを表示するようにした（**図 8**）．また，提出されたファイルはサーバ上ではシステム内部処理用のファイル名で格納されるが，ダウンロードする際には，元の提出ファイル名でダウンロードできるように実装した．

5. 利 用 事 例

4 章の実装を，著者らの所属大学のいくつかの授業科目で利用した．ここではプログラミング演習科目における利用事例を中心とし，加えてその他のいくつかの授業科目での利用事例を紹介する．

5.1 C プログラミング演習での利用

4 章の実装を利用して，C 言語のプログラミング演習科目における受講生の自習環境を試験的に提供した．具体的には授業中に課したプログラミング課題に関して，その解答プログラムのソースコードを送信すると，Web サービス内でコンパイルと実行を行い，教師側があらかじめ用意した正解例との差分を表示することとした（**図 9**，**図 10**）．従来，この授業では教室内でのみ利用できるコマンドライン版のチェックスクリプトを提供していたが，これを Web 化して教室外や学外からでも利用できるようにしたことになる．

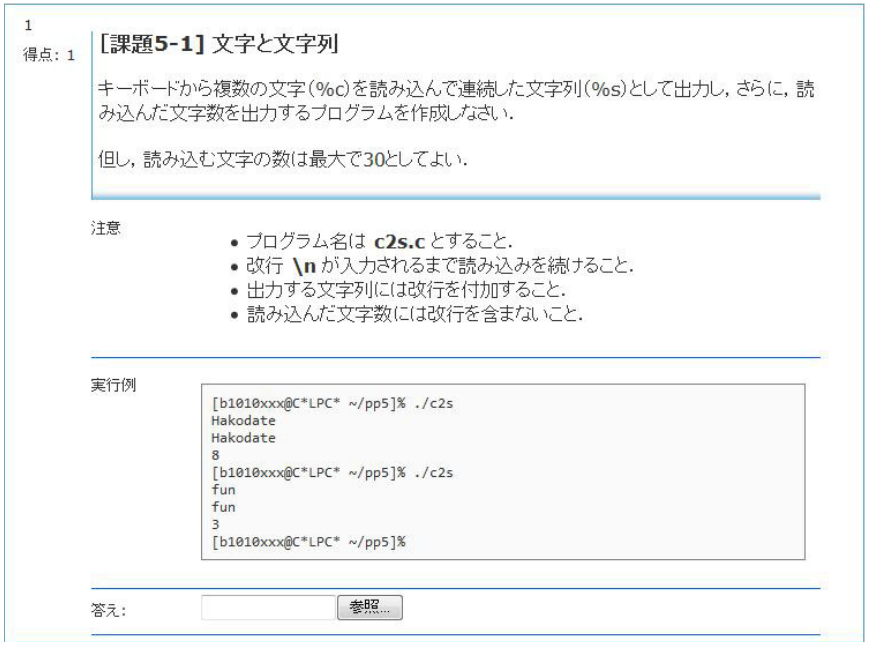

図 9　小テスト受験画面
Fig. 9　Screenshot of quiz attempt.

5.1.1 Web サービスの実装例

元々用意されていたコマンドライン版のチェックスクリプトは Ruby で記述されており，あらかじめ授業担当教師により用意されたテストケースを用いて，受講生から提出されたコンパイル済みのプログラムを実行し，結果が教師の想定したものと一致するかどうか，一致しなければ差異を表示するものであった．これを Web サービスとして提供するための仲介スクリプトを PHP で記述し，ソースプログラムを提出させて PHP スクリプト内でコンパイルしたうえで，既存のコマンドライン版チェックスクリプトを実行するように実装した．ソースプログラムを提出させることにしたのは，学生がどのような OS や環境で自習する場合にも対応可能とするためである．Web サービスを稼働させるサーバには Apple 社の Mac mini 1 台（Intel Core 2 Duo 2 GHz，Mac OS X 10.5.8）を使用した．

チェック結果は，Moodle 上に小テスト受験に対するフィードバックとして保存される．自習用であるため，このフィードバックは 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よう設定した（図 10）．

