# Translation Instruction On-line

Robert Reynolds

National Chi Nan University
Fifth Annual Conference on Translation Instruction
National Taiwan Normal University, January 6, 2001

## *Introduction*

I first taught "Introduction to Translation" in 1998 at Chaoyang University of Technology's Department of Applied Foreign Languages. This is a required two-semester course taken in the junior year. Its purpose is to give students an extensive introduction to the pleasures and pains of English to Chinese and Chinese to English translation. It is an important class for all students, since most will regularly do short, business related translation in their future careers. It is also intended to serve as a beginning course for students interested in taking advanced translation courses in the senior year. Students are therefore highly motivated to do well in this course. It was not, however, an easy class to teach.

## *Problems*

It is worth considering why such courses are so difficult to teach: a clear statement of the problem is often the first step to a solution. Here are the main problems I encountered.

The first, and most important, problem in this course was class size. The first time I taught the course, there were 52 students. The second time I taught it, there were 57. These are large numbers for any kind of language course.

Second, translation exercises are always very tough to grade. They are not like composition: a simple theme that students turn into a paper which meets a general list of requirements. Nor are they grammar exercises, where you just check the tense, number, spelling, or whatever and you are done. They belong in a sort of half-way zone; students must be both creative (but not too creative), and faithful (but not mechanical). This makes consistency in grading harder than any other kind of class I have taught. Students will also have vast variation in their approaches, so unless you can compare how they translate, often on a word by word basis, it is very difficult to prepare a class discussion that really catches the main approaches and say more than just vague generalities. This often requires reading every assignment twice: once for comprehension and overview, then once more for individual grading. Even so, it's an almost hopeless job to try and work out a grading metric that you can apply consistently.

Third, the paperwork for a beginning translation course is voluminous; it stacks up on the teacher's desk like airplanes over O'Hare Airport in a thunderstorm, both going out and coming in. Going out, the instructor must provide students with copies of the exercises. (Type one and photocopy 57 times in time for class.) Some of the students will inevitably lose their copy. Zero for the assignment, and a wasted effort for these students. Coming in, students turn in typed translations, a stack of 50 papers for the teacher, who must then

arrange papers by student number, register in his grade book, and begin to grade. If you attempt to grade item by item, the best way to get a sense of student errors and problems for in-class discussion, you will double grading time. If you attempt to add comments, you will triple grading time. (if you do both, that is two times three or six times normal grading time). Omitting comments is particularly regrettable, since comments are often highly repetitious in translation grading, do not require lengthy cogitation, and would be very helpful to the individual students. If you drop papers on the floor, add an extra 20 minutes to half hour to straighten. If you spill coffee on the papers, you not only lose time but face as well. Grading completed, you now add up scores and enter in grade book. Now add at least five late papers scratching out X's in your grade book and entering grades in correct columns, misplacing at least one per assignment, causing problems when you attempt to actually add these numbers at the end of the semester.

Fourth, you must now return assignments to students. After you do, you no longer have the papers and had better have good notes for any kind of in-class discussion. If you want to actually work through some student translations, instead of just presenting your own version, you must have photocopies, unless you want to copy down answers, retype on your own computer, print them out, and photocopy onto transparencies.

## *Solutions Needed*

Now that we've considered the problems, the solutions should be clearer. Here's a summary of what we need.

First, we need a solution to the problem of distribution. This includes the teacher getting exercises to students, students getting answers to the teacher, and the teacher getting grades, corrections, and comments to students. In addition, it would be nice to have a way to distribute the main points of in-class discussions to students, especially for those who were sitting so far back in class that they could not see or hear (and consequently fell asleep). We would also like something which allows access to student answers, even after the papers have been returned.

Distribution solutions should therefore meet the following conditions: a) they should be reliable; b) they should be prompt; c) they should be accessible. By accessible, I mean that neither teacher nor students have to be at designated places at designated times to transmit or receive class related materials.

Second, we need a solution to the problem of sorting the assignments both by student and by translation. This is the most tedious, though not the lengthiest, part of the entire job. Language teachers who give homework assignments are doomed to be paper-pushers; this is not conducive to giving assignments, which in turn is not conducive to learning. Not only are we registering the papers as received (or not), we are viewing and grading exercises item by item. We need a way to search quickly through every student's answer to a single question, sentence by sentence if necessary. This would have the following benefits: a) it would improve the consistency of grading, a very important goal if you want to see results from grading, and a very sensitive issue for students; b) it would greatly facilitate preparation for in-class discussion; if you can't get a handle on common student mistakes, you will wind up either boring your students or leaving them hopelessly lost.

