British Journal of Educational Technology Vol 42 No 3 2011 482–499
doi: 10.1111/j.1467-8535.2009.01030.x

The role of community feedback in the student example authoring process: An evaluation of AnnotEx

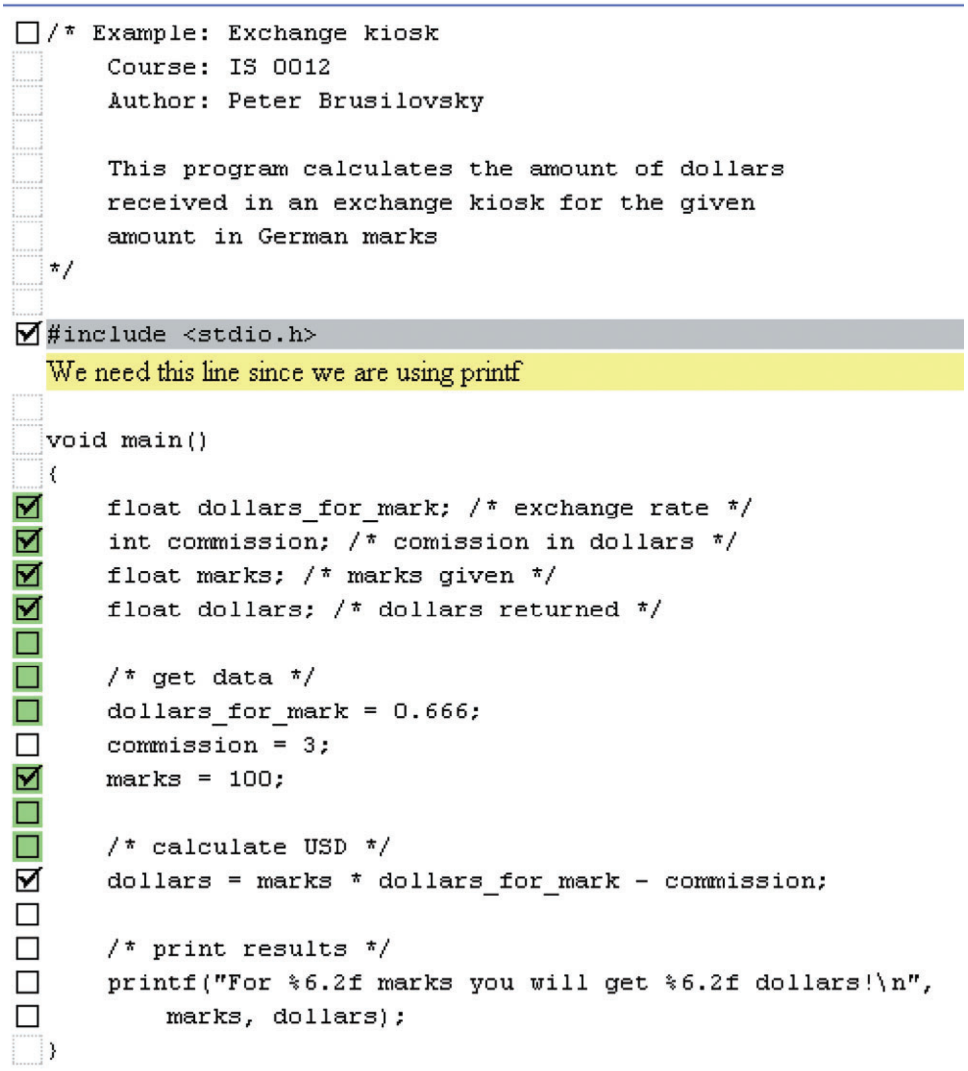
I-Han Hsiao and Peter Brusilovsky

I-Han Hsiao is a PhD student at the School of Information Sciences, University of Pittsburgh. Peter Brusilovsky is an Associate Professor at the School of Information Sciences, University of Pittsburgh. Address for correspondence: I-Han Hsiao, School of Information Sciences, University of Pittsburgh, 135 North Bellefield Ave. Pittsburgh, PA 15260, USA. Tel: +1 412 624 9437; email: ihh4@pitt.edu. Peter Brusilovsky, School of Information Sciences, University of Pittsburgh, 135 North Bellefield Ave. Pittsburgh, PA 15260, USA. Tel: +1 412 624 9404; fax: +1 412 624 2788; email: peterb@pitt.edu