5.1.2 授業での利用

著者らの所属大学で 2010 年度後期に開講された C 言語のプログラミング演習科目（1 年次必修科目）において，演習課題がひととおり終わった後，期末試験に向けての自習用の環境として試験的に提供した．この授業は約 40 人 ×6 クラスで開講されており，受講生は全

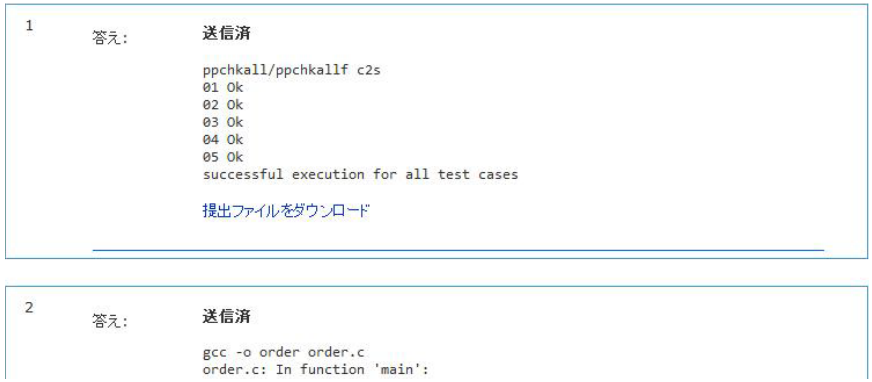

図 10　小テスト受験結果画面
Fig. 10　Screenshot of quiz result.

表 7　自習環境を利用しましたか（88 件中）
Table 7　Did you use the self-learning environment? (88 answers).

件数	（割合）	回答
3	（3.4%）	かなり利用した
51	（58.0%）	少しは利用した
28	（31.8%）	利用していない
2	（2.3%）	その他/分からない
4	（4.5%）	未回答

クラスで約 250 人であるが，開講曜日/時限が異なるため，授業中に利用する場合は同時利用は 40 人程度である．この授業では Moodle 自体は元々利用しているが，課題提出等には用いていない．

授業では演習 1 回に 4〜6 個程度の課題を課しているため，自習環境も演習の各回に対応させて，1 つの小テストページに 4〜6 個のプログラミング問題を載せることとした．試験的な提供であったため，多くの受講生はこの自習環境を教師の説明に従って 1 回の授業中に少し試した程度にとどまったが，103 人の受講生が総計 1,321 課題分利用しており，1 人あたり平均 12.8 個の課題で自習環境を利用したことになる．

自習環境としての提供であったため，授業担当教師は環境の準備と提供前の動作確認を行ったのみであり，受講生の利用結果の確認は基本的に行っていない．

利用後に受講生を対象として使用感等についてのアンケート調査を実施した．

5.1.3　アンケート結果

アンケートへの回答は任意で行い，回答件数は 88 件であった．これはこの授業科目の全履修者数の約 1/3，自習環境利用者数の約 85%である．自習環境を利用したかどうかの質問に対しては，「かなり利用した」と「少しは利用した」を合わせて回答者の 6 割程度が利用したと答えている（**表 7**）．

利用した人に対する質問項目で，この自習環境が復習や試験対策に役立つかを聞いたところ，「とても役立つ」と「少しは役立つ」を合わせて 9 割の回答者が役立つと答えており，受講生は主観的な評価として十分な有用性を感じている（**表 8**）．

また，同じアンケートで，従来のコマンドライン版のチェックスクリプトや従来の課題提出方法の代わりに，今回提供した環境を授業中のチェックおよび課題提出用に導入することについて聞いたところ，導入推進側の意見が 68.2%である一方，反対側の意見も 15.9%と一定程度あった（**表 9**）．反対側の意見として，『コマンドラインでの操作も覚えるべき』『コマンドラインでプログラムを組んだあと，提出のためにブラウザを使うのは操作が面倒』等の意見があった．

表 8　自習環境は授業の復習や試験対策としてどの程度役立つと思いますか（利用した 54 件中）
Table 8　Is the self-learning environment usaful for reviewing and exam preparation? (54 answers).