Third, we need a solution which will put numbers into my computer, where they belong. Personally speaking, the less I have to transcribe, transpose, transact, or otherwise mangle numbers, the better off everyone is.

Fourth, we need to get rid of these stacks of paper which make such a mess of teachers' desks and offices, attracting coffee stains and ants.

We can also start making a wish list, like a way to build up the exercise bank every teacher needs to do well at teaching this class.

## *Solution Found*

The solution I found was an open source computer program called WebAssign. WebAssign was developed by Prof. Larry Martin (now deceased) at Chicago's North Park University in 1997. WebAssign let teachers distribute, receive, and grade homework on the Internet using the World-Wide Web http and cgi protocols, and took the form of a script written using the computer language perl. Before we look at the specifics of implementing this solution, however, let's see why it worked.

Our first point was the problem of distribution. The Internet meets our distribution needs. In particular, it possesses the three virtues we want, though not to an infinite degree. It is semi-reliable, but subject to the breakdowns and crashes, system-wide and local, which characterize new systems. It is prompt, but subject to the frequent overloading which often slows it to a snail's pace. It is accessible basically without any limits to teachers. Students often remark that they do not have adequate access, but this is usually after they've missed an assignment. There are ways to get around this in implementation.

Point two, sorting papers, by student or text, now reduces to sorting files, and this is what computers do best. We can now search through all student answers to a particular question. We can grade student by student or question by question. We can copy comments from question to question when students have made the same or similar mistakes. In practice, I feel that that this has actually did improve grading consistency. It also had the salutary effect of sending me to class feeling like I was prepared to discuss homework.

Point three, getting numbers into the computer, is also solved, because the numbers are produced automatically by the program, although at the moment they still require some massaging to get into a spreadsheet. However, as we will see below, it should be possible to almost completely eliminate handling numbers by hand.

Point four, the instructor can almost completely eliminate paper. The only paper now essential to the course is when printing material to turn into transparencies; even this can be eliminated if the teacher has access to computer projectors.

Finally, once all this is available, we can store all course exercises, answers, corrections, grades, and discussion materials on the computer. They are in reasonably accessible form, so that we have the beginnings of the test bank on our wish list.

## *Implementing the solution*

So how did I actually put this solution into effect? Like all computer solutions, the effort involved was substantial, but given the advantages listed above, it has been well worth my trouble, and I expect the same would hold true for most teachers.

The first step is to obtain the program. Professor Martin's program is not the only such program, and undoubtedly more will soon become available. Such programs generally come in two forms: open source programs, which are generally developed for non-commercial concerns, and given away at no cost, and proprietary programs, which are usually commercial works provided with various forms of service.

Professor Martin's program is in practice of the first type, but he has not formally renounced commercial use of his code. This is a complicated problem, which this paper will not be able to discuss in detail. It is, however, something to be wary of. Copyright issues are one of the most important issues teachers face today, especially in computer-related applications.

Returning to Professor Martin's program, WebAssign, the program is what is called a Perl script. This means it is a text file, written using a human readable computer language called Perl, which is itself a free open source application. Scripts are very easy to use and to modify.

After you have obtained the script, you will need access to a web server, including permissions to run programs and set up access groups. This means that many teachers will have problems using this program through their school's server. Permissions are essential for program security, and many schools do not allow teachers to have the necessary ones. This is a problem which time (two to three years perhaps) may cure. Hopefully network security will improve, and schools will become more open to online instruction. In my case, I got around this problem by installing WebAssign on the department web-server, rather than a university wide server. If your department has its own web-server, probably just a regular computer running the required software and possessing the right connections, you should be able to run WebAssign.

Next, you will need assistance from someone who understands and uses server software. Although I know something about servers, I did not succeed in setting up the program myself. It was only with the assistance of our department's computer technician, Mr. Li Jengru, that I was able to get WebAssign up and running. Even Mr. Li was stumped by one or two problems, resolved through the timely aid of Professor Hung Chao-kuei of Chaoyang's Information Management Department, an expert in server software, Perl, and other professional arcana. This was by far the most technical part of the entire process, and the details will obviously vary for each case.