Abstract
This paper explores a new approach to engage students in authoring educational content. This approach was implemented in AnnotEx (Example Annotator) system, which allows students to annotate computer programming examples with line-by-line explanations and review annotations produced by ther peers. A controlled study of AnnotEx presented in this paper evaluated the impact of the community peer-reviewing process on the quality of produced annotations and student learning. The study confirmed that community feedback increases the volume and the quality of produced annotations and positively affects the work of weaker students. The peer-rating process enabled the community to distinguish good and bad annotations. Peer comments provided efficient guidelines for improving annotations and caused a significant increase in quality.

Introduction and motivation

Learning from examples is a common approach when mastering the art of computer programming. In this field, examples help students to master the semantics of programming language and to form problem-solving skills. Multiple code examples, ranging from small code snippets to complete programs, can be found in any programming textbook and are frequently also provided on an attached CD or a web site supporting the textbook. Researchers in the area of computer science education have also recognised the educational power of examples and have suggested a number of interactive systems, which have attempted to increase the value of examples as tools for learning (Brna, 1998; Burow & Weber, 1996; Davidovic, Warren & Trichina, 2003; Gómez Albarrán, 2005; Linn, 1992; Pirolli & Anderson, 1985; Weber, 1996).



```
/* Example: Exchange kiosk
    Course: IS 0012
    Author: Peter Brusilovsky

    This program calculates the amount of dollars
    received in an exchange kiosk for the given
    amount in German marks
*/

#include <stdio.h>
We need this line since we are using printf

void main()
{
    float dollars_for_mark; /* exchange rate */
    int commission; /* comission in dollars */
    float marks; /* marks given */
    float dollars; /* dollars returned */

    /* get data */
    dollars_for_mark = 0.666;
    commission = 3;
    marks = 100;

    /* calculate USD */
    dollars = marks * dollars_for_mark - commission;

    /* print results */
    printf("For %6.2f marks you will get %6.2f dollars!\n",
        marks, dollars);
}
```

Figure 1: WebEx system

An original approach to using examples to support online learning in programming courses was suggested in the WebEx system (Web Examples), developed by our research group several years ago (Brusilovsky, 2001). WebEx provided Web-based interactive access to examples enhanced with line-by-line comments, allowing students to browse the comments at their own pace and chosen sequence (Figure 1). More recently, the NavEx system (Navigation to Examples, an extension to WebEx) was introduced in order to provide students with personalised guidance to the most appropriate examples (Yudelson & Brusilovsky, 2005).

Both WebEx and NavEx were used for a number of semesters in several programming classes in C, Java and SQL that were taught in several institutions, ranging from large research universities to community colleges (Brusilovsky, Grant, Hsiao, Moore & Sosnovsky, 2007). The classroom studies demonstrated that these tools were highly appreciated and heavily used by students (Brusilovsky & Yudelson, 2008). The amount of student work with explained examples increased significantly and helped students to gain a better knowledge of the subject.

With the rapid growth of WebEx and NavEx usage and an increase in the number of courses and colleges using these systems, we were surprised to encounter an unexpected problem: the lack of explained examples. When WebEx was originally envisioned, we expected that a community of teachers would contribute to the authoring of annotated examples, creating a large and diverse shared collection. A user-friendly authoring tool was readily available for prospective authors. However, despite the active use of the system, almost no new annotated examples were contributed by instructors, other than by the authors of the system. While classroom teachers regularly maintain a good set of code examples in their courses, they have limited time to annotate these examples, perhaps dozens, which are necessary to support a course. Thus, there are simply too many examples and too few annotations.