件数	（割合）	回答
24	（44.4%）	とても役立つ
26	（48.1%）	少しは役立つ
2	（3.7%）	あまり役立たない
0	（0.0%）	まったく役立たない
2	（3.7%）	その他/分からない

表 9　今回の自習環境を授業中のチェックおよび課題提出に導入することをどう思いますか（88 件中）
Table 9　Should we introduce the self-learning environment for checking and submitting answers? (88 answers).

件数	（割合）	回答
14	（15.9%）	絶対導入すべき
46	（52.3%）	どちらかといえば導入した方がいい
10	（11.4%）	どちらかといえば導入しない方がいい
4	（4.5%）	導入すべきではない
10	（11.4%）	その他/分からない

表 10　自習環境について良かったと思う点を記述してください（88 件中）
Table 10　Describe merits of the self-learning environment (88 answers).

件数	（割合）	回答
48	（54.5%）	自宅/学外/教室外から利用できる
4	（4.5%）	チェックが早い
4	（4.5%）	分かりやすい
3	（3.4%）	まとめてチェックできる
3	（3.4%）	復習できる
6	（6.8%）	その他
20	（22.7%）	未回答

この自習環境自体の良かった点を自由記述させたところ，教室外/学外/自宅から利用できることと答えたのが，全回答の 54.5%，利用した人の 59.3%と多かった（**表 10**）．これは既存のコマンドライン版チェックスクリプトがプログラミング科目用の教室からしか利用できないことによるものであり，プログラミング問題そのものの直接的な利点ではないが，チェックスクリプトを Moodle に組み込んだことによる派生的な利点である．もちろん，教師側が利用可能範囲を制限したい場合は，Moodle の小テストの既存設定機能で一定の制限

表 11　自習環境について今後改善した方が良いと思う点を記述してください（88 件中）
Table 11　Describe demerits of the self-learning environment (88 answers).

件数	（割合）	回答
10	（11.4%）	課題 1 つずつ提出したい
8	（9.1%）	動作の安定性，不具合，重いときの対処
5	（5.7%）	使い方や使い方の説明
2	（2.3%）	表示関係の不適切さ（リンクの場所や「小テスト」という表示）
4	（4.5%）	チェック結果の表示が分かりにくい
4	（4.5%）	操作方法として Web でのチェックや提出では演習がやりにくい
3	（3.4%）	（アップロードでなく）プログラムを直接入力したい
8	（9.1%）	その他/分からない
11	（12.5%）	特になし
33	（37.5%）	未回答

をかけることも可能である．

また，自習環境の問題点を自由記述させた（**表 11**）ところ，同科目の演習 1 回に 5 問程度の課題を課していることに対応させて自習環境でも演習 1 回分の問題数ずつ提出ページを用意していたが，むしろ課題を 1 つずつ提出したいという意見が目立った．その他，チェック結果がうまく見られないページがあったこと，使い方の説明が不十分だったこと，チェック結果の表示が分かりにくいことをあげている回答が，それぞれ数件ずつあった．これらはプログラミング問題自体の問題ではなく，自習環境の提供の仕方の問題であるが，使い方が分かりにくいという問題は他の利用状況でも発生しうると考えられ，注意が必要である．

5.2　Java プログラミング演習での利用

本稿の投稿段階で運用途中であるが，前節で述べた自習環境の提供ではなく，Java 言語によるプログラミング演習科目において自動採点および課題提出としての利用を開始している．

この演習は 5.1 節で紹介した C プログラミング演習の次のセメスタに受講する授業科目であり，約 30–40 人 ×5 クラスで開講されている．受講生は全クラスで約 180 人おり，開講曜日/時限の関係で最大時約 100 人が授業中に同時利用している．全 14 回の演習で各回に単位取得上必須となる課題 1 と必須ではない課題 2 があり，課題 1 はもちろんのこと，課題 2 についても半数以上の学生が取り組んでいる．また，この演習でも受講学生に対するアンケート調査を行っており，その結果の一部を**表 12**，**表 13** に示す．担当教師へのヒアリングも行っている．