If your efforts are crowned with success, the end result is that you have a correctly configured script in the correct location on a server. You will never touch the script yourself. You will have a user account on the server, which you may access from your office computer. This account should be divided into at least two directories: one to keep web page files, which are essential to make things look nice, and a second to keep program files.

Web pages will include things like assignment explanations, notes, and dates. Most important, this is where you will create a link to WebAssign so that students can access the program. WebAssign files should be kept in a different directory from your web pages for security. This is very important, and will probably be generally true for all such programs. In the program directory, you will have three kinds of files: 1) most important, a text file containing the assignment itself ( I call this the assignment file). The assignment file contains the text of the assignment, plus formatting codes that WebAssign needs to turn the text into various kinds of questions: fill in the blank,

multiple choice, essay, correct the sentence, math or whatever. 2) a configuration file for each assignment file, telling the program where and when to send and retrieve files. 3) Student answer files, which are produced at the rate of one file per student per assignment. In practice, this works out to a lot of files (remember I had fifty seven students using the system), and if you want convenient access to them, you will have to devise your own directory structure. This is still a simpler matter than finding a place to put hundreds of sheets of paper.

Of these three types of files, the third type should eventually be handled entirely by the program. Perhaps the best long range solution is a database. When considering these online programs, those which boast of using a database probably deserve first consideration, for ease of configuration and maintenance.

The second type is more troublesome. I don't think that such configuration files can ever be completely dispensed with, since they will include information such as when the assignments are available, when they are due, what weight questions receive and so forth. WebAssign uses a simple format to handle this, but the best long term solution should probably be a graphical user interface of some sort. This could reduce the chance of problems in configuration to almost zero. I know of no open source programs that currently offer this.

Last, the assignment file itself is a knotty problem. It is the most essential piece of the system, but requires the teacher to understand something of how to make the system work. These means the teacher will have to spend at least an hour or two learning the format. In addition, it is easily broken; one extra line at the beginning will make the whole assignment display incorrectly. Common symptoms of mistakes include incorrect formatting, incorrect numbering, blanks where there should be no blanks, formatting codes appearing in the text and so forth. It often takes several revisions before it will display correctly.

There is one final component which should be mentioned here, one that caused many problems for us. You must have a list of students in a very particular format in order to make the system work. This is a complex problem, but it is a problem all these online instruction systems must face. The ultimate solution is to allow the program to hook up with elements in the school's administrative network, such as the email server. This would simply things immensely. In my case, however, the care and maintenance of this student list was a significant part of the time I devoted to the system.

When both configuration file and assignment file are ready, you send them from your computer to your account on the web-server. This should be as simple as copying files from one directory to another.

Then you must provide a link to the configuration file somewhere on your web page(s). I use one web page especially for WebAssign assignments, which also includes other information such as date due, date to be returned, links to class notes, and so forth. I keep a copy of this web page on my computer and every time I have a new assignment, I add the information, then copy it to the correct directory in my user account. Thus, for every assignment, you will have to copy at least three files to your account: configuration file, assignment file, and updated web page with links to the configuration files.

When the assignment displays correctly, you are ready to go. Now it is the students' turn. I have found that students adapt fairly easily to the demands of

WebAssign. The most common problem is forgetting their passwords. This is where a connection to the school's administrative system would be very useful.

Another very common problem is that students will accidentally submit the assignment before they have finished inputting all their answers. You submit the assignment by pressing a screen button labeled "submit." This writes the student's answers to a file in your account. Unfortunately, once this file is written, it will not accept any further answers. If you press submit too soon, you cannot do anything more for the assignment. This is something which revision of the program should address.

A third problem I have seen is that students sometimes put their answers in the wrong places. This is not acceptable; a minimum requirement for college students is that they can fill in blanks correctly. Answers put in the wrong blanks are treated as wrong.

The due date for the assignment arrives and you are now ready to go. Currently WebAssign does not allow you to set cutoff dates for assignments. The best way to stop students from submitting assignments after the due date is to manually break the link to the assignment on your web page. This is a very important point for me, and in looking for similar programs, I would certain give one that could set cutoff dates a high preference.

Now it's time to begin correcting the assignment. I strongly suggest doing this question by question. The correction interface on WebAssign is not pretty but it is adequate; it allows you to insert comments for each essay and a grade. It turns all the questions into one long web page, so when you use your browser's search function to find words or phrases, you are indeed searching all the student answers to a particular question. You can use the browser's copy, cut, and paste functions to do anything you want with your comment text. If you need an example from a student answer, you can also use the copy function to copy student answer text and paste it in a word processing application.