This paper explores an alternative solution to example authoring. In the spirit of modern 'users as creators' trend promoted by Web 2.0, we explore the feasibility of harnessing the students' energy to create *example annotations*. If successful, this approach could offer several benefits. Not only does it remove the burden from the instructors, allowing them to focus on other pedagogical tasks, but there is also evidence that authoring (rather than only using) examples can contribute to students' knowledge of the subject. Jonassen and Reeves (Jonassen & Reeves, 1996) contend that students are likely to learn more by constructing hypermedia instructional materials than by studying hypermedia created by others. Meanwhile, Chi *et al* (Chi, Bassok, Lewis, Reimann & Glaser, 1989) showed that self-explanations in the context of learning about mechanics from worked-out examples had rather dramatic effects on the participants' ability to solve problems on their own.

To explore the potential of engaging students in authoring example annotations and to determine the best approaches to organise this process, we designed the AnnotEx system (Example Annotator). AnnotEx supports a *community-based approach* to creating example annotations. It involves community members (students) in both developing annotations and reviewing annotations producing by their peers (to assure the quality of the final product). The pilot classroom study of the first version of AnnotEx (Hsiao & Brusilovsky, 2007) provided some evidence in favor of both involving students in creating example annotations and the peer-reviewing technology employed in the system.

The work presented in this paper attempts a deeper exploration of the community-based approach to annotating program examples. It explores this approach in the context of similar work, presents an enhanced version of the AnnotEx system, and reports the results of a controlled study. The study was designed to explore the effects of our peer-reviewing technology, as well as to assess the learning potential of the community-authoring approach. The rest of this paper is organised as follows: In the second section, we first survey related work that uses collaborative example-based learning. In the third section, we describe the system, AnnotEx, which is designed to cater to a community-based collaborative authoring environment. In fourth section, the study design is presented. Major results and a detailed analysis are presented in fifth section. In sixth section, we report a more specific analysis of the annotations and comments. In seventh section, we report the subjective analysis, then summarise the results in eight section.

Related work

Our work on the community-based approach for authoring example explanations was motivated by two streams of work in the area of educational research: the studies of example explanation and peer reviewing. According to Chi and her colleagues (Chi *et al*, 1989), students can learn much when attempting to explain examples. 'Self-explanations', formulating the unwritten steps of an example or concept, help students understand examples and problems (Chi *et al*, 1989; Recker & Pirolli, 1990). Other cognitive science studies have shown that students acquire less shallow procedural knowledge by specifically giving an explanation (Aleven & Koedinger, 2002). The benefits of generating self-explanations extend to explanations created in response to specific questions (Pressley *et al*, 1992).

The value of peer reviewing in the context of education has also been explored by a number of authors (Cho, Schunn & Wilson, 2006; Fujihara, Ohnishi & Kato, 2006; Pinkwart, Aleven, Ashley & Lync, 2006; Sitthiworachart & Joy, 2004; van den Berg, Admiraal & Pilot, 2006). The original application of this technology in education was, however, not content authoring, but peer assessment and grading. CPR (calibrated peer review) and SWoRD (scaffolded writing and rewriting in the discipline) are two classic examples in the work of peer review. CPR supports student learning by giving them writing assignments about important course topics (Robinson, 2001). Through the peer review process, students learn to read for content. At the same time, it is an exercise to develop reviewing skills. In the broader sense of education implications, a perceived helpfulness is likely to mediate between the feedback and later written revisions (Rucker & Thomson, 2003). SWoRD is a web-based peer review system (Cho & Schunn, 2006). It supports the whole cycle of writing, reviews, back reviews and rewriting. SWoRD also examines review accuracy. It has been widely used in many courses and disciplines. The empirical evaluations of SWoRD have shown that it is effective in improving writing and helps students gain content knowledge, as well as improve their writing and reviewing skills.

