Technical Reports on Information and
Computer Science from Kochi
Vol. 1 (2009), No. 2

Web シラバスシステムにおける OpenID 利用の試み

吉田 俊雄[1], 菊地 時夫[2]

要旨

高知学園短期大学では，授業シラバスの公開手段として Plone/ArcheTypes を応用した Web シラバスシステムを導入している．このシステムはインターネット経由でその設置場所を意識せずに利用でき，これまでに資格取得のための単位確認インタフェースを導入するなど機能拡張性を持っている．本研究では，ここに新たに学生の個人認証を導入し，履修歴確認などの機能拡大への対応を目指した．システム標準の認証機能では，学生 DB の再設置や情報漏えいが問題となるため，本研究では Web 上で安全に認証を実現する技術である OpenID を用いて学内認証をシステムで再利用する方法を試みた．OpenID には認証機能を提供する OpenID Provider，認証機能を利用する Relying Party，認証されるユーザの三者があり，それぞれ大学の学生 DB，Web シラバスシステム，学生が対応するが，高知大学においてそれらに近い条件の試験環境を構築し，OpenID 利用の実現性を検証した．構築にあたっては Apache httpd の ReverseProxy 機能が問題となったが，設定を調整することで解決し，OpenID によるログイン成功に至った．今後，認証に加えて履修歴などを利用するにあたりセキュリティの強化が必要であるが，その一環として安全なプロトコルを提案し考察した．

1 はじめに

1.1 概要・目的

高知学園短期大学においては，授業シラバスの公開手段として Plone/ArcheTypes を応用したコンテンツ管理システム（CMS）を導入している．本システムにより，授業シラバスを作成し，登録し，公開し，保守・管理し，運用するという一連のプロセスが整備・効率化された [1][2]．さらに，CMS の拡張性を用いて新しい機能を追加していくことが可能となっており，これまでに資格取得のための単位確認インターフェースが拡張導入された [3]．本研究の実態も，CMS の拡張性を用いたシステムの可能性を調査・検証したものである．

本システムは，高知学園短期大学の授業シラバス公開という学内向けサービスを提供しているものの，物理的には学外である高知大学に設置されている．このような配置がとれることの利点には，サーバコンピュータの設置やメンテナンスといった作業をシラバス公開業務から切り離し，コンピュータ管理や運用資源の在り方を大学の状況にあわせて選択できるようにすることがある．本研究ではこうした外部システムの利点を損なわずに学生データ利用を実現する方法の模索・検証を焦点とし，Web 上で安全に認証を実現する OpenID 技術の利用を試みた．

1.2 OpenID

本研究の主要な要素の一つである認証技術である．本技術を利用すれば，ウェブアプリケーションから見れば自前の認証機構を用意する必要がなくなり，ユーザから見れば複数アカウントの作成や管理を行う必要がなくなる．OpenID の開発コミュニティサイト [4] にその仕様が公開され，また仕様に基づいた様々な実装が紹介されている．

1.3 Plone

Plone は高機能アプリケーションサーバ Zope 上で動作するコンテンツ管理システムである．Zope においては，ウェブサーバ，データベース，ページ生成，ログイン機構，といったウェブサイト構築・管理・運営のための一通りの機能が揃っており，なおかつ拡張性にも富ん

[1]高知大学理学研究科数理情報科学専攻
Graduate School of Science, Kochi University
[2]高知大学情報科学教室
Department of Information Science, Kochi University

でいる．その拡張機能の一つとして，よりユーザフレンドリーなインタフェースを提供するのが Plone である．高知学園短期大学の Web シラバスシステムのほか，高知大学のサークルページも Plone で作成されている．

OpenID 認証において認証機能を利用する役割を担う．

1.4 Gracie

Gracie は OpenID 認証においてアカウント発行と認証機能を提供する認証サーバで，プログラミング言語 Python で実装されている．Plone と同様オープンソースであり，GNU GPL に基づいて無料での使用や改変が可能である．本研究において Plone とは対の存在となる．