This is a giant leap from the original hand process. I would also like to point out that this type of computer assisted grading allows visually handicapped instructors to grade their student assignments without assistance from a third party.

Once grading of the assignment is complete, you have to make the grades and comments available to the students. This is done by changing one character in the configuration file. I also like to put reference answers up on my class site. I do this by typing out my answers (or copying good student answers) into a file containing the original exercise questions, converting it into html, and copying it to my server account. I also have to add a link to the answers to the class web page.

## *Problems with the Solution*

There is no such thing as a free lunch. For the same reason, my solution to the labors of "Introduction to Translation" requires work before you reap its benefits, and even then it has limitations and disadvantages. In discussing these, it is convenient to distinguish between limitations and disadvantages specific to WebAssign, and limitations and advantages common to computer and network-based solutions in general. I will first discuss the latter.

Computer-based solutions have the common limitation that they require access to computers. Students have made this point a number of times. Ten years ago, it would certainly have been a significant deterrent. Now, however, it is much less so.

Computers are common in households and ubiquitous in universities. The solution proposed here does not require students to go out and buy a computer. All they need is access to a computer for perhaps an hour a week. This condition is easily satisfied in Taiwan.

Network-based solutions have the further limitation that they require a network connection. This is not so easily satisfied, since many student have neither modem nor Internet account to use at home. On-campus computers can make up for this somewhat, and off-campus access to the Internet is certainly much easier to find than it was five years ago, or even three years ago.

Another way to minimize student need to get on-line is to post homework assignments outside of the WebAssign program which can be downloaded and printed using a regular printer. Students can then obtain copies of the homework from classmates and fill in the answers by hand. Then when they get on line, all they have to do is copy the ready answer into the appropriate space. This is an approach I have been experimenting with this semester and according to students it does reduce the need to get on-line. Reference answers are posted on-line and can be printed out the same way. The only time students need to get on-line then is to enter their assignments, and to view their grades and comments.

For teachers, there are two barriers to on-line solutions: 1) installation. Unless the teacher has extensive expertise in computers, this will always require technical assistance. 2) learning how to operate the system. This problem will obviously vary with the quality of the program. The most persistent problem, is certain to be preparing the assignment file. Even with future improvements, this seems likely to be a basic problem of all such systems. The only way to avoid it is to provide a standard series of assignments, and the teacher simply makes scheduling decisions, and grades essays. This naturally reduces the flexibility and usefulness of the system.

Limitations specific to WebAssign are perhaps more of a deterrent. WebAssign can certainly be improved, by adding a cutoff date function for example. Installation could be automated, the grading interface could be improved, converting scores to spread sheets could be improved, a GUI interface for writing configuration files could be written, and so forth.

On the other hand, WebAssign has a number of strengths which I will cite here in hopes of providing a guide to teachers interested in evaluating similar software. First, WebAssign is of course free, subject to the conditions imposed by its author, Larry Martin. Second, source code is provided, allowing the competent, brave, or foolhardy to revise and adapt where necessary. Third, it is a small Perl script, not a huge program installation, and will run on most web servers which have the ubiquitous Perl installed. I have had only one student complain about what appear to be incompatibilities between WebAssign and browser software. Although I noted above that a database for answer storage would simplify maintenance and structure, the fact that WebAssign does not use one actually makes it more accessible.

Assignment file formatting is a central problem and it needs to be redone in a way which renders errors much harder to make. It is of course a good thing that it is a simple text format, and does not require the instructor to write his own javascript or java programs for each assignment. Nor does it require special interfaces to insert non-readable markings. In the end, however, teachers will have to spend some time to learn

format. Equally important, they must be willing to write their own assignments and check them for errors and consistency.

## *Conclusions*

Summarizing our conclusions, WebAssign and similar programs simplify aspects of the teacher's work by substituting electronic files for written papers. They improve access to student work for both teacher and student. They can greatly improve consistency of grading. If used properly, they can also reduce to time required to complete many highly repetitive (boring) tasks, such as recording grades. To some extent, they may tend to steer teachers toward certain types of homework, due to the limitations and conveniences of the program, but with due care this can be avoided. Translation instruction, in my opinion, is actually better suited than other language course work to on-line work.

On the other hand, the uses of on-line instruction described in this paper do not fundamentally change classroom content and roles, and instructors who are looking for fundamental change will have to go elsewhere.