この演習では学生に課している演習課題ごとに学生の提出するプログラムの形式的なチェックと動作のチェックの両方を行う Web サービスを教師が用意しており，自動採点と自動フィードバックの両方を学生が提出時点で閲覧できるようにしている．ただし，自動採点の点数はあくまでも「仮採点結果」であるとし，別途教師が手動で採点した結果を成績に反映させることとしているが，自動採点による「仮採点結果」も学生にとって課題のできを認識するための有用な目安として働いていると考えられる（表 13）．

表 12　提出ページの課題チェックの細かさは課題の進めやすさにどのように影響しましたか（121 件中）
Table 12　Is the detailed feedback helpful for learning? (121 answers).

件数	（割合）	回答
57	（47%）	細かくチェックされるので課題を進めやすかった
54	（44%）	多少は課題を進めやすかった
4	（3%）	課題の進めやすさには関係ない
5	（4%）	細かく指定され過ぎていて課題を進めにくかった
1	（0%）	分からない

表 13　自動採点機能は演習に役立つと思いますか（121 件中）
Table 13　Is the auto-scoring system helpful for your learning? (121 answers).

件数	（割合）	回答
64	（52%）	すごく思う
53	（43%）	多少は思う
3	（2%）	あまり思わない
0	（0%）	まったく思わない
1	（0%）	分からない

Web サービスとして用意している提出物チェックの主たる部分は，学生の提出したプログラムに対するテストプログラムとして作成しているが，動作に関するテストとは別に，ソースコード中の Javadoc コメントが演習中に指定された形式で記述されているかどうか，パッケージ名やクラス名が課題で指定されたとおりになっているかどうかなどの形式的なチェックも含めている．動作に関するテストも詳細に行われるため，ヒアリング結果からも教師は Web サービスで得られた「仮採点結果」を参考に，Web サービスでチェックされない箇所を手動でチェックすることで，提出物の採点作業を省力化できていると考えられる．

一方で，Web サービス上の提出物チェックは課題ごとに詳細に用意する必要があるほか，課題内容を変更すれば提出物チェックも変更する必要があり，提出物チェックの用意や維持のために教師には相応の負荷がかかる．ただし，教育と関連の低いシステム管理業務に煩わされるのではなく，あくまでも教育内容に深く関連する作業に専念できているともとらえられる．

5.3　その他の利用事例

「プログラミング問題」機能はプログラミング科目での利用を想定して設計および開発されたが，実際にはプログラミング科目以外でも利用可能であり，5.1 節および 5.2 節で述べたプログラミング演習科目以外に，プログラミングそのものを扱わない 2 つの科目においても利用を行った．これらの科目では学生へのアンケート調査等は行っていないが，授業担当教師へのヒアリング調査により，以下の利用状況および教師による評価が得られた．

1 つは，学生に提出させるレポートの形式が XML で指定されている授業科目で，Moodle 上に用意したレポート提出ページ（小テストモジュールを使用）にレポートが提出される際に，教師の指定した形式に従っているかどうかを Web サービスでチェックしたものである．この授業ではレポートの採点自体は教師が手作業で行っていて，Web サービスによるチェック結果は提出した学生が見ているだけで教師は確認していないが，「プログラミング問題」とレポート形式チェック Web サービスの利用は，形式の間違ったレポートの提出とそれに対する教師の対応を軽減するという意味で役に立っているといえる．

