Vol.2009-CE-101 No.7
2009/10/10

プログラミングによる国際交流のための翻訳サーバの設計と実装

長　慎也[†1]　兼宗　進[†2]
青木浩幸[†3]　李　元撰[†3]

母国語を用いてプログラムが書けるプログラミング言語において，学生間でプログラムを交換することによる国際交流を実現するシステムを提案する．本システムでは，「ドリトル」のプログラムを，他国語によるプログラムに翻訳することができる．本システムを用いて，韓国の大学にて日本語のプログラムを翻訳し，読解させた結果を報告する．

Design and Implementaion of Translation Server for International Collaboration by Programming

Cho Shinya,[†1] Kanamune Susumu,[†2] Aoki Hiroyuki[†3]
and Lee Won-gyu[†3]

We developed programming language “Dolittle” in which students can write their programs using words originated from their native language. Since Dolittle are used by students of various countries, we thought they can collaborate with students of other country by programming. We have developed the program translation server in which translate students’ program written in their native language into foreign language. This paper reports the design and implementation of the translation server and the result of the experiment in which Korean student read translated programs written by Japanese.

1. はじめに

本発表では，プログラミングを通じた国際交流活動とそのためのシステムを提案する．プログラミングの考え方（アルゴリズム）は，特定の国の文化に依るものではないので，国際交流の題材としてふさわしいものと考えられる．具体的には，プログラミングを通して，次のような活動が可能であると考えた．

- 他国の生徒が作ったプログラムを実行し，鑑賞したり感想を交わしたりする．
- 他国の生徒が作ったプログラムを読み，プログラミングに関する学習を国際的に行う．
- 互いにプログラムを修正することで，共同プログラミングを行う．

しかしながら，アルゴリズムの表現方法，つまりプログラミングに用いる言語（「プログラミング言語」のことではなく，プログラム中に用いる識別子などの単語を記述するための自然言語を指す）は，学習者がよく知っている言語，特に母国語を用いるのがよい．

我々はこれまで，プログラミング言語「ドリトル」を用いて教育実践を行ってきた．学習者は自分の母国語を識別子 (変数名，オブジェクト名，メソッド名) として用いることが可能である．識別子に母国語を用いることは，学習者の言語そのものの習得の負担を下げ，言語を用いた本質的なプログラミングに集中して取り組むことができ，その効果が示されている[1)]．ドリトルは，日本語，英語，韓国語に由来する識別子を用いて書くことができる．**図 1** は，ほとんど同じ動作をするプログラムをそれぞれ日本語，英語，韓国語で書いたものである．

しかし，母国語を用いて書かれたプログラムは，他の国の学生には理解することができず，そのままではお互いのプログラミングで理解することは難しい．そこで，ドリトルのプログラムを，違う国の言葉で書かれたプログラムに自動的に変換する翻訳サーバ D-Trans を作成し，その有効性を試す実験授業を行った．

2. プログラム翻訳の指針

本発表における「プログラムの翻訳」とは「プログラム動作自体は変えずに，プログラム

†1 一橋大学
Hitotsubashi University
†2 大阪電気通信大学
Osaka Electro-Communication University
†3 高麗大学
Korea University

```
turtle1＝turtle！create.
「turtle1！100 forward 90 rightturn.」!4 repeat.
greeting＝label！”Hello“create.
greeting！100 200 position.
```

```
거북이1＝거북！만들기.
「거북이1！100 전진 90 우회전.」!4 반복.
인사＝라벨！”안녕하세요“ 만들기.
인사！100 200 위치.
```

```
かめ1＝タートル！作る.
「かめ1！100 歩く90 右回り.」!4 繰り返す.
あいさつ＝ラベル！”こんにちは“作る.
あいさつ！100 200 位置.
```

図 1 ドリトルのプログラム (英語，韓国語，日本語).