The use of peer-reviewing technology to assure the quality of community-produced educational content has been pioneered by educational repositories such as Merlot (Cafolla, 2006), however, it was targeted to teachers and domain experts as content producers. It was not until recently that several research groups attempted to bring the work on student authoring and peer review together and explored the feasibility of peer-review-based student content authoring. ExplaNet (Masters, Madhyastha & Shakouri, 2008) the first system to explore this approach, is a web-based learning environment where students can author and share explanations to questions provided by teachers. Students submit explanations and review explanations authored by their peers. Students then revise and resubmit their answers. The study of ExplaNet demonstrated that students can benefit from the process of viewing peer-authored explanations in an anonymous, asynchronous, web-based environment. The learning benefits that students receive from face-to-face peer instruction and collaboration can be extended to a virtual environment.

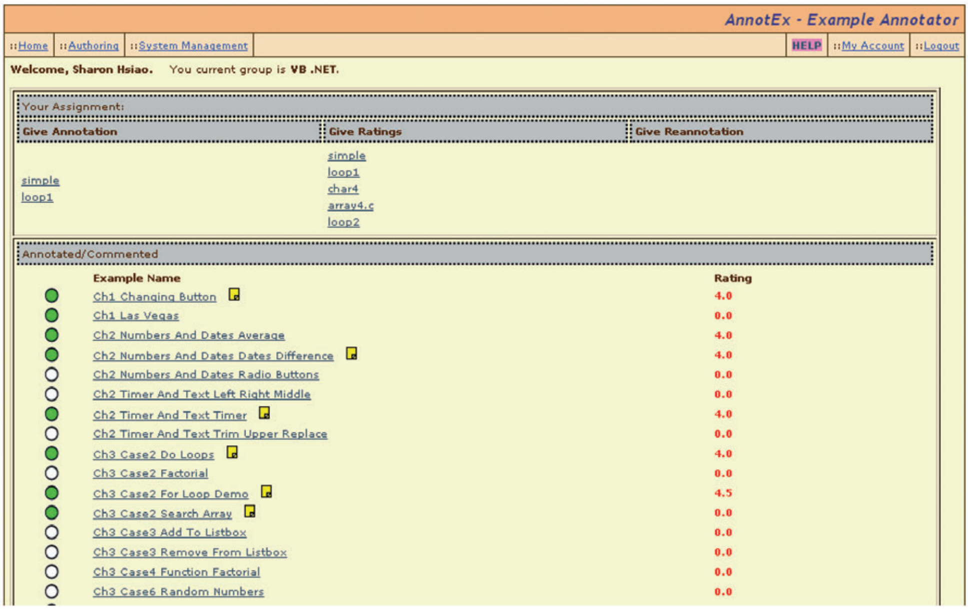
Our preliminary study (Hsiao & Brusilovsky, 2007) followed this line of work: we used a collaborative example-authoring system to collect example annotations from students and observed the value of re-annotation based on community feedback. Students were initially assigned to annotate two examples. After annotating, they provided ratings and comments for six other students' example annotations. Lastly, the low ratings group was randomly reassigned back to the students. The study confirmed that the community successfully filtered out good and bad annotations, and that the re-annotation process improved the quality of the annotations. In addition, the annotating example assignment was perceived as being highly helpful in promoting understanding of the content.

The success of the pioneer projects reviewed above and the inspiration of Web 2.0 caused an increase in the popularity of the idea to involve students into authoring and reviewing educational content, both as a research area and as a practical approach (Abad, 2008; Denny, Luxton-Reilly & Hamer, 2008; Gotel, Scharff & Wildenberg, 2008; Masters, Madhyastha & Shakouri, 2008). However, the number of detailed studies is still too small, so we hope that the work presented below help further the development of this stream of work.