もう 1 つは，ソフトウェア設計に関する授業科目で，授業指定の UML モデリングツールで記述させた UML モデルを Moodle 上の提出ページに提出させ，ファイル形式が合っているかどうかと，提出されたモデルが出題時の基準を満たしているかどうかを Web サービスを用いてチェックしている．この授業でもレポートの採点自体は教師が手作業で行っているが，採点時の参考データとして Web サービスによるチェック結果も参照している．ファイル形式のチェックに関しては先に述べた XML 形式のレポートの場合と同じ効果があると考えられ，出題時の基準チェックに関しては採点支援という面で役立っているといえる．この授業ではどの種類の UML モデルを扱ったかによって，レポート提出ページでチェックすべき内容が異なるため，教師の管理下にある Web サービスでチェック内容を柔軟に変更できることが効果を発揮している．

6. 考　　察

6.1　システム管理負担に関する考察

コース管理システムのシステム管理者および授業を担当する教師にとってのシステム管理負担の側面について考察する．

6.1.1　コース管理システムのシステム管理者

4 章で述べたような実装をコース管理システムに適用することは必要になるものの，運用中のシステム管理業務の増大は多くないものと考えられる．Web サービスとの連携によってコース管理システムに管理者業務を生じるような問題が発生する可能性はないとはいえない．

4.6 節で述べたように Web サービスとの通信にはタイムアウトを設けており，Web サービス側に異常が生じた場合でも Moodle 自体への影響が少ないように実装している．

提出物チェックにつど Web サービスを利用することにより，従来よりもネットワークアクセスが増加することになる．これは提出物のサイズや提出頻度，コース管理システムと Web サービスを提供しているサーバとの間のネットワーク的な距離等が影響する．これらは教師や授業内容によって異なるため，システム管理に影響しないとはいいきれないが，本稿で紹介した利用事例の範囲内では，コース管理システムの他のネットワークアクセスと比べて顕著のものは現れなかった．

6.1.2　授業を担当する教師

教師は Web サービスを設置，運用，および，管理する必要がある．Web サービスは提出物チェックのみに限定した単機能のものであるため，独立して稼動可能な一般的なシステムと比べて管理すべき対象がきわめて少なく，管理業務も多くはない．5 章で述べた各利用事例では導入後にシステム管理に相当する業務はまだ発生していない．もちろん，Web サービスのために独自に Web サーバを運用する場合には相応のサーバ管理業務は必要となる．サーバ管理業務を軽減するために何らかのクラウドコンピューティングサービスを利用するという選択も考えられうる．

4 章で述べた実装では，コース管理システムである Moodle からの Web サービスの利用は基本的に教師権限でしか設定できず，Moodle を稼働させているサーバ以外からは基本的に Web サービスにアクセスする必要がないこと，また，Web サービスは教師もしくは教師に近い立場の人物が管理する場合がほとんどであると考えられることから容易にセキュリティ対策を施すことができる．また，Moodle から Web サービスに渡す情報や Web サービスに保存する情報についても，Moodle 上の教師および Web サービスを用意する側で制御可能であるため，情報流出等に関する対応も同様である．ただし，Web サービスを提供するためにクラウドコンピューティングサービス等を用いる場合は相応の対策が必要になると考えられる．

6.2　授業における提出物チェック方法に関する考察

Web サービスによって提出物チェックを提供することについて考察する．

6.2.1　受講生や教師の利便性

コース管理システム自体をつど改造しなくても教師が独自の提出物チェック方法を組み合

わせることができるようになったことで，教師がコース管理システム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かけることなく，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，開発言語も限定されず，Web サービス自体はコース管理システムと別のサーバで稼働させてよいため，Web サービスの稼働環境についても自由度が高い．

また，教師が用意する Web サービスの実装内容次第では，コース管理システム単体では自動抽出しにくい不正解例等の統計情報の自動抽出が可能であり，授業への効率的なフィードバックが可能であると考えられる．

6.2.2　教材の一体的管理

授業での 1 つの課題がコース管理システム上の問題記述と Web サービス内の課題チェック処理に分離されるため，長期的な授業科目運用を考えると本来 1 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．今後の可能性として，課題チェック自体は Web サービス上で実行するが，個々の課題に関する課題チェックのための情報を Moodle 上のコースファイルとして保持し，Web サービス内からコースファイルにアクセスしながら課題チェックを行うということも考えられる．