Gracie は Password Authentication Module (PAM) により UNIX 認証を取り入れ，OpenID を発行する．

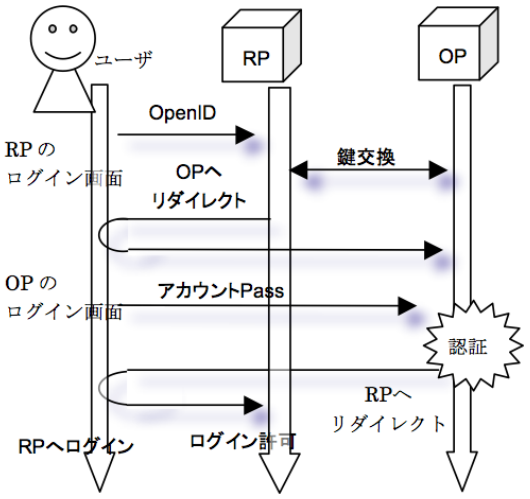

図 1. OpenID 認証の流れ

2 OpenID の概要

2.1 OpenID の用語

OpenID において使用される用語についていくつか述べておく．

Identifier (ID) OpenID アカウントのユーザ ID であり，“http” または “https” で始まる Uniform Resource Identifier (URI) もしくは Extensible Resource Identifier (XRI)

User-Agent (UA) ユーザが使用するウェブブラウザ

Relying Party (RP) OpenID 認証を利用するウェブアプリケーション

OpenID Provider (OP) OpenID を発行し，ウェブアプリケーションの要求に応じて認証や認可を行うサーバ

2.2 OpenID のプロトコル

OpenID は次の三者が関わるようになっている．

ユーザ: アプリケーションを利用する

RP: ユーザにサービスを提供し，OP に認証を依頼する

OP: RP に認証を提供する OP

OpenID 認証は図 1 のような手順となる．

まず，ユーザはサービスを利用するために UA を通じて RP に ID を提示する．次に，RP が ID をもとに問い合わせ先となる OP の URL を見つける．そして，RP が OP との間で信頼の担保となる鍵交換を行い，OP におけるユーザの認証状況の検証を開始する．最後に，RP が必要に応じて UA を OP へリダイレクトするなどし，OP 側から ID 利用の認可を受けつつ，認証済みであるのを確認できたらユーザへのサービス提供を開始する．

2.3 既存技術との比較

アカウントの発行と認証の機能をアプリケーションから分離し，ログインセッションをアプリケーション群で再利用するような，いわゆるシングルサインオンの技術は OpenID が登場する以前から存在している．

それら既存技術においては，認証局と認証利用アプリケーションとユーザがあらかじめ認証共同体を形成する必要があり，共同体に所属する特定ユーザと特定アプリケーションの間で認証の再利用を行う．

OpenID においては，認証共同体を構成するという概念はなく，グローバルネットワーク上の三者が認証を行うために OpenID プロトコルにしたがうようになっている．OpenID プロトコルはユーザ，アプリケーション，認証局それぞれの役割を担うための制約が少なく，既存システムに追加的に組み込みやすい反面，ユーザと認証局が一体であるような状況も許してしまうので，認証された事実をどこまで尊重するべきかという問題が生じてくる．

また，OpenID の ID はグローバルなネットワーク上のリソースを一意に示す URI であるということも特筆すべきである．ID に要求される一意性に加え，ID がア

カウント発行組織も表明するという役割を演じていて，OpenID プロトコルの単純性に寄与している．

2.4 通信の信頼性

RP と OP の間で Diffie-Hellman 鍵交換を行い，お互いが交換するパラメータを鍵署名によりチェックすることで改ざん検出を行う．また，ユーザと OP の間での認証も，SSL 暗号化技術を利用することで安全にログインセッションを確立できる．

2.5 運用時における注意点

OpenID は，所詮は ID +パスワード方式の認証に過ぎない．したがって，その種の認証に対して有効なフィッシング攻撃やリプレイ攻撃は OpenID 認証に対しても有効である．