内で使用する識別子の由来となる言語を変える」こととする．本発表で提案する方式では，それぞれの識別子を変換のための辞書を準備し，それを利用して置き換えを行うことで翻訳を実現する．

ドリトルには多数の組み込みオブジェクトや組み込みメソッドが存在しており，プログラミングに必要な基本的な機能を提供している．たとえば，タートルグラフィックスを描画可能な「タートル」オブジェクト,「タートル」に対して前に進む動作を起こす「前進」メソッド，また，オブジェクトを生成するための「作る」というメソッドなどがある．これらのオブジェクトやメソッドの名前に用いられる識別子を「組み込み識別子」と呼ぶ．これらの組み込み識別子の名前は，学習者がプログラムを書いている途中で変わることはないので，固定の辞書を用意しておき，それを用いて置換すればよい．この辞書を「組み込み辞書」と呼ぶ．

ところが，学習者が作るプログラムの中には，学習者が自分で作った識別子も含まれる．これらの識別子を「ユーザ識別子」と呼ぶ．ユーザ識別子の例としては，タートルオブジェクトの名前として用いる「カメ太」「カメ吉」などの名称が挙げられる．他国の学習者にプログラムを読んでもらうには，ユーザ識別子も適切に翻訳される必要がある．それには，ユーザ識別子を変換するための辞書である「ユーザ辞書」を用意しなくてはならない．学習者がどのような識別子を使うは事前に用意できないので,「ユーザ辞書」の内容は常に変化する．このため，辞書の内容をいつでも編集できるようにする仕組みが必要となる．

3. システムの設計

3.1 必要な機能

プログラミングによる国際交流においては，次のような活動を想定しており，これを実現する機能がシステムに必要となる．

- **Step0:** (同じ国の学生同士) 学生が自分の作品をアップロードし，他の学生がダウンロードして読み，実行してみる．
- **Step1:** 他国の学生により作られた作品を教師がダウンロードし，ユーザ辞書を作成する．そのユーザ辞書を用いて翻訳されたプログラムをダウンロードし，自国の学生が読み，実行する．
- **Step2:** 他国の学生が書いたプログラムに対するコメントをつける．そのプログラムの作者は，コメントが作者の母国語に翻訳された状態で読むことができる．
- **Step3:** 他国の学生が書き，自国語に翻訳されたプログラムを変更し，それをアップロードする．

3.2 サーバ形態にする理由

本システムにおいては，翻訳の仕組みをドリトルに組み込み，クライアントで翻訳する方式も考えられるが，今回は次の理由から，サーバーで行うことにした．

(1) 将来的に，さらに多くの言語に拡張することが可能とする．
(2) 教員などのユーザーが固有名詞などを辞書に追加して共有できる．
(3) 自国語で書いたクライアント上のプログラムは翻訳によって変更されない．

4. システムの実装

3.1 で述べたような活動を行えるよう，必要な機能を設計した．

D-Trans は，次のような機能をもった Web アプリケーションである．

- プログラムビューア
- プログラムエディタ
- コメント機能
- 翻訳機構
- 辞書エディタ

学生は**プログラムビューア** を用いて，自分や他の人のプログラムをダウンロードし，プログラムを見たり実行したりすることができる．他国の学生が作ったプログラムを見る場合