7. 今後の予定

本稿で紹介した利用事例の中では 5.2 節で紹介したものが「プログラミング問題」の最も本格的な利用形態となっている．この事例の利用結果をさらに分析し，授業運用や教育内容への影響等の視点から引き続き評価を行っていく予定である．

また，開発したプログラミング問題モジュールは Moodle コミュニティへのオープンソースでの公開を予定している．

8. ま と め

汎用的なコース管理システムを用いて授業自体の管理を行いながら，個々の授業ごとに特有の提出物自動チェック機構を Web サービスの仕組みを用いて容易に組み込めるようにする連携方法を提案し，これをオープンソースのコース管理システム Moodle に実装した．また，実際に教師が授業ごとに用意する提出物チェック用 Web サービスと連携させて運用した利用事例を報告した．

利用形態は個々の授業や教師に任されている面が多く，この連携方法自体の評価もそれらに依存して網羅的ではないが，すでに実践されている範囲でも一定の有効性が確認できたと考える．

参 考 文 献

1) Dougiamas, M.: Moodle, available from ⟨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) 渡辺博芳，高井久美子，武井惠雄，古川文人，及川芳恵：大学の教育基盤としての CMS と個別の学習支援システムをどう連携するか？，情報処理学会第 2 回 CMS 研究会，pp.17–22 (2006).
4) 白木幸宏，菅尾貴彦，中野裕司，喜多敏博：CAS 統合認証下における学習支援ツールの開発，情報処理学会第 2 回 CMS 研究会，pp.37–44 (2006).
5) 奥田麻衣，石田三樹，越智泰樹，平嶋　宗：ICT の活用と論述力支援の実践，情報処理学会研究報告 2008-CE-97，pp.75–80 (2008).
6) 鈴木恵二，伊藤　恵，齋藤朝輝，奥野　拓：高度 IT 人材育成システム開発と e-ラーニングによる Java スキルアップ，情報処理学会・コンピュータと教育研究会情報教育シンポジウム SSS2005 プレカンファレンス，pp.28–31 (2005).
7) 渡辺貴充，伊藤　恵：テスト駆動開発を用いた e-learning システムの学習評価，日本ソフトウェア科学会第 25 回大会予稿集 CD-ROM，No.5C-1 (2008).

(平成 23 年 3 月 28 日受付)
(平成 23 年 9 月 12 日採録)

伊藤　　恵（正会員）

昭和 45 年生．平成 10 年北陸先端科学技術大学院大学情報科学研究科情報システム学専攻後期課程修了．同年北陸先端科学技術大学院大学情報科学研究科助手．平成 13 年より公立はこだて未来大学システム情報科学部情報アーキテクチャ学科講師．博士（情報科学）．日本ソフトウェア科学会，教育システム情報学会，情報処理学会ソフトウェア工学研究会，情報処理学会 CLE 研究会，IEEE-CS 各会員．

美馬　義亮（正会員）

昭和 33 年生．昭和 59 年東京大学大学院理学研究科情報科学専攻修士課程修了．同年日本アイ・ビー・エム（株）入社．システムソフトウェア，ユーザインタフェースの研究に従事．平成 12 年より公立はこだて未来大学大学システム情報科学部講師．平成 17 年より同大学助教授．平成 19 年より同大学准教授．インタラクティブシステム，情報デザインに関する研究に従事．ACM，ソフトウェア科学会，日本デザイン学会，日本認知科学会各会員．

大西　昭夫

昭和 51 年生．平成 12 年北海道工業大学工学部卒業．同年アルプス電気（株）入社．車載電装ファームウェアの開発に従事．平成 14 年より文教向けシステム会社にて e ラーニングシステムの開発に従事．平成 18 年より 2 年間，北海道東海大学（現東海大学札幌キャンパス）にて非常勤講師．平成 19 年より（株）VERSION2 を創業し代表取締役に就任．外国語メディア教育学会，北海道英語教育学会，日本ムードル協会各賛助会員．