インターネットを介してやりとりされる認証過程で，認証受理のアサーションが攻撃者に盗聴されるとリプレイ攻撃が成立してしまう．それに対しては，アサーションの審査を 1 回に限定する属性値 nonce を付加するといった対策が推奨されている．

また，OP の実装形態によっては，発行される OpenID 自体にユーザ名がそのまま含まれることがあり，ログイン先のサイトも明らかとなるため，アカウントを防御するのはパスワードだけという状況が生まれる可能性がある．

2.6 OpenID Attribute Exchange

OpenID の拡張仕様である OpenID Attribute Exchange[5] において，認証情報に限らず様々なユーザ属性値を扱うための仕様が定められている．これは，RP が OP に要求を出すことによって，ユーザの許可のもと，ユーザ属性値を指定して受信したり OP 上のユーザ属性値を変更したりする手順を示したもので，この仕様により，OpenID 認証を用いて Web シラバスシステム上で学習記録のデータを受け渡す手がかりとなる可能性がある．

3 試験環境の構築

3.1 概観

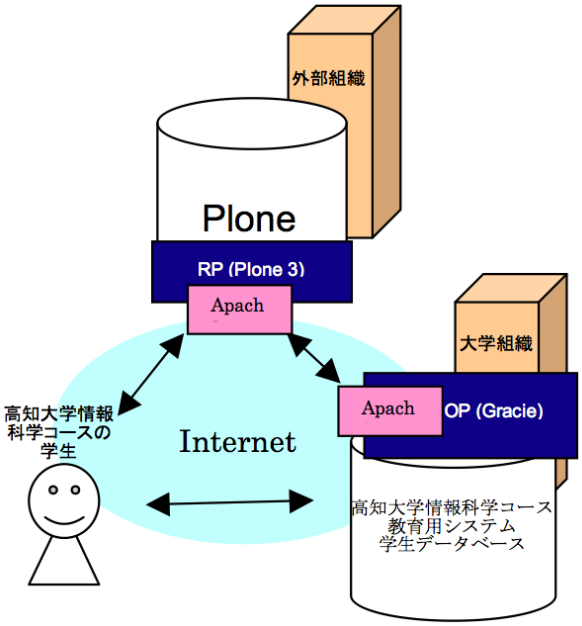

図 2. 試験環境概観

図 2 に試験環境の概観を示す．目標の環境では，RP が Web シラバスシステム，OP が高知学園短期大学の学生データベースとなる．Plone 上で Web シラバスシステムは実装可能であり，高知大学情報科学コース教育用システムの学生データベースは実際に大学で運用中のものであるから，試験環境は目標に近い条件といえる．

3.2 構成の解説

3.2.1 Plone 3

Plone はバージョン 3 から OpenID を標準サポートしている．Plone 配布サイト [6] よりバージョン 3 以降の最新版インストーラをダウンロードし，適当なディレクトリに展開し，インストールスクリプトを実行する．

Plone 3 のインストールマシンとしては高知大学菊地研究室で独自に運用しているサーバコンピュータを採用した．マシン構成は次の通りである．

ハードウェア	PowerMac G5
プロセッサ	PowerPC 2.5 GHz
メモリ	3.5 GB
OS	Max OS X 10.4.11
ウェブサーバ	Apache 2.0.63

このマシンでは地球環境情報学研究室サーバや高知大学気象情報頁バックエンドといった複数のウ

ェブサービスがすでに稼働中であった．今回新たに追加する Plone 3 にアクセスするための URL は http://shippo.lab.tkikuchi.net/portal とした．

Plone 3 インストールにおいては，Plone 以外の必要アプリケーションを個別にインストールする方法と，全ての必要アプリケーションをセットでインストールする方法が選択できる．

個別にインストールする場合，それぞれのアプリケーションのバージョンに制約があり，それらのインストール状況によっては，アップデートが必要だったり複数のバージョンを管理することになったりするので注意が必要である．

全ての必要アプリケーションをセットでインストールする場合は，必要アプリケーション相互の整合が解決済みであるためインストールスクリプト実行という 1 つの行程でセットアップ完了に至る．