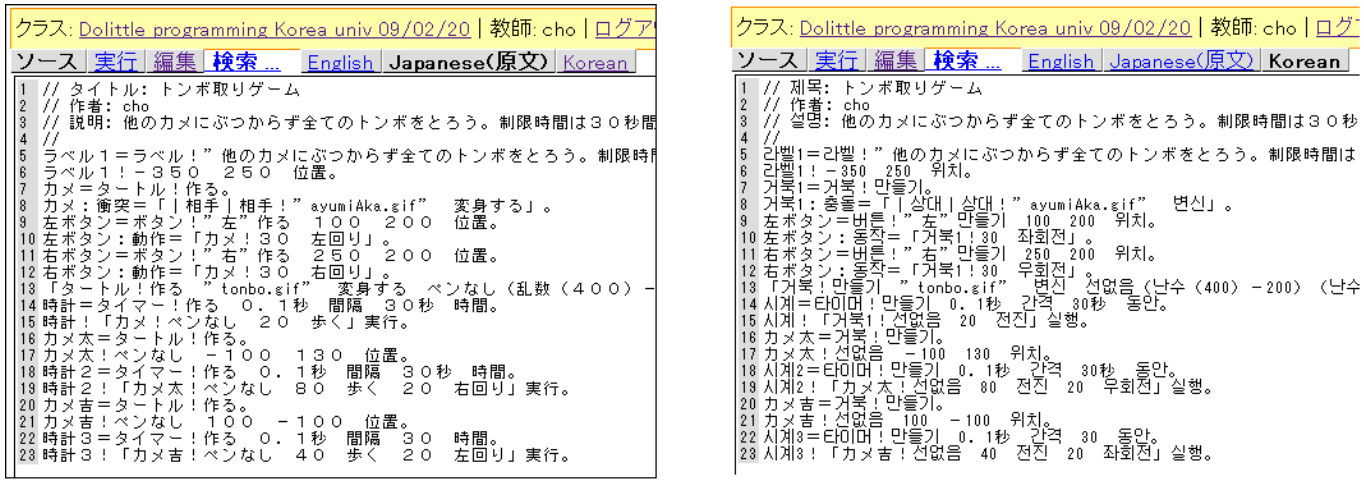

図 3 プログラムビューア（左・原文日本語，右・韓国語訳）

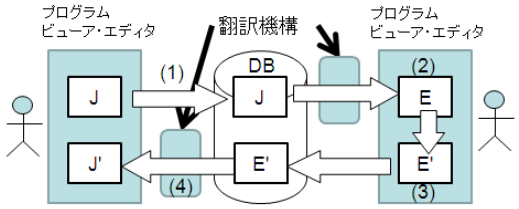

図 2 プログラム閲覧，編集，翻訳機能

は，プログラムの内容が翻訳されて表示される．**図 3** は，プログラムビューアでプログラムを表示させている画面である．

プログラムエディタ を用いて，自分自身や他の学生が作ったプログラムを編集し，アップロードすることができる．他国の学生が作ったプログラムは，編集をする前に自分の国の言葉に翻訳される．たとえば，**図 2** のような活動を行うことができる．

(1) 学生 A(日本の学生) がプログラムをアップロードする
(2) 学生 B (英語圏の学生) がそのプログラムの英訳を読む
(3) 学生 B はそのプログラムの一部を変えてアップロードしなおす
(4) 学生 A はその変更されたプログラムを再び日本語にして読む

コメント機能 を用いて，学生同士でプログラムに関するコメントを読み書きできる．他国の学生がつけたコメントは，読む学生の言葉に翻訳されて表示される．

翻訳機構 は，他国語でプログラムを読んだり編集したりするときに利用される．

翻訳は「組み込み辞書」「ユーザ辞書」の 2 種類の辞書を用いて行う．辞書は「エントリ」から構成され，エントリは「キー」と「値」からなり，キーで示される単語は値で示される単語に翻訳されることを意味する．翻訳の手順は次のようになる (**図 4**)

- ドリトルのプログラムを字句要素ごとに分割し，識別子を取り出す
- それぞれの識別子について次の操作を行う:
  (1) 組み込み辞書を調べ，その識別子をキーとして持つエントリがあれば，そのエントリの値に置換する．
  (2) 1 において該当エントリがなければ，ユーザ辞書について同様の操作を行う
  (3) 2 において該当エントリがなければ，翻訳は失敗し，元の語がそのまま翻訳結果となる

