特集号
招待論文

教育用オープンソースソフトウェア群のローカライゼーションと共通翻訳メモリの開発

― 一貫性のある用語による教育支援システムを目指して―

常盤 祐司[†1]　出口 大輔[†2]　宮崎 誠[†3]　平岡 斉士[†4]　喜多 敏博[†4]　梶田 将司[†5]

[†1]法政大学　[†2]名古屋大学　[†3]畿央大学　[†4]熊本大学　[†5]京都大学

大学教育で利用されているオープンソースソフトウェアの多くは英語圏で開発されているため日本語への翻訳が必要となる．今後大学では複数のシステムにより教育支援のための IT 環境を構築することが予想されるが，システム間で用語が異なっているとユーザに違和感を抱かせる．この課題に対して，大学教育で頻出する用語および用例を包含する大規模な共通翻訳メモリを作成することによって一貫性のある翻訳が実現できると考えた．また，翻訳メモリによる翻訳では実現できないコンテキスト依存の翻訳を可能とする gettext 形式の PO ファイルを併用することにより，Java で実装された Sakai と PHP で実装された Moodle および Mahara をクラウドベースの翻訳支援システムである Transifex を用いて同一のプロセスで翻訳した．本稿では大学教育用オープンソースソフトウェアに対して一貫性のある翻訳を目指して行ったプロジェクトについて述べる．

1. はじめに

日本の大学，短大，高専を対象とした学習管理システム（Learning Management System）に関する調査でMoodleが独自開発システムや商用システムを抑え，最も高い利用率になったのは2008年である[1]．その後，教育支援システムとして，Moodleに加えSakai，Maharaなどのオープンソースソフトウェア（以下，OSS）の導入が進み，大規模公開オンライン講座（MOOC）基盤として利用が進んでいるOpen edX，CanvasなどもOSSである．このように，多様なシステムが利用できるようになってきたことから，今後大学では複数のシステムにより教育支援のためのIT環境を構築することが予想される．ただし，これらのシステムの多くは英語圏で開発されているため，日本国内で利用する場合には翻訳を行う必要がある．しかしながら，それぞれのソフトウェアを開発するコミュニティで独自の国際化（以下，i18n）および日本語への翻訳を含むローカライゼーション（以下，l10n）が行われているため，それらのシステムで使われている同一の用語の翻訳については一貫性が保証されていない．たとえばgradeといった用語は，Sakai CLE（以下，Sakai）では「成績」「採点する」，Moodleでは「評定」「評点」と翻訳されている．そのため複数のシステムを行き来して利用するようなユースケースではユーザが違和感を覚えることが懸念される．

この課題に対する解決策の1つが事例ベースの翻訳ともいえる翻訳メモリを使った翻訳である．筆者らは大学教育全域を対象とする大規模な共通翻訳メモリを作成することによって一貫性のある翻訳が実現できると考えた．本稿では大学教育用OSSにおいて一貫性のある翻訳を実現することを目指して行った翻訳プロジェクトの実践結果およびその可能性を述べる．

大学教育向 OSS において
一貫性のある翻訳を目指す！

2. 課題解決に向けて

日本Sakaiコミュニティ（以下，JaSakai）では翻訳をテーマとしたアンカンファレンスを2012年夏に開催した．そこでは大学向けOSSのMoodle，Mahara，Sakaiの翻訳の現状について参加者から報告が行われた．その際に浮き彫りになったことは，特定のボランティアが独力でそれらのOSSを翻訳しているという実情であった．それぞれのOSSが日本に導入された時点ではコミュニティ

が形成されておらず，当初はやむを得ない状況であったと思われるが，すでに10年以上経過した現時点においてこの状況は望ましいとはいえない．ひとりの翻訳者による体制では翻訳が主観的になりがちで，コーパスあるいは翻訳メモリのない状況ではOSSの翻訳に一貫性を持たせることは困難である．また，コミュニティで翻訳を行うとしても，その時点では広域に分散したチームで翻訳を行うためのツールがなく，翻訳者が自身のPCにインストールして利用するBenten[2]あるいはOmegaT[3]などで対応せざるを得ない状況であった．

こうした課題を解決するには，OSS群を横断したコミュニティによる取り組み体制の確立と翻訳支援ツールの調達が必要であった．今回，体制については，翻訳メモリによる翻訳，翻訳支援システム，Sakaiに代表されるJavaによるi18n実装，MoodleおよびMahara（以下，Moodle／Mahara）に代表されるPHPによるi18n実装などに関する知識を有するメンバを集めることができた．まずJaSakai幹事会からメンバを募り，Java実装，翻訳メモリによる翻訳および翻訳支援システムに精通したメンバがそろった．そして，このメンバのうち一人が以前所属していた機関の共同研究者を，PHP実装に精通したメンバとして勧誘した．どのメンバも所属する機関でSakai，Moodle，Maharaのいずれかを利用しており，本プロジェクトの成果をその機関が享受できるエコシステムが形成されることがモチベーションとなっている．

また翻訳支援ツールについてはCrowdin[4]およびTransifex[5]というWebアプリケーションを翻訳するためのWebベースの商用翻訳支援システムが利用でき，それらはOSS開発コミュニティに対して無償で環境を提供していることが分かった．併せて科研費採択により予算措置が確実になったため，「共通翻訳メモリを用いた教育用OSS群のコミュニティによる翻訳」プロジェクトが2013年4月に開始された．