インストールが完了したら，サイト URL にアクセスして管理者アカウントでログインし，OpenID 認証を有効にする．このアクティベーションのみで，Plone 3 は RP のログインサービスを提供するようになる．

3.2.2 Gracie

OP 側となる Gracie を動作させたのは Plone 3 インストールマシンとは物理的に異なるマシンであり，お互いは学内ネットワークかインターネットを通じて相互通信が可能である．Gracie にアクセスするための URL は http://mail.is.kochi-u.ac.jp とした．

Gracie 配布サイト [7] よりパッケージアーカイブをダウンロードした後，適当なディレクトリに展開しインストールスクリプトを実行する．Gracie をインストールしたマシンの構成は以下の通りである．

ハードウェア	富士通 PRIME POWER 250
OS	Solaris 10

本研究において Gracie を稼働させるにあたり，PAM ハンドルモジュールの変更，および日本語表示の対応を行った。

Gracie は Python のモジュールを利用することで PAM による UNIX 認証を組み込むようになっているが，Gracie が指定するモジュールはインストール済みであるにもかかわらずロードに失敗してしまったため，試験環境においては代替モジュールを使用するようにソースコードを改変し，差し当たって OP サービスを起動させるようにした．

また，日本語の表示に対応するようスクリプト上の簡単な変更を加え，かつサイトの説明を日本語訳のものに書き換えた．

Gracie への http アクセスは Apache を経由させるようにした．なお，Gracie に提供させる OpenID Identifier は http://servername/id/UNIX アカウントユーザ名という形式になっている．

3.2.3 Apache

Plone や Gracie はそれぞれ Zope や Gracie 自身のウェブサービス機能単独でも運用可能だが，試験環境においてはそれらへのアクセスに Apache を経由させている．このような構成にするには次のような理由がある．

- すでに Apache によるウェブサービスが稼働中である
- Apache の豊富な機能をサービスに付加できる
- Apache を残すことで引き続きその機能を利用できる

試験環境の構築を行った高知大学のネットワーク環境では Apache によるサービスがすでに行われている．執筆時点において Apache はウェブサーバとしてもっともオーソドックスであり，Apache サポートで運用中のサイトも多数あるため，追加されるウェブアプリケーションは Apache と連携させるのが自然である．

また，Apache はウェブサーバとして改良を重ねてきた歴史が長く，信頼性の高い，多機能で有用なサービスを提供できる．アクセス制御，正規表現解析，URL の書き換え，CGI 制御，データベース連携などといった機能は強力で，それらをウェブアプリケーションのために利用可能な状態にしておく利点は大きい．

さて，Zope や Gracie を Apache と連携させるにあたって，http スキーム標準の 80 番ポートはすでに Apache が使用中であるので，未使用な非特権ポートで Zope と Gracie のサービスを開始することとし，Apache の Virtual Host 機能を用いて 80 番ポートからそれぞれの割り当てポートへ中継するよう設定した．

そこで，それぞれのマシンにおいて Apache の設定ファイルに表 1,2 のように書き加え，Apache に反映させた．なお，ログファイルの保存先指定など細かい設定については省略してある．どちらの指定も中継には Apache のモジュールである mod_proxy を利用するようになっている．

表 1. Plone 側 Apache 設定

```
<VirtualHost *:80>
ServerName shippo.lab.tkikuchi.net
proxyPass / http://localhost:8580/
VirtualHostBase/http/shippo.lab.
tkikuchi.net:80/VirtulaHostRoot/
</VirtualHost>
```

表 2. Gracie 側 Apache 設定

```
<VirtualHost *:80>
ServerName mail.is.kochi-u.ac.jp
RewriteRule
^/(.*)$ http://127.0.0.1:8000/$1 [P]
</VirtualHost>
```