ある識別子の翻訳が失敗した場合，ユーザ辞書にその識別子をキーとしてもつ新しいエントリが作られる．これを未知語エントリと呼ぶ．未知語エントリは空の値をもつ．

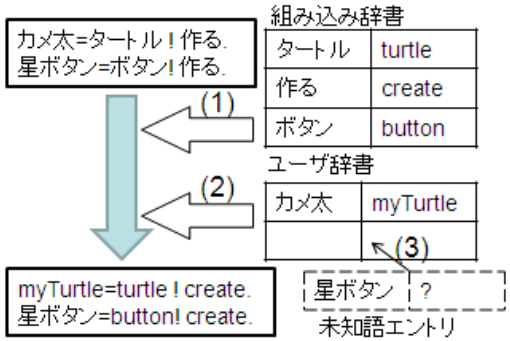

図 4 翻訳の手順

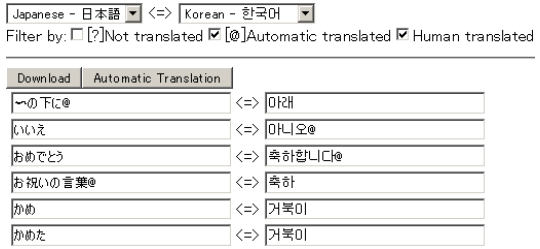

図 5 辞書エディタの画面

未知語エントリは，**辞書エディタ**を使って編集可能である．**図 5** に，辞書エディタの画面を示す．辞書エディタは主に教師が利用し，**図 6** に示すように，未知語エントリの値が空白になっている部分に訳語を追加していく．訳語がわからない場合は自動翻訳を用いることができる．自動翻訳には Google 翻訳 API[2] を利用している．もし自動翻訳の結果に不備があれば，後から修正することも可能である．

5. 日韓での翻訳実験

システムが正しい翻訳を行い，学生が翻訳されたプログラムを理解できるかどうかを検証するための実験を行った．実験では，日本人学生によって書かれたドリトルのプログラムを韓国語訳したものを，韓国人学生に読ませた．

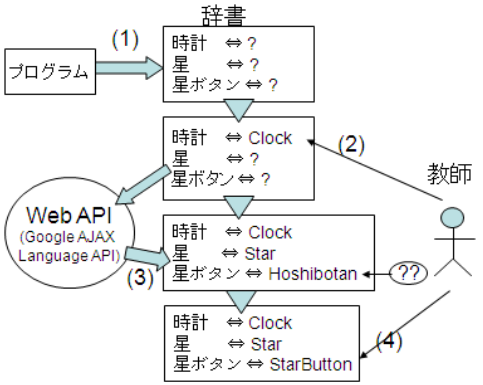

図 6 辞書エディタの利用

- **場所** 高麗大学 (韓国)
- **日時** 2009 年 2 月 20 日 14:00-17:00
- **学生** 高麗大学の学部生 1 年，4 名
- **教師** 高麗大学の博士課程学生で，日本語はわからない

5.1 Step0: 教師による準備

学生に読ませるための「サンプルプログラム」を 14 個用意した．これらはすべて日本語で書かれている．このステップでは，教師がこれらのプログラムをダウンロードし，日本語から韓国語へのユーザ辞書を，辞書エディタを用いて作成した．この教師は日本語はわからないので，自動翻訳を利用した．

翻訳を行った結果，訳語の衝突が発生した．これは，ユーザ辞書の翻訳結果が，すでに組み込み辞書に登録済みの単語と一致してしまったため，**図 7** のようにプログラムの意味がかわってしまった，という現象であった．これを避けるため，教師は辞書の編集をやり直さなければならなかったが，日本語がわからないため，どの部分を再編集しなければならないのかがわからなくなった．(この部分は，日本語と韓国語が両方分かる人の手助けが必要となった)

5.2 Step1: プログラムを読む

韓国の学生たちは，Step0 で作ったサンプルプログラムを，D-trans を通じて読む活動を行った．各学生にはワークシートを配布した．ワークシートには，D-trans が日本語から韓

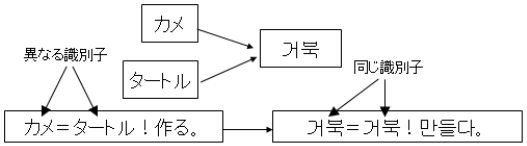

図 7 翻訳の衝突

表 1 Problems in reading program

プログラム番号	指摘があった単語数 (個数/全体)	原因
1	5/25	名前 (2), 表現 (3)
4	1/17	表現
5	2/12	名前 (1), 表現 (1)
12	1/7	名前