3. プロジェクト概要

3.1 OSSと翻訳支援ツール

本プロジェクトでは複数のOSSが前提となるが，選択したOSSは国内にコミュニティが存在し，筆者らがコミュニティメンバとなっているMoodle, Mahara, Sakaiである．また，Moodle／MaharaがPHP，SakaiがJavaで実装されており，異なるi18nの影響を確認するという点も考慮した．つぎに翻訳支援ツールとしてはさまざまなOSSの翻訳に利用されており，筆者らの一人がすでに他のOSS翻訳プロジェクトで利用していたTransifexとした．

TransifexはFedoraのi18nリーダであったDimitris Glezos氏が設立したTransifex Inc.が提供しているクラウドおよびWebベースの翻訳支援ツールであり，翻訳メモリが利用できる．一般的な翻訳支援ツールは書籍あるいは論文など文章の翻訳を対象としているが，Transifexはその出自から単語，数式，比較的短い文章が混在するソフトウェアの翻訳を得意としている．Transifexにおいて翻訳を行う際の画面例を図1に示す．

3.2 共通翻訳メモリ

翻訳メモリはThe Localisation Industry Standards AssociationにてTMX1.4b Specification[6]として標準化されており，2005年4月に定められた版が最新である．図2に示すように翻訳メモリはXML形式のテキストファイルである．日本語に翻訳する場合，翻訳元が英語，翻訳先が日本語になるが，それらは<tuv>タグにて指定され，xml:langの値で識別される．英語ソースおよびそれを翻

図1　Transifex 画面事例

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE tmx SYSTEM "http://www.lisa.org/tmx/tmx14.dtd">
<tmx version="1.4">
  <header adminlang="en" (中略) o-tmf="Transifex.com"/>
  <body>
    <tu>
      <tuv xml:lang="en">
        <seg>Not a member of the designated site</seg>
      </tuv>
      <tuv xml:lang="ja">
        <seg> 指定されたサイトのメンバではありません </seg>
      </tuv>
    </tu>
    <tu>
      <tuv xml:lang="en">
        <seg>Select new Category for Page &amp;quot;{0}&amp;quot;</seg>
      </tuv>
      <tuv xml:lang="ja">
        <seg> ページ &amp;quot;{0}&amp;quot; に新しいカテゴリを選択 </seg>
      </tuv>
    </tu>

(以下，<tu> ～　</tu> の組が続く)