表 3. ID と GET 要求 URL の対比

OpenID Identifier	https://mail.is.kochi-u.ac.jp/id/tyoshida
GET_Method_URL	https://mail.is.kochi-u.ac.jp/portal?janrain_nonce=2008-09-19T11%3A37%3A38Z17MUVy&openid1_claimed_id=https%3A%2F%2Fmail.is.kochi-u.ac.jp%2Fid%2Ftyoshida&openid.assoc_handle=%7BHMAC-SHA1%7D%7B48d34af3%7D%7BvN6joA%3D%3D%7D&openid.identity=https%3A%2F%2Fmail.is.kochi-u.ac.jp%2Fid%2Ftyoshida&openid.mode=id_res&openid.response_nonce=2008-09-19T11%3A38%3A15ZkU2Q1z&openid.return_to=http%3A%2F%2Fshippo.lab.tkikuchi.net%2Fportal%3Fjanrain_nonce%3D2008-09-19T11%253A37%253A38Z17MUVy%26openid1_claimed_id%3Dhttps%253A%252F%252Fmail.is.kochi-u.ac.jp%252Fid%252Ftyoshida&openid.sig=UPq4bLxP7WhdvJIPUmUIpq20FVU%3D&openid.signed=assoc_handle%2Cidentity%2Cmode%2Cresponse_nonce%2Creturn_to%2Csigned HTTP/1.1

以上の設定で，Apache を経由したアクセスが可能となるが，この段階では OpenID 認証の際に Proxy Bad Header エラーを生じ，例外終了となってしまった．

OpenID 認証過程においてリダイレクト処理が必要になる場合があるが，その際，GET メソッドが必須となる．GET メソッドは URL に文字列を付け加えてリクエストすることでアクセス先に引数を送信するが，OpenID のプロトコル上引数の量が多く，結果として URL が長大化し，Apache に異常なプロキシーヘッダとして捕捉されてしまうのだと考えられる．表 3. は OpenID の ID とその認証過程で使用される GET 要求 URL の対比である．これを回避するためには，表 4. の一行を Apache の設定ファイルに書き加え，反映させる．

表 4. Apache の Proxy 設定

```
ProxyBadHeader    Ignore
```

なお，Apache のプロキシー機能を使用する場合は，モジュール `mod_proxy` を有効にする必要がある．それには，Apache のコンパイル時に静的に組み込むか，Load-Module 機能により動的に読み込むかさせる必要がある．

以上の設定により，Apache のプロキシー機能を適用した OpenID 認証が可能となった．

3.2.4 試験環境における OpenID 認証

図 3. Plone 上で OpenID 入力

構築された試験環境でのログイン画面遷移を図 3 に示す．Plone すなわち RP に対しては ID のみの提示でログインが実現する．なお，最新のプロトコルバージョンである OpenID2.0 においては OP のホスト名を提示するだけでもよい．ID 送信後，Gracie すなわち OP においてロ

グインセッションを確立するためにリダイレクトで遷移し，図 4. の画面で ID/Password によるログインを行う．予めログイン済みで RP での ID 利用が許可されていればこの段階は省略される．OP にログインし，RP 上での ID 利用の許可を行って Plone へのログインとなった様子を図 5 に示す．

図 4. 認証のため Gracie へ遷移

図 5. Plone へのログイン成功

4 OpenID を用いて Web シラバスシステムを利用するための手法提案

4.1 Web シラバスシステムと個人情報の問題

Web シラバスシステムにおいて履修歴などの学生個々の属性情報を利用すればアプリケーションの有用性は高められるが，システムは大学組織外のハードウェアで稼動しており，それらの情報は全て学外組織の者に見られるという前提に立たねばならない．

本研究では，このような状況を受け入れた上であえてそうした個人情報を扱うアプリケーションを実現させようという立場をとっている．その場合，組織外へ情報を送信するが，その情報が組織外においてすっかり記録されてしまったとしても情報流出は防がれる，という矛盾を解決せねばならない．

4.2 情報の無意味化

履修歴など（以下，利用情報と呼ぶ）は，氏名や住所などが結びついて個人を特定できる状態になって初めて個人情報となる．したがって，利用情報そのものが盗聴されたとしても，それが個人に結びつかなければ勝手に作ったデータと区別がつかなくなり無意味化する．そういうわけで，利用情報と個人特定データを完全に分離することで，上述したような矛盾の克服を目指す．