国語に翻訳したサンプルプログラムを掲載した．これらのプログラムの中で，理解できなかった部分・不自然な部分をワークシートに書き込み、指摘させた．

表 1 に，学生たちが指摘した単語の個数を示す．

「指摘があった単語数」 に示した数値は，同じ単語が 2 回出現した場合は 1 個と数えている．「理由」には，なぜその単語を指摘したか示す．その原因は 2 種類あった:

- 「カメ太」「カメ吉」，あるいは学生自身の名前などを自動翻訳し，変な翻訳結果となったため (表中の「名前」).
- 翻訳された単語の意味は理解できるものの，表現に違和感を覚えたため (表中の「表現」).

しかし，これらの指摘は全体としては少なく，プログラムの理解をする上ではあまり問題にはならなかった．また，変な名前が出てきたとしても単なる変数名と割り切り，同じ識別子であるかどうかだけをトレースしていけばプログラムの振る舞いを理解することは可能であった．

5.3 Step2: プログラムの修正

学生にサンプルプログラムのうち 1 つを，韓国語に翻訳された状態で修正させた．このプログラムは，**図 8** のような野球のボールを描くプログラムであるが，描画位置がずれているため，絵としては不完全になっている．4 人の学生すべてが，このプログラムを 10 分ほどで正しく修正できた．その後でこれらのプログラムを再度日本語に戻しても，正しく実行することができた．

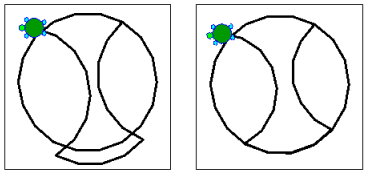

図 8 ボールを描くプログラム (左

6. 議　　論

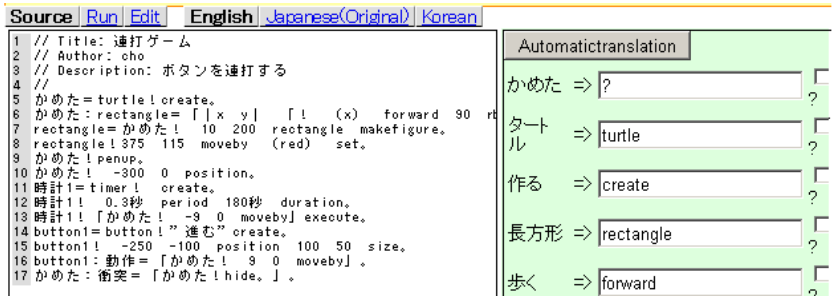

図 9 改良後の辞書エディタ

実験においては，翻訳において不具合がいくつか見つかったが，いずれもプログラム理解を妨げるほどの大きな問題ではなかったと考えられる．したがって，翻訳の性能は一定の品質を保つことができたといえる．

しかし，教師が日本語を知らなかったこともあり，翻訳の間違いを授業前に発見するのが難しいという問題もあった．今回，日本語を知らない教師を選んだのは，韓国人の多くが日本語を自由に読んだり書いたりできないため，実際の典型的な授業を想定してのことである．今回に限っては，筆者らが日本語と韓国語が両方理解できたため，仲介をすることができたが，通常日本側の教師も韓国語は理解できないはずであるので，実際には英語を仲介としたコミュニケーションが必要と考えられる．今後，お互いの国の言葉がわからない教師同士での授業実践も必要である．

また，言葉を理解できるかどうかにかかわらず，エントリ数が膨大でありユーザ辞書の編

集が大変であるとの指摘を教師から受けた．ユーザ辞書は，D-Trans のシステムに 1 回でも登録されたすべてのプログラムに出現するすべてのユーザ識別子を格納しているため，大量のエントリが一斉に辞書エディタに表示された．この場合，翻訳の結果に間違い（自分の国の言葉に翻訳されたものが明らかにおかしい場合など）があっても，見落としてしまう可能性が高い．そこで，次のような対策が可能であると考え，この実験の終了後，辞書エディタを**図 9** のように改良した．