  </body>
</tmx>
```

図 2　翻訳メモリ事例

訳した日本語で一組の翻訳セットになるが，それらは <tu> タグで囲まれる．翻訳メモリはこのように文字列単位の辞書と考えられる．翻訳メモリを利用できる翻訳支援システムでは，翻訳する文字列に類似した翻訳事例を提示してくれるため，一貫した翻訳が実現される．多くの翻訳支援システムの例にもれず Transifex は翻訳時に翻訳メモリを生成するので，翻訳後にそれをダウンロードすることによって次期バージョンあるいは別システムの翻訳に利用することができる．本プロジェクトではこうして生成される Sakai, Moodle, Mahara の翻訳メモリを結合して大学教育向共通翻訳メモリを開発した．

3.3　Web アプリケーション翻訳の課題

1990 年代から翻訳メモリは文書の翻訳に利用されていた．しかしながら文書の翻訳と Web アプリケーションである OSS の翻訳には 2 つの大きな違いがある．

まず，文書の場合は Word 文書などが準備されるが Web アプリケーションの場合には，実装により翻訳対象のファイル形式が異なっている．本プロジェクトで対象とした Sakai は Java で標準的に用いられているリソースバンドルファイル（.properties ファイル）による方式であるが，Moodle ／ Mahara では PHP プログラム中に埋め込まれた形式となっている．

2 つ目の違いは，利用される文字列の長さが挙げられる．インタラクティブな Web アプリケーションの場合，氏名（name），電話番号（Mobile）など英語で 1, 2 ワードの翻訳単位が多くなっている．また，特に 1 ワードの場合，コンテキストによって翻訳が異なることがあり，システムを通じて 1 つの単語が一意に訳されることはまれである．翻訳メモリにはその文字列がプログラム中で出現する位置情報はなく，コンテキスト依存の翻訳を管理することができない．

この 2 つの課題を解決するために，多くの翻訳システムが標準的にサポートしている gettext 形式の PO ファイルの利用を考えた．まず，後述するように PO ファイルはコンテキスト情報を保持することができる．また，翻訳対象の文字列が含まれる .properties ファイルおよび PHP ファイルについては Python スクリプトで PO ファイルにコンバージョンできることを確認した．これにより OSS の実装により翻訳対象が異なったファイル形式であっても，同一の PO ファイル形式でソースを取り扱うことができる．

こうしたフィージビリティ・スタディを踏まえ，本プロジェクトでは次章に示すような翻訳手順を考案した．

4.　新たな翻訳プロセス

新たに考案した翻訳手順を図 3 に示す．図中の開発範囲と示した領域を除けば，一般的な翻訳手順となる．ここでは簡単に図 3 の説明を行う．図中の①，②…は下記の番号に対応する．

① Moodle, Mahara, Sakai のソースファイルを準備する．

② PHP あるいは .properties（図中，.prop.）ファイル形式のソースファイルを新たに開発した src2po translator にて PO ファイルに変換する．

③新たに開発した po updater を用い，以前のバージョンで生成した前版 PO ファイルがあればそれを利用して，ソース文字列が一致した文字列に日本語訳を設定する．

④ Transifex が用意するクライアントモジュールを使って，更新された PO ファイルを Transifex にアップロードする．

⑤以前のバージョンにて得られた Moodle, Mahara, Sakai の翻訳メモリを結合した前版翻訳メモリ（図中，前版 TMX）を用いて翻訳する．翻訳が完了した時点で, 更新された最新の共通翻訳メモリ（図中，共通 TMX）が得られる．

⑥ Transifex が用意するクライアントモジュールを使って翻訳済 PO ファイルを Transifex からダウンロードする．

⑦翻訳済 PO ファイルを新たに開発した po2src transla-

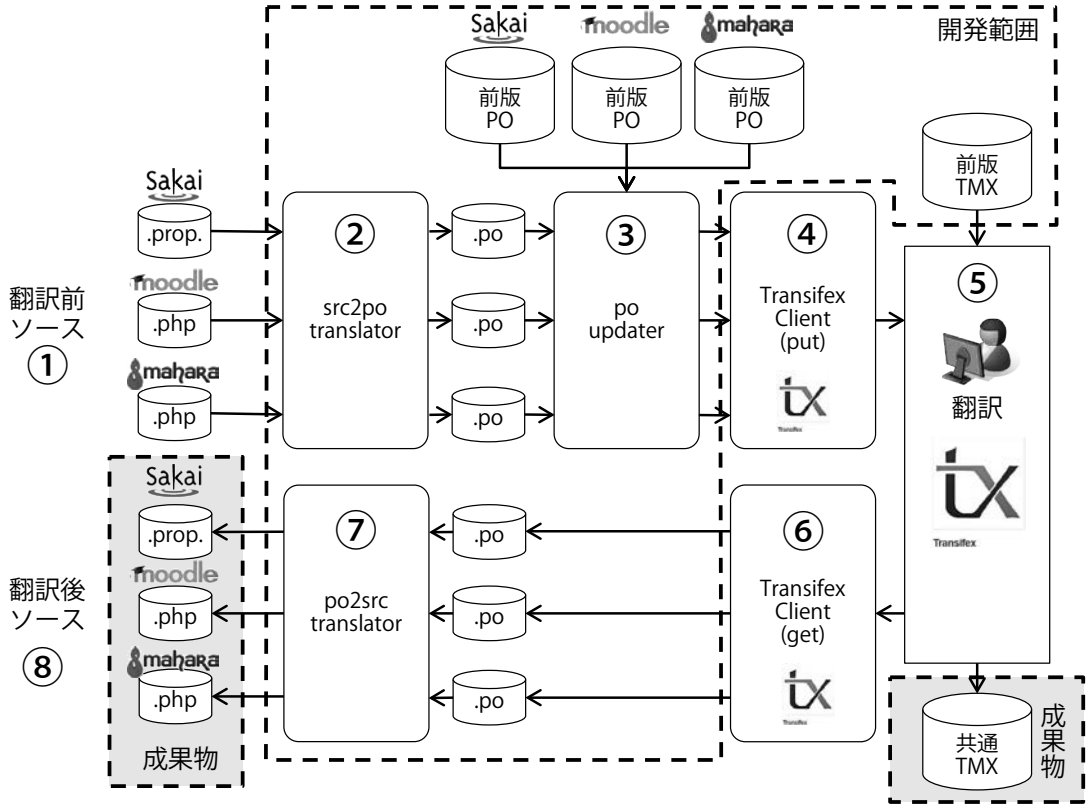

図 3　考案した翻訳プロセス

tor にて PHP あるいは .properties ファイル形式のソースファイルに変換する．

上述したプロセスにて Sakai, Moodle, Mahara を Transifex にて翻訳し，共通翻訳メモリ作成に必要となるそれぞれの翻訳メモリを生成した．

5. 翻訳プロセスの実装

5.1 Sakai

5.1.1 Sakai における i18n

Sakai は Java で実装されたオープンソースの e ラーニングシステムである．Java で記述されたソフトウェアの i18n においては，java.util.ResourceBundle クラスを利用するのが最も簡単かつポピュラーな方法であり，Sakai でもこの方法が用いられている．以降，Sakai で用いられている i18n と l10n の仕組みを簡単に紹介する．

Java には，プログラムの設定等をプロパティファイル（.properties）として外部に記述し，このファイルを書き換えることでプログラムの振る舞いを変更する機能がある．java.util.ResourceBundle を用いることにより，このプロパティファイルをロケールに応じて変更し，利用する言語に応じてソフトウェアの見た目や振る舞いを変更する i18n を実現することができる．もし，プログラム中のプロパティファイル“chat.properties”を日本語（ja ロケール）に対応させる場合は，“chat_ja.properties”というファイルを作成し，“chat_ja.properties”の中の必要な個所を翻訳すればよい．この枠組みを利用することにより，各種言語へと対応させる l10n をソフトウェア本体の開発から分離することが可能となる．

ここで，Java で用いられるプロパティファイルは，「hello.world=Hello World!」のようにキー（hello.world）と値（Hello World!）が“=”で区切られたテキストファイルである．java.util.ResourceBundle で扱うことができるプロパティファイルの文字コードは ISO-8859-1 のみであり，他の文字コード（たとえば日本語や中国語など）を利用する場合は，Unicode エスケープシーケンス（¥udddd 形式，d は 16 進の数値）に変換する必要がある．たとえば「hello.world= こんにちは世界!」は「hello.world=¥u3053¥u3093¥u306b¥u3061¥u306f¥u4e16¥u754c!」と記述する必要がある．このように，Unicode エスケープシーケンスで書かれたプロパティファイルは，一見して何が書かれているかをすぐに理解することが難しい．そのため，プロパティファイルを直接編集して翻訳することは困難である．このような問題を解決するために，native2ascii コマンドや ResourceBundle Editor [7]，Benten

[2] 等の翻訳支援ツールが開発されている．Sakaiを各種言語へ翻訳するl10nの作業においても，これらのツールが利用されている．しかしながら，これらのツールは特定の翻訳者が一人で作業を行うためのものである．そのため，第2章で述べたような問題が潜在的に存在し，新しい翻訳システムに対する期待が高まっている．

上述のように，Sakaiではjava.util.ResourceBundleと各言語用のプロパティファイルを用いてi18nとl10nが行われている．この仕組みを変えずにTransifexによるコミュニティ翻訳を導入するためには，Sakaiのプロパティファイル群とTransifexの間を橋渡しするプログラムの開発が必要である．これを実現するために，gettext形式のPOファイルを介して両者を接続するsrc2po translatorとpo2src translatorを開発した．src2po translatorとpo2src translatorによる橋渡しの流れを図4に示す．以降の5.1.2節では，Sakaiのプロパティファイル群から翻訳用ソースとなるPOファイルを生成してTransifexへアップロードする仕組みを説明する．次に5.1.3節では，Transifexから翻訳用ソースをダウンロードしてSakaiのプロパティファイル群へ書き戻す仕組みについて述べる．

翻訳対象のファイルを
PO ファイルに変換して
コンテキスト情報を保持する

5.1.2　翻訳用ソース作成

Sakaiでは，announcement，chat，gradebook，といったツールごとにモジュール化がなされており，開発もモジュール単位で独立に進められている．そのため，ツール（モジュール）内での用語の使い方には一貫性があるものの，モジュール間では一貫性が保たれていない場合が存在する．そのため，Sakai全体を翻訳単位として翻訳用ソースを作成した場合は，実際の挙動と翻訳対象文字列を比較しながら翻訳作業を行うことが困難になると予想される．一方，各モジュールは複数のJavaソースファイルから構成されており，翻訳対象となるプロパティファイルもモジュールごとに多数存在する．そのため，一つひとつのプロパティファイルを翻訳単位として翻訳用ソースを作成すると，モジュール内での翻訳の一貫性を保つことが困難となる．これらのことから，Sakaiのモジュールを単位として翻訳用ソースを作成することが適切だと判断した．

Sakaiのモジュール単位で翻訳用ソースを作成するためには，それぞれのモジュールに含まれるプロパティファイルから翻訳対象を抽出し，１つの翻訳用ソースとし

```
1.  #. java.acccess
2.  #: access.properties:1
3.  msgctxt "access.properties:java.acccess"
4.  msgid "Access:"
5.  msgstr " アクセス :"
```

図 5　PO ファイルのデータ表現（左端は行番号）

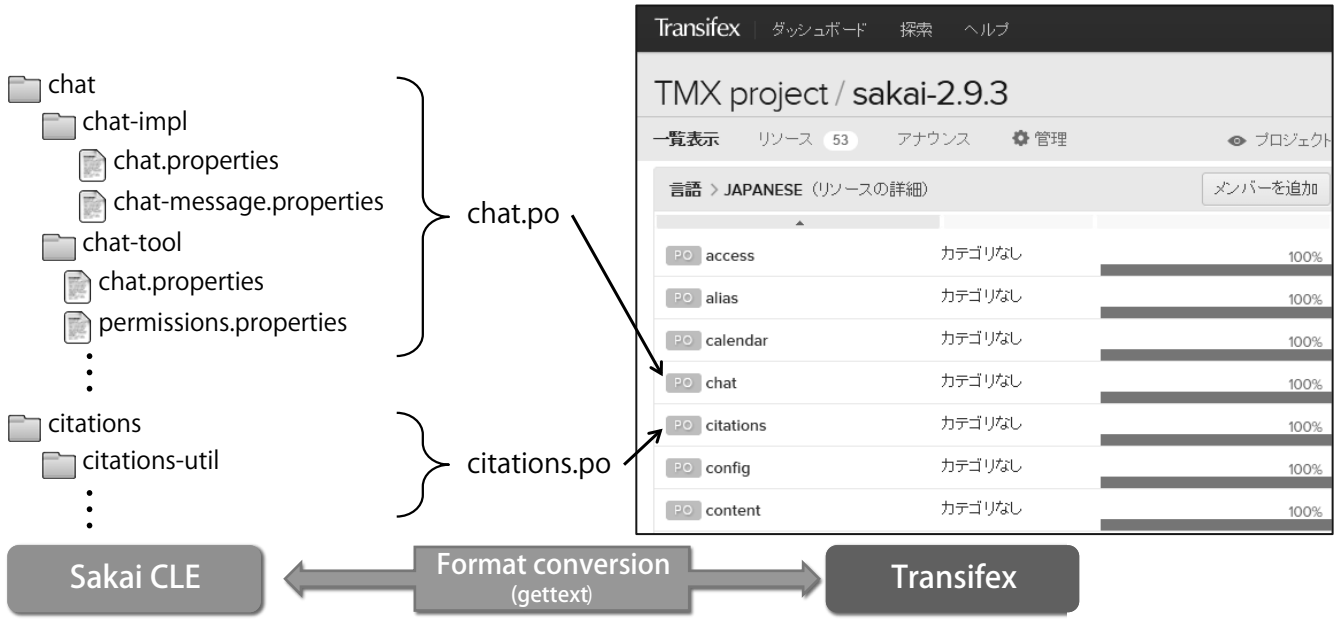

図 4　Sakai モジュール単位で Transifex と同期する仕組み

て整形する機能が必要である．さらに翻訳用ソースは，元のJavaプロパティファイルへの変換に必要な情報を保持している必要がある．これらの条件を満たすために，世界的に広く利用されているgettext形式のPOファイルを採用し，src2po translatorの開発を行った．図5にPOファイルを用いた翻訳対象の記述例を示す．1行目はプロパティファイルに含まれる翻訳対象のキーを表しており，“#.”で始まるコメント欄に情報を埋め込んでいる．2行目はこのキーが含まれるプロパティファイル名であり，3行目のmsgctxtで始まる行は翻訳対象を一意に定めるためのコンテキストである．Sakaiでは，プロパティファイルが異なると同じキーを違う意味で使う可能性があり，キーのみでは翻訳対象を一意に定めることができない．よって，プロパティファイル名とキーの組合せをコンテキストとすることで，翻訳対象が一意に定まらないという問題を回避している．最後に，4行目はキーに結び付けられた翻訳対象文字列であり，5行目は各種言語（図5では日本語）への翻訳結果を表している．

このように，モジュール内に存在する複数のプロパティファイルから抽出した翻訳対象を1つのPOファイルで表すことが可能となる．このようなプロパティファイルからPOファイルへ変換するsrc2po translatorは，Pythonのライブラリとして開発が進められているjprops [8]（Javaプロパティファイルの入出力）とpolib [9]（POファイルの入出力）を組み合わせることで実現している．

最後に，生成された翻訳用ソースをTransifex Clientツール [10] を用いてTransifexにアップロードする仕組みを構築している．

5.1.3　翻訳済ソースの反映

5.1.2節の処理により，翻訳対象となるソースはすべてTransifexにアップロードされ，Transifex上で複数の翻訳者が協力しながら同時に翻訳作業を進めることが可能となる．ここで得られた翻訳結果をSakaiに反映するためには，Transifexから翻訳結果をダウンロードし，Sakaiのプロパティファイルに書き戻すpo2src translatorが必要となる．翻訳結果のダウンロードには，5.1.2節で紹介したTransifex Clientツールが利用でき，翻訳結果が反映されたPOファイル形式でダウンロードすることが可能である．次に，polibを利用して翻訳結果と書き戻すべき翻訳対象の情報を抽出し，jpropsの機能を用いて元のプロパティファイルへと書き戻すことでpo2src translatorを実現している．

5.1.4　翻訳済ソースの再利用

5.1.3節で得られたPOファイルは次期バージョンの翻訳で再利用できる．図3において③で示されるpo updaterではsrc2po translaterにて生成されたPOファイルと前バージョンの翻訳時に得られた前版POファイルを比較して，同一のソース文字列がある場合にそれらをPOファイルのmsgstrに埋め込む．この処理をPO前処理と呼ぶことにする．このようにして同一ソース文字列でもコンテキストにより異なる翻訳個所について，翻訳メモリでは実現できないコンテキスト依存の翻訳を可能としている．

5.2　Moodle ／ Mahara

5.2.1　Moodle ／ Maharaにおけるi18n

Moodleはオープンソースのeラーニングプラットフォームであり，Maharaはオープンソースのｅポートフォリオシステムである．開発コミュニティは違っているもののどちらもPHPで実装されているため，本節ではまとめて紹介する．PHPのi18nではgettext関数を利用することができるが，Moodle ／ Maharaでは文字列型変数の連想配列による方法が用いられている．以降Moodle ／ Maharaで用いられているi18nとl10nの仕組みを簡単に紹介する．

Moodle ／ Maharaでは，言語パック（Language packs）を導入することで各種言語に対応している．たとえば，Moodleに導入した日本語（jaロケール）の言語パックは，Moodleの設定ファイルに設定されているファイルシステム上のデータディレクトリ直下に存在する“lang”ディレクトリに“ja”というロケールを表した名称のディレクトリが作成され，“ja”以下に翻訳PHPファイルが集められて配置されている．Moodleで日本語を指定するとホームディレクトリ以下の言語PHPファイルの代わりに言語パックの“ja”以下の対応する翻訳PHPファイルが読み込まれることで言語が切り替わる仕組みとなっている．Maharaの場合は，データディレクトリ直下に“languagepacks”ディレクトリがあり，言語パックを導入すると“ja.utf8”が作成される．“ja.utf8”以下には，Maharaのインストールディレクトリ以下に配置されている言語PHPファイルのディレクトリ構造と同じ構造で翻訳PHPファイルが配置されているという点がMoodleと異なるが，言語が切り替わる仕組みそのものはMaharaもMoodleと同様である．

Moodle ／ Maharaで用いられている言語PHPファイルは「$string[‘hello_world’] = ‘Hello World!’;」のように$stringという１つの文字列型変数に対して，キー（hello_world）と値（Hello World!）という連想配列で表される．

また，翻訳PHPファイルの文字コードはUTF-8であるので，値には翻訳するロケール言語を直接記述することができる．

上述のように，Moodle／Maharaでは各言語パックの翻訳PHPファイルを文字列型変数$stringの連想配列として記述することでi18nとl10nが行われている．この仕組みを変えずにTransifexによるコミュニティ翻訳を導入するためには，前節のSakaiと同様，Moodle／Maharaの言語PHPファイル群とTransifexの間を橋渡しするプログラムの開発が必要である．これを実現するために，第5.1節で述べたsrc2po translatorとpo2src translatorを流用し，gettext形式のPOファイルを介して両者を接続することを行った．以降の5.2.2節では，Moodle／MaharaのPHP翻訳ファイル群から翻訳用ソースとなるPOファイルを生成してTransifexへアップロードする仕組みを説明する．次に5.2.3節では，Transifexから翻訳用ソースをダウンロードしてMoodle／Maharaの翻訳PHPファイル群へ書き戻す仕組みについて述べる．

5.2.2　翻訳用ソース作成

Moodleの言語パックには，翻訳言語PHPファイルが“ja”ディレクトリ直下の一階層に集められている．もし直接エディタ等で翻訳する際には，翻訳対象のファイルがまとめられているためディレクトリ移動の必要がないという利点がある一方，gettext形式で翻訳するためにすべての翻訳PHPファイルを１つのPOファイルにまとめた場合，翻訳時にどの機能について翻訳しているのか把握できず，コンテキストに依存する翻訳が困難となることが予想される．このことから，Sakaiでモジュール単位にPOファイルを生成するのと同様，Moodleも機能単位で翻訳用ソースを作成することが適切であると判断した．Moodleのインストールディレクトリ以下は“auth”や“blocks”ディレクトリ等の機能ごとに分かれており，それらの階層中の“lang/en”ディレクトリに存在している言語PHPファイルが“ja”の翻訳PHPファイルに対応している．よって，Moodleの機能単位に翻訳用ソースを作成するためには，機能ごとのディレクトリにある“lang/en”の言語PHPファイルに対応する言語パックの翻訳PHPファイルを翻訳対象として抽出し，１つの翻訳用ソースとして整形する機能が必要である（図6）．さらに翻訳用ソースは，元の翻訳PHPファイルへの変換に必要な情報を保持している必要がある．これらの条件を満たすよう5.1.2節で述べたgettext形式のPOファイルを用いたsrc2po translatorを流用し，Sakaiのモジュールに含まれるプロパティファイルを走査するロジックをMoodleの機能ごとのディレクトリ内に含まれる言語PHPファイルを走査し，対応する翻訳PHPファイルを翻訳用ソースに含めるよう改修している．POファイルの翻訳対象の記述方法についてはSakaiで作成するPOファイルにならっており，生成した翻訳用ソースはTransifex Clientツールを用いてTransifexにアップロードしている．Maharaの場合，5.2.1節で述べたようにディレクトリ名称と言語パックの翻訳PHPファイルの配置構造はMoodleと少し異なるが，機能単位にPOファイルを生成し，Transifexにアップロードする仕組みそのものはMaharaもMoodleと同様である．

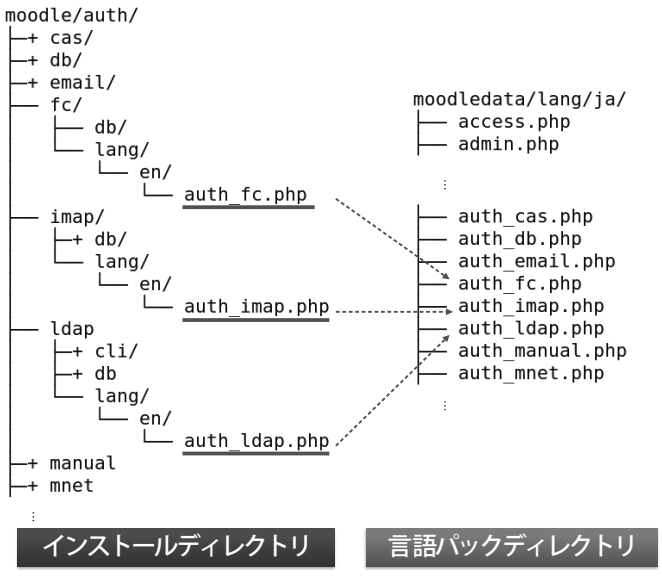

図6　Moodleにおける機能別POファイル生成

5.2.3　翻訳済ソースの反映

5.2.2節の処理により，翻訳対象となるソースはPOファイルとしてすべてTransifexにアップロードされ，Transifex上で複数の翻訳者が協力しながら同時に翻訳作業を進めることが可能となる．ここで得られた翻訳結果をMoodle／Maharaに反映するためには，Transifexから翻訳結果をダウンロードし，Moodle／Maharaの翻訳PHPファイルに書き戻すことが必要である．ここでも5.1.3節で述べたpo2src translatorを流用し，polibを利用して翻訳結果と書き戻すべき翻訳対象の情報を抽出した後，Moodleの場合は言語パックの“ja”ディレクトリに，Maharaの場合は言語パックの“ja.utf8”ディレクトリ以下のそれぞれの翻訳PHPファイルに書き戻すことで実現している．ただし，実際に運用する際には，新しく言語パックが更新された際に翻訳結果が 上書きされることがないようローカル言語（独自翻訳）として以下に保存する方がよいと思われる．ローカル言語の翻訳は，言語パックより優先して採用される．

Moodle:[データディレクトリ]/lang/ja_local/
Mahara:[ホームディレクトリ]/local/lang/ja.utf8/

Sakai, Moodle, Mahara の翻訳メモリから共通翻訳メモリを作成する

6. 共通翻訳メモリ作成と検証

6.1 共通翻訳メモリおよび PO ファイル作成

Sakai, Moodle, Maharaの翻訳から得られたそれぞれの翻訳メモリを結合し，本プロジェクトで提案する共通翻訳メモリを作成する．次に，この共通翻訳メモリと5.1.4節で述べたPO前処理について，Sakaiの最新版であるSakai10.2（以下，Sakai10）の翻訳を通じて効果を検証する．

まず，Sakai10の前バージョンであるSakai-2.9.3, および当時最新バージョンであったMoodle-2.6.1，Mahara-1.8.1をTransifexにて翻訳し，それぞれの翻訳メモリを得た．次に，これらの翻訳メモリを結合して共通翻訳メモリを作成するが，異なるシステムの翻訳メモリを結合する際には，翻訳メモリの「汚れ」[11]と言われる不整合が生じる．そのため下記に示す浄化を行って共通翻訳メモリを作成した．

- 同一ソースが異訳された翻訳の統合
- 大文字小文字を識別するCase Sensitiveにより同一の単語に対して複数の翻訳がある場合の統合
- #，*，{0}などの数学記号だけで構成されたソースになっている翻訳の削除
- Go，or，atなどソースが2文字以下の単語の翻訳の削除
- 句読点（。、）とカンマ，ピリオドの混在を統一

作成された翻訳メモリのサイズを見ると，Sakai単体の翻訳メモリにおいて<tu>で設定される英日翻訳セットは12,377組であるが，Sakai, Moodle, Maharaの翻訳メモリから生成した共通翻訳メモリでは約3倍の33,554組の翻訳セットを含んでいる．また，同時にSakai-2.9.3の翻訳により生成されたPOファイルを用意した．これらの翻訳メモリおよびPOファイルは図3において前版TMXおよびSakai用の前版POファイルに対応する．

6.2 検証方法

新たな翻訳方法の効果を確認するために次の4ケースの方法でSakai10の翻訳環境の設定を行った．評価基準として，翻訳環境設定時に翻訳メモリおよびPO前処理で，翻訳対象の文字列をシステムが自動的に翻訳済にする割合とした．なお，翻訳対象であるSakai10の文字列数は17,484である．

(1) Case 1
Sakai翻訳で得られた翻訳メモリを使う．ただし，PO前処理は行わない．

(2) Case 2
Sakai, Moodle, Mahara翻訳で得られた翻訳メモリを結合した共通翻訳メモリを使う．ただし．PO前処理は行わない．

(3) Case 3
翻訳メモリは使わず，PO前処理のみを行う．

(4) Case 4
PO前処理を行い，共通翻訳メモリを使う．

表 1　翻訳メモリおよび PO 前処理による翻訳率

	翻訳メモリ	PO 前処理	未翻訳文字列数	翻訳率 (%)
Case 1	Sakai 翻訳メモリ	なし	3,603	79
Case 2	共通翻訳メモリ	なし	4,047	76
Case 3	なし	あり	2,601	85
Case 4	共通翻訳メモリ	あり	2,426	86

6.3 結果

翻訳メモリおよびPO前処理の有無の違いによる翻訳率の変化を表1に示す．

Case 1とCase 2により共通翻訳メモリによる翻訳率の向上を比較することができる．翻訳率の向上が期待されるCase 2では，むしろ未翻訳の文字列が444増加し，翻訳率も3%減となった．Case 3ではPO前処理による翻訳率の向上を評価することができる．共通翻訳メモリを用いたCase 2の翻訳率と比較すると9%の向上が見られた．またCase 4では共通翻訳メモリとPO前処理の両方の効果を評価することができ，試みたケースの中では最も高い86%の翻訳率を得た．

Case2で翻訳率が低下した原因は，主として共通翻訳メモリを生成する際の浄化によってCase Sensitiveの用語を整理したことに起因する．その具体的な事例を以下に述べる．翻訳メモリはXMLの特性を継承しているためにCase Sensitiveであり，“Add”と“add”は異なる単語とみなす．しかし，“Add”および“add”は共に“追加”と翻訳されて登録されているので，翻訳メモリでは“Add”の翻訳セットを削除している．そのため翻訳

対象として“Add”が出現した場合，自動的に翻訳済とならず未翻訳になってしまう．こうした事例は，“yes”, “save”, “remove”, “done”, “none”などでも発生しており，結果として翻訳率が低下する原因となった．翻訳支援ツールの中にはこうしたCase Sensitiveを無視するオプションを有するツールもあるが，今回使用したTransifexはCase Sensitiveの有無を設定することはできない．

7. 考察

ここでは共通翻訳メモリの意義について考察する．共通翻訳メモリは，用語の選択に客観性を持たせるためにカタカナ新語辞典[12]および大学用語集[13]などを参照して整備および作成しており，標準的な用語を用いた一貫した翻訳を実現するために意義あるものと考える．しかしながら，第6章で明らかになったように，共通翻訳メモリはそこに含まれる翻訳セット数が単独OSSの翻訳メモリに比較して約3倍になっているにもかかわらず翻訳率向上に寄与せず，むしろ共通化する際の浄化の影響によって翻訳率を低下させることになった．

そこで適用が期待できるケースを考えると，OSSのバージョンアップによって新機能が追加され新たな翻訳が必要になるケースが挙げられよう．このケースではPO前処理と共通翻訳メモリの翻訳を組み合わせ，それぞれの長所を活かした翻訳プロセスとなる．まずPO前処理を行い前バージョンまでに完了しているコンテキスト依存の翻訳を適用した後，未翻訳の個所を共通翻訳メモリを使って翻訳していく．Transifexなどの翻訳支援システムでは100%マッチでなくても翻訳する文字列に近い文字列が翻訳メモリにあれば適合率とともに翻訳候補を提示してくれる．共通翻訳メモリは単体のシステムの翻訳から生成される翻訳メモリに比べ多くの翻訳セットを含んでいるので提示される候補の増加が期待できる．

また，より大きな意義を持つのは，新たに開発されたOSSを初めて翻訳するケースあるいは既存のOSSをスクラッチから改めて翻訳し直すケースであろう．一般的にOSSの翻訳では翻訳メモリが利用できない初回の翻訳が最も時間がかかると言われている．また，いったん翻訳をしてしまった場合には用語の変更は困難である．したがって共通翻訳メモリの最善の適用は，そのOSSを初めて翻訳するケースであると考える．初回の場合は前版POファイルがないのでPO前処理は行えないが，本プロジェクトで作成した共通翻訳メモリを図3における前版TMXとして利用することができる．

共通翻訳メモリと
POファイルで
翻訳の環境を構築する

8. おわりに

大学教育用OSSのl10nとして必須となる翻訳に注目し，大学教育に供されるすべてのOSSにおいて一貫性のある翻訳を目指したプロジェクトについて報告した．

まず共通翻訳メモリについては，Sakai, Moodle, Maharaの翻訳で得られた翻訳メモリを結合し，重複した翻訳，数学記号などの意味を持たない翻訳などを浄化してそれを作成することができた．

また，翻訳メモリでは考慮されないコンテキスト依存性については，gettext形式のPOファイルにコンテキスト情報を持たせ，翻訳支援システムにアップロードするためのPOファイルを生成する際に前版で得られたコンテキスト依存の翻訳を反映することで，共通翻訳メモリだけでは達成しえない環境設定時の翻訳済の割合を向上させることができた．これらを適用したSakai10の翻訳事例では，設定時に86%が翻訳済となり，新たに翻訳する文字列は2,400程度となった．

本報告では新たに考案した翻訳方法について述べたが，他のOSS翻訳で得られた翻訳メモリの追加，共通翻訳メモリの保守，翻訳されたシステムにおける違和感の解消，同一文字列が異訳されている場合のコミュニティ間の調整，本方式の国際コミュニティへの展開など今後実施すべきことは多い．

Sakaiを開発しているApereoコミュニティでは，すでにSakai Spainチームと日本Sakaiコミュニティが共同で行う翻訳プロジェクトがスタートしており，そこでは無償のTransifexも提供されている．

今後もさまざまなOSSおよび商用のソフトウェアが大学教育向に開発されることになろうが，本プロジェクトの成果により，ユーザにとってより使いやすい環境が提供されていることを願っている．なお，本プロジェクトの成果はWebサイト[14]にて公開している．

謝辞　本研究はJSPS科研費25280127の助成を受けたものです．

参考文献

1）メディア教育開発センター：e ラーニング等の ICT を活用した教育に関する調査報告書（2008 年度），(2009)．
2）Benten，http://sourceforge.jp/projects/benten/
3）OmegaT，http://www.omegat.org/ja/omegat.html
4）Crowdin，https://crowdin.com/
5）Transifex，https://www.transifex.com/
6）TMX 1.4b Specification，http://www.gala-global.org/oscarStandards/tmx/
7）ResourceBundle Editor，http://essiembre.github.io/eclipse-rbe/
8）Jprops，http://mgood.github.io/jprops/
9）Polib，http://polib.readthedocs.org/
10）Transifex Client，http://docs.transifex.com/developer/client/
11）西野竜太郎：アプリケーションをつくる英語 −エンジニアよ，世界市場を狙え！−，インプレスジャパン (2012).
12）カタカナ新語辞典，三省堂 (2014).
13）大学行政管理学会学事研究会編：大学用語集，学校経理研究会 (2010).
14）TMX プロジェクト，http://www.sakaiproject.jp/tmx/

常盤 祐司（正会員）yuji.tokiwa.dc@hosei.ac.jp
1978 年慶應義塾大学工学部修士課程修了．同年石川島播磨重工業（株）入社．1984 年日本アイ・ビー・エム（株）入社．2005 年より，法政大学　情報メディア教育研究センター教授．大学における教育・研究・事務・経営システムの構築に従事．教育システム情報学会，IEEE，ACM 各会員．

出口 大輔（非会員）ddeguchi@nagoya-u.jp
2001 年名大・工・情報卒．2006 同大大学院情報科学研究科博士後期課程修了．博士（情報科学）．2004 年〜 2006 年まで日本学術振興会特別研究員．2006 年より名大大学院工学研究科研究員，2008 年より同大学院情報科学研究科助教，2012 年より同大情報連携統括本部情報戦略室准教授．主に画像処理・パターン認識技術の開発とその ITS および医用応用に関する研究に従事．電子情報通信学会，IEEE 各会員．

宮崎 誠（正会員）m.miyazaki@kio.ac.jp
熊本大学大学院自然科学研究科電気システム専攻修士課程修了．同大学院社会文化科学研究科教授システム学専攻博士後期課程在学．工学修士．同大学院社会文化科学研究科教授システム学専攻特定事業研究員（2008 年），同大大学教育機能開発総合研究センター特定事業研究員（2010 年），法政大学情報メディア教育研究センター助手（2011 年）を経て，2014 年より畿央大学教育学習基盤センター助教．

平岡 斉士（非会員）hiraoka.naoshi@gmail.com
2007 年京都大学大学院教育学研究科博士課程修了，博士（教育学）．2007 年より同大学学術情報メディアセンター／情報環境機構・助教．2013 年より同大学学際融合教育研究推進センター・特定准教授．2014 年 10 月より熊本大学大学院社会文化科学研究科教授システム学専攻准教授．大学教育における e ポートフォリオ設計・活用法の研究に従事．教育システム情報学会，日本教育工学会，日本心理学会，日本認知心理学会各会員．

喜多 敏博（正会員）kita@ield.kumamoto-u.ac.jp
1967 年に奈良に生まれる．京都大学大学院工学研究科博士後期課程研究指導認定退学，熊本大学工学部助手，総合情報基盤センター准教授，e ラーニング推進機構教授，現在に至る．工学博士（名古屋大学，2005 年）．e ラーニングシステム，非線形システム，電子音楽に興味を持つ．

梶田 将司（正会員）kajita.shoji.5z@kyoto-u.ac.jp
1995 年名古屋大学大学院工学研究科情報工学専攻博士課程満了，博士（工学）．同准教授を経て 2011 年より京都大学情報環境機構 IT 企画室教授．「人や社会と調和する情報環境」の実現を目指した情報通信基盤技術や情報サービスに関する研究・教育に従事 . 情報処理学会，電子情報通信学会，日本音響学会，日本教育工学会，教育システム情報学会，日本高等教育学会，IEEE，ACM 各会員 .

採録決定：2015 年 1 月 6 日
編集担当：海老原吉晶（オムロン（株））