4.3 OpenID 認証への適用

OpenID 認証において利用情報と個人特定データを分離するためには，次のようにする．

- ユーザは ID ではなく OP のホスト情報のみを提示して認証開始する
- OP は RP にユーザの固有 ID ではなく，一回限り有効でランダムな一時 ID を生成して渡す

以上をふまえた上で，認証の過程全体を追っていくと次の 1〜6 のようになる．

1. ユーザは OP のホスト情報を提示する
2. RP はホスト情報に示された OP と鍵交換を行う
3. RP は UA を OP へリダイレクトする
4. ユーザは OP において ID とパスワードを使ってログインする
5. OP は UA を一時 ID とともに RP へ鍵の署名付きでリダイレクトする
6. RP は鍵を用いて認証完了の署名を審査し，一時 ID でのログインを許可する

ユーザが OP にログインしている間は OP 上でユーザと一時 ID の対応付けを行うが，ログアウト後は一時 ID とともにその対応付けを破棄するようにする．

これが本来の OpenID 認証と異なるのは，ID がランダムに生成されるため，ID とユーザとの間に相関性がない点である．ユーザが OP にログインしている間は相

関性が生まれるが，ログアウト後は，対応付けが破棄され，ID の持ち主についての手がかりは全く失われる．

信頼されない RP がこの認証過程で確かめられることは，ユーザは OP から一時 ID を発行してもらう権限を有しているという事実のみ，言い換えればユーザは確かに組織に所属しているという事実のみである．

4.4 提案手法についての考察

4.4.1 実装のバリエーション

一時 ID と学生アカウントの対応

ID とアカウントの対応表をそもそも発生させないようなプロトコルも考えられる．OP から認証結果を RP に返す段階で OP は anonymous な OpenID とともに利用情報を RP に渡すようにする．ここでいう anonymous な OpenID とは，OP に登録された異なるユーザ全てに対して全く同じ文字列として与えられる特殊な OpenID である．RP においては anonymous な OpenID を受け取った場合，RP 自身が anonymous OpenID ユーザに対し適当な ID やセッションを割り当ててログイン中のユーザ識別を行う．

このようにした場合，OP において対応表を作る必要がないため，OpenID と学生アカウントとのつながりは発生せず，追跡可能なタイミングが完全になくなる．ただし，OP 上において OpenID と学生アカウントの対応表がないということは，RP から追加的にユーザ属性を OP に問い合わせることが不可能なので，このプロトコルに従うアプリケーションは柔軟性がやや制限されてしまうと考えられる．また，このプロトコルにおいては OP 側にとどまらず，RP 側も OpenID の仕様にはない振る舞いを要求されることになる．

一時 ID の決定方法

OP 側で対応表を持つようにする場合，一時的な ID の決定について注意する必要がある．ID を発行する際は次のような点が条件となる．

- ID 使用者がその ID を所有し続けないように管理する
- ID そのものが ID 使用者のいかなる情報も表明しないようにする
- RP に同時にログインするユーザが重複しないようにする

以上を満たす上で，過去に生成して使用したどれとも一致しないランダムな文字列を生成して ID としてもよいが，同時にログインするユーザに関してのみ ID の重複に気をつけるようにし，一度生成した ID をむしろ積極的に使い回すようにして ID 生成・管理の処理負担を軽減できる可能性がある．すなわち，IP アドレスを動的に割り振る DHCP サーバのような処理を行う．このような ID においても，その使用者の匿名性はランダムである場合と同様に保たれる．

ただ，この方法においては，OP と RP でのログインセッションがそれぞれ独立管理されているために，RP 利用中のユーザ A が OP においてはログアウトしてしまった場合，新しく RP にログインしようとするユーザ B にユーザ A と同じ ID が OP から割り当てられかねないという問題があるので，導入には工夫が必要となる．

4.4.2 利用できる情報

学生の属性情報には氏名，住所，メールアドレス，電話番号など，それ自体が個人情報であったり，影響力を発揮したりするものもあり，これらについては利用情報とすることが出来ない．すなわち，提案手法で扱える利用情報にはそもそも限界がある．また，一つ一つの利用情報に意味がないとしても，長期間にわたってそれらが蓄積されていった場合に，統計的な意味合いが生じたり，データマイニングの対象になったりする可能性もある．