- ユーザ辞書のエントリを，個別のプログラムに関連づけた．これにより特定のプログラムに登場するエントリだけを編集可能にし，一度に表示されるエントリを絞り込んだ
- 教師だけでなく，学生もエントリを編集できるようにした．

7. 今後の課題

7.1 文字列，注釈等の翻訳

現行の実装では，翻訳の対象となっているのは変数，メソッドなどの識別子だけであり，文字列や注釈は翻訳の対象になっていない．これらの要素は，自然言語で書かれた文章や，ファイル名，音楽演奏のための文字列など，多様な目的でつかわれているため，それぞれの目的にあった方法での翻訳が必要となる．これには，そこに書かれている文字列がどのような意味をもっているかを経験則的に推定する必要がある．

7.2 型の概念を利用した翻訳

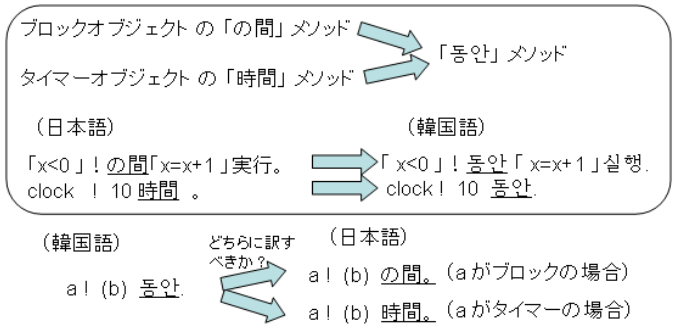

図 10 型の概念を利用した翻訳が必要な例

今回採用している翻訳方法は単純で，用意された辞書に従って単純に文字の置換を行っている．しかし実際には，同じ単語であっても，訳される単語を変化させる必要がある．たとえば，ドリトルの組み込みライブラリの中にも，**図 10** のように，日本語版では，名称の異なる 2 種類のメソッドが，韓国語版では同じ識別子に対応しているものがある．この場合，韓国語から日本語に翻訳をする場合，どちらのオブジェクトに対する命令かを自動的に判定しなければならない．

これらの判定を行うには，プログラム中の式がどんなオブジェクトか，つまり式がどんな型をもつかを推定する必要があるが，ドリトルは動的な型をもつ言語であるので，型推論[3]などの高度な技術を適用する必要がある．実際には，学生の書くプログラムのほとんどは，同じ変数への代入が 1 回しか起きない，または同じ型の値を何度も代入しているものであるので型推論を行うのはそれほど難しくないと考えられる．

7.3 プログラムによる交流活動の確立

今回の実験では，他の学生のプログラムを読んだり書いたりする活動を行ったが，これらの活動は「プログラムの翻訳」を行うか行わないかによらず，次のような理由で難しい点も多かった．

- 学生たちは他の学生のプログラムを読むことはできても，実際にプログラムの振る舞いを「なんとなく」理解するにとどまり，本当にわかっているかどうかの確証が持てなかった．
- 学生たちは他の学生のプログラムを積極的に直そうとしなかった．その理由の 1 つとして，すでにできあがっている作品に対して改造を加えるのは，原作者に失礼であると考えることが多いからである．

これらの点を踏まえ，次のような交流活動を促進するための活動を開発していく必要があると考えた．

- **コード説明** 同じ国の 2 人の学生が「説明する人」と「説明を受ける人」をそれぞれ担当し，他国の学生の書いたプログラムを動かしてみる．説明する人が自国語に翻訳されたプログラムを見て説明を行い，説明を受ける人はプログラムを見ないで作り上げる．もとのプログラムと同等のものが作れれば，プログラムの翻訳が正しくできているといえる．
- **トラブル解決** 学生がプログラムを作っている途中で，プログラムに問題があってうまく動かない場合に，異なる国の学生がそれを見てアドバイスを与える活動．問題となっているプログラムは，読む人それぞれの母国語に翻訳され，翻訳機能をもったチャットを通じてアドバイスを行う．または，教師のほうからわざと間違いが入っているプログ

ラムを与え，何人かで間違いを指摘しあい，修正を行う活動も考えられる．

- **共同開発** 2 人以上で共同で 1 つの作品を制作する活動．学生ごとの役割は教師からあらかじめ与えられていて，たとえば 1 人の学生は絵を描くプログラムを担当し，他の学生は絵を動かす担当，さらに他の学生は音楽を演奏する部分を担当するなど，比較的分業のやりやすいように工夫しておく．
- **コメント交換**
  他国の学生同士で作品に対するコメントを送りあう活動を行う．「プログラムの感想を送り合う」という機能は実装されてはいたが，今回の実験では試行していない．

8. 関 連 研 究

Squeak eToys[4] は，多言語に対応した教育用プログラミング環境である．Squeak eToys は，プログラムの識別子が書かれたタイルを操作してプログラムを作る．そのタイルに書かれた識別子は言語設定により様々な国の言語で表示することができる．Scratch[5] も同様に，タイルを用いてプログラム作り，タイルに書かれた識別子は多言語で表示できる．さらに，ユーザはプログラムを Web サイトで公開でき，他の国のユーザはその公開された作品をダウンロードし，自分の国の言葉で書かれたプログラムに翻訳したものを実行できる．

Squeak Etoys や Scratch においては，翻訳される範囲は組み込みオブジェクトや組み込み命令に限定されている．ユーザ自身が作ったオブジェクトや命令は翻訳されずにそのまま表示されてしまう．D-Trans ではこれにユーザ辞書を加え，プログラムのすべてを翻訳できるようにし，また辞書エディタを用いて新しいプログラムへの翻訳も対応できるようにしている．

9. ま と め

母国語を用いたプログラミング言語を習得した学習者が，プログラミングを通じて国際交流を行うためのプログラム翻訳システム D-Trans を提案した．D-Trans では，学習者自身がつけた変数やオブジェクトの名称も含め，すべてを相手国の言葉に変換し，しかも自然な訳を得られ，プログラムの意味を変えることない結果を得ることを目標とし，まずはプロトタイプシステムを実装しどの程度達成できるかを検証した．

プロトタイプシステムでは，辞書作成インタフェースを利用して，教師が自由にエントリを追加できるようにした．翻訳されたプログラムを外国学生に見せたところ，単純なプログラムであれば問題なく動作し，外国人でも改造ができる程度の結果が得られた．一方， 辞書作成者は，自動翻訳の間違いや，辞書内の単語の衝突を気づかない場合もあった．衝突に対処するためにプログラムごとに辞書を作成する仕組み，複数人が辞書を協力して作る仕組みが必要であることがわかった．

今後は，授業での交流活動をどのように行うかを計画し，実際の授業での翻訳性能を検証していく予定である．

謝　　辞

この研究は，日本学術振興会 二国間交流事業 共同研究「初中等教育における国際コラボレーションプログラミングの研究」（2005.7-2007.6）および日本学術振興会 二国間交流事業 共同研究「国際コラボレーションプログラミングを実現するフレームワークの研究」（2007.7-2009.6 ）による助成を受けました．協力していただいた日韓の先生方，学生の皆様に謝意を表します．

参 考 文 献

1) 兼宗進，中谷多哉子，御手洗理英，福井眞吾， 久野靖：初中等教育におけるオブジェクト指向プログラミングの実践と評価，情報処理学会論文誌. プログラミング，Vol.44, No.13, pp.58–71 (20031015).
2) : Google AJAX Language API, http://code.google.com/apis/ajaxlanguage/.
3) Palsberg, J. and Schwartzbach, M.I.: Object-oriented type inference, *OOPSLA '91: Conference proceedings on Object-oriented programming systems, languages, and applications*, ACM, pp.146–161 (1991).
4) : Squeak eToys http://www.squeakland.org/author/etoys.html.
5) : Scratch, http://scratch.mit.edu.