AnnotEx: example annotator system

AnnotEx, Example Annotator System (http://adapt2.sis.pitt.edu/annotex/), was developed to support community-based authoring of annotated programming examples. It allows a community of students (for example, a class) to author annotations to examples, as well as to provide comments and ratings on the annotations produced by their peers. Each member from the community has three tasks to complete in the example annotating process. The first task is to author the annotation of the example. The second task is to provide ratings/comments about the example annotations. The third task is to re-annotate, ie, to edit and expand the original annotations. AnnotEx is a Web-based system which can be accessed anywhere with a web browser and an Internet connection.

The AnnotEx interface (Figure 2) divides the screen into two sections. The upper section represents student tasks; the lower section illustrates the example pool of the community. The tasks are sequentially arranged from left to right, based on the process flow,


AnnotEx - Example Annotator

::Home ::Authoring ::System Management HELP ::My Account ::Logout

Welcome, Sharon Hsiao. You current group is VB .NET.

Your Assignment:

Give Annotation	Give Ratings	Give Reannotation
simple loop1	simple loop1 char4 array4.c loop2	

Annotated/Commented

	Example Name	Rating
●	Ch1 Changing Button 🗒	4.0
●	Ch1 Las Vegas	0.0
●	Ch2 Numbers And Dates Average	4.0
●	Ch2 Numbers And Dates Dates Difference 🗒	4.0
○	Ch2 Numbers And Dates Radio Buttons	0.0
○	Ch2 Timer And Text Left Right Middle	0.0
●	Ch2 Timer And Text Timer 🗒	4.0
○	Ch2 Timer And Text Trim Upper Replace	0.0
●	Ch3 Case2 Do Loops 🗒	4.0
○	Ch3 Case2 Factorial	0.0
●	Ch3 Case2 For Loop Demo 🗒	4.5
●	Ch3 Case2 Search Array 🗒	0.0
○	Ch3 Case3 Add To Listbox	0.0
○	Ch3 Case3 Remove From Listbox	0.0
○	Ch3 Case4 Function Factorial	0.0
○	Ch3 Case6 Random Numbers	0.0

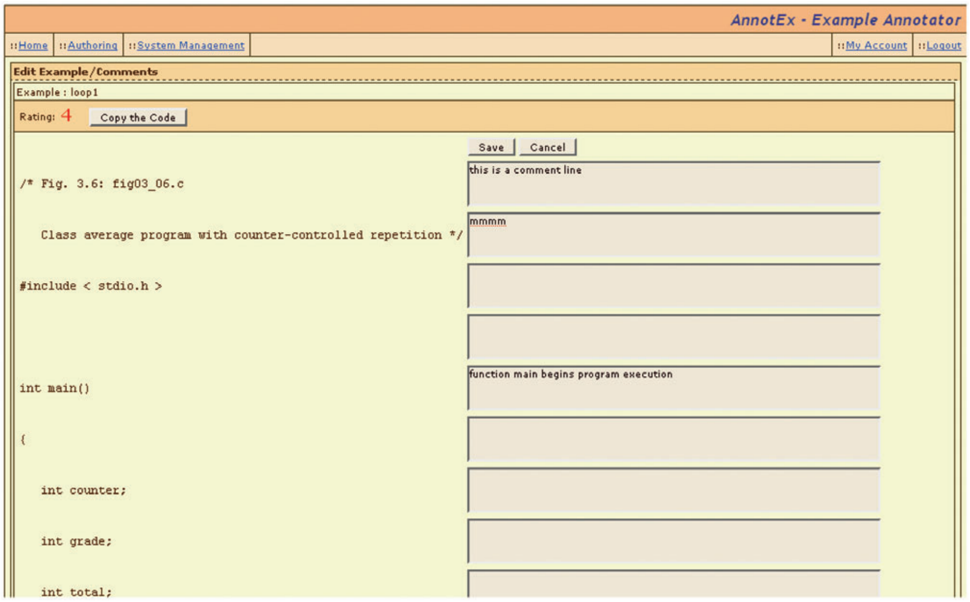
Figure 2: The main page of a community on AnnotEx

annotating, rating/commenting and re-annotating, respectively. Upon the completion of each task, she/he can continue on to the next task. The example pool of the community is available at all times, regardless of which task s/he is doing. AnnotEx is enhanced by an evaluation prototype. A five-star rating mechanism has been adopted to indicate the quality of the evaluation. Ratings are collected from the second task. The average ratings of the example from the community will be shown on the main page.

In Figure 2, the main page of a community on AnnotEx, the green circles mark which examples are annotated, while white ones are not annotated. Yellow post-it icons show comments on the annotations. Ratings are shown at the right. Figure 3 presents the first task, an annotation task. The interface is divided into left and right. The left side displays the example code, line-by-line. The right side is the place for students to write their own corresponding annotations, line-by-line. Students can also click on the button at the top to copy the program code. Figure 4 is the interface of the second task, rating and commenting. The top of the screen is the area to provide ratings. The main body consists of three parts: (1) on the left, the example code again appears in black; (2) blue letters in the middle are annotations, corresponding line-by-line to the example code; and (3) on the far right, students provide comments, line-by-line. The third task, re-annotating, has the same interface as the first task.

Study design

The goal of this experiment was to assess the impact of community feedback and the value of feedback-based re-annotations. We aimed to investigate the effects of



Example : loop1

Rating: 4

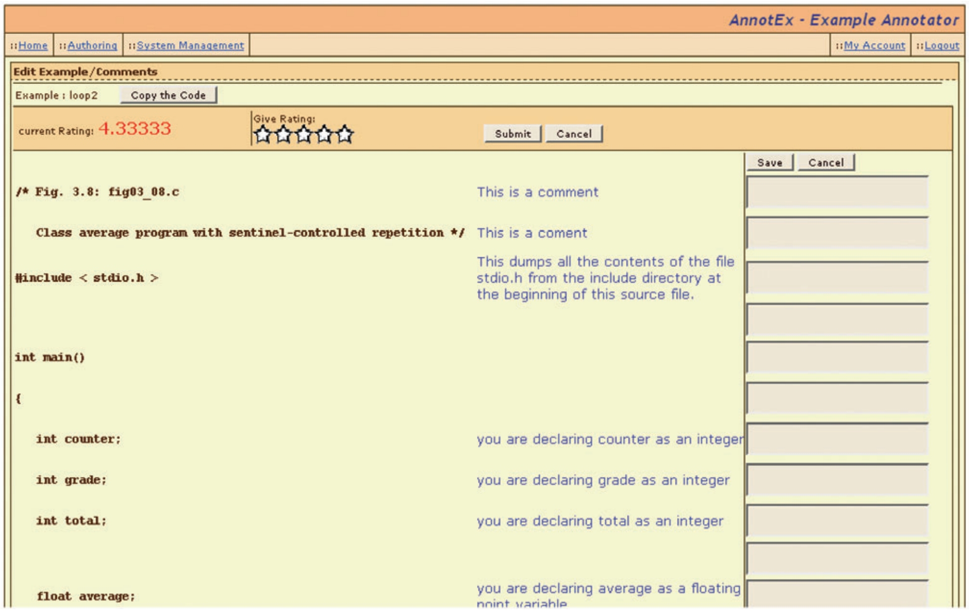
```
/* Fig. 3.6: fig03_06.c

   Class average program with counter-controlled repetition */

#include < stdio.h >


int main()

{

   int counter;

   int grade;

   int total;
```

this is a comment line

mmmm

function main begins program execution

Figure 3: The page to provide annotations to an example



Example : loop2

current Rating: 4.33333

```
/* Fig. 3.8: fig03_08.c

   Class average program with sentinel-controlled repetition */

#include < stdio.h >


int main()

{

   int counter;

   int grade;

   int total;


   float average;
```

This is a comment

This is a coment

This dumps all the contents of the file stdio.h from the include directory at the beginning of this source file.

you are declaring counter as an integer

you are declaring grade as an integer

you are declaring total as an integer

you are declaring average as a floating point variable

Figure 4: The page to provide ratings and comments to an example annotation

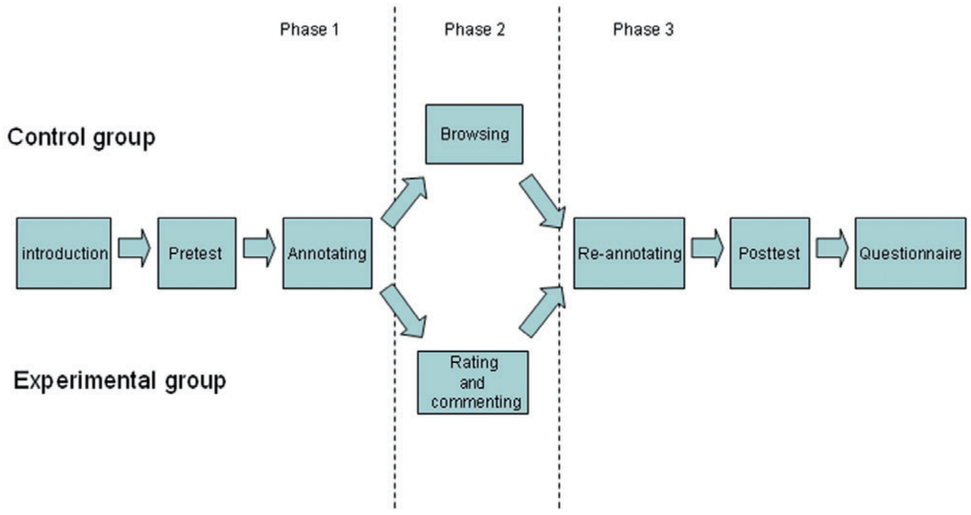


Figure 5: Study design process flow

re-annotations in a controlled study with and without community feedback. The hypothesis was that community feedback would help improve the annotation quality. Thus, the study was designed with control and experimental groups. Only the subjects in the experimental group were able to give or use peer feedback. Following the design of previous peer-feedback studies (Cho & Schunn, 2006), the peer feedback included both textual comments and numeric ratings.

Subjects

There were 30 subjects from the National Taiwan Normal University. They were either freshman or sophomore computer science students. The subjects were randomly divided into control and experimental groups, consisting of 15 subjects each. Each subject was rewarded with a 200NTD gift card (about $6 USD), after completing the experiment.

Design

The examples used in the study were focused on topics covered in the Introduction to Programming Language course. Examples were randomly assigned to students for annotation and commenting. The study lasted 90 minutes. The overall process flow of the experiment is presented in Figure 5.

Introduction

The study began with an introduction, which included an explanation of the experimental purpose and operation.

Pretest
A quiz on Loops was given to both the control and experimental groups. Questions included answering what the final value of the variable would be and what the program printed out after a code fragment is executed.

Phase 1 (annotating)
Each student from both groups was asked to write annotations to one example on the topic of Loops.

Phase 2 (rating and commenting)
The control group subjects were able to browse all the examples annotated by the community, which included their own annotations. However, they were not able to give comments or ratings to any of them. The experimental group could not only review all the examples with community annotations, but each student was also specifically asked to give comments and ratings on the annotations in six examples. Ratings were scaled from 1 to 5, strongly negative to strongly positive.

Phase 3 (re-annotating)
The same example assigned at the first phase was re-assigned for re-annotation. The subjects of both groups were able to use whatever they had learned from seeing annotations produced by the community to improve their own annotations. In addition, the subjects of the experimental group were able to see peer comments left for their own annotations and use these comments when trying to improve their past work.