以上のことから，Web シラバスシステムに提案手法を取り入れる前に，扱っても問題ない情報とは何であるのか，またそれらを扱うことで最終的にどのような情報が RP にわたることになるのか慎重に吟味する必要があることに注意せねばならない．

4.4.3 RP で作成したデータの証明

RP 上アプリケーションのバグなどのために，例えばそこで作成した履修計画表に誤りがあると，ユーザはその誤った履修計画表を用いて履修申請を行い，登録ミスとなってしまう場合が考えられる．その時，作成した当時の一時 ID はすでに破棄されているため，ユーザに起因した登録ミスではないことを確認するための手段が必要となる．

RP と OP との間で履修登録期間中に維持し続けるような共有鍵を交換しておき，RP がその鍵を用いて計画表に署名を行う．ユーザは RP の署名のついた計画表をプリントアウトするなどして保存しておき，履修登録の申請に誤りのあることが分かった時に，署名付き計画表を当局に提出する．当局は正しい計画表であることを確認するとともに，OP の鍵を用いて計画表の署名をチェックし，RP で作成されたものであることを確認する．

5 結果・考察

試験環境において OpenID アカウントでのログイン成功により，Web シラバスシステムにおける学生認証実現の可能性が示された．さらに，既存のシステムに対する

根本的な変更ではなく機能付加によって実現した点，試験環境を実際に構築するにあたって使用した主要な技術は全て無料で利用できる点，構築に際して生じた問題は比較的容易に解決できる点などから，この方式の導入コストは低く済ませられることが明らかとなった．

今後は，試験環境を目標環境に置き換えるための検討，学生アカウントによるユーザ属性を使ったアプリケーションの検討，提案手法の実装と検証が課題となる．

提案手法はアプリケーションと認証が分離されてはじめて無理なく成立するものであると考える．また，組織外においてイントラ向けのサービスを行い，かつそのサービス上で組織員のデータを安全に扱いたいという特殊な要求においてはじめて意味を持つ手法ともいえる．

とはいえ，本研究における提案手法に限らず，そうした要求によく応えられる手法が確立すれば，Web シラバスシステムを複数の大学で共用したり，Web シラバスシステム以外のアプリケーションに応用したりといった利用可能性も考えられるようになるだろう．

謝辞

研究室の諸先輩方の先行研究なくして本研究は成り立ちえず，研究室仲間の励ましによって本研究をまとめる力を与えられました．また，著者のひとり（吉田）の生活を支え，日々の活力を与え励まし続けてくれた家族に心より感謝いたします．最後に，多くの素晴らしい技術を惜しげもなく世に公開し続け，多くの示唆と知恵と本研究に取り組む機会を与えてくださった，オープンソースソフトウェアの開発者に敬意を表します．

参考文献

[1] 木谷 由実・菊地 時夫: Plone/ArcheTypes を用いたシラバス作成システム, 高知大学理学部紀要, **27-F**, No. 2, 9pp, (2006).

[2] 須藤 藍子・菊地 時夫: Plone/ArcheTypes を用いたシラバス公開システム, 高知大学理学部紀要, **27-F**, No. 3, 9pp, (2006).

[3] 中橋 一真・菊地 時夫・濱田 美晴: Plone/ArcheTypes によるシラバスシステムの構築 —ORMapping による機能強化—, 高知大学理学部紀要, **29-F**, No. 1, 9pp, (2008).

[4] OpenID Foundation: OpenID http://www.openid.net/, (2007-2009)

[5] Hardt, D., Bufu, J. and Hoyt, J.: OpenID Attribute Exchange1.0-Final, http://openid.net/specs/openid-attribute-exchange-1_0.html, (2007)

[6] Plone Foundation: Plone CMS: Open Source Content Management, http://plone.org/, (2000-2009).

[7] Finney, B: Gracie, http://trac.whitetree.org/gracie/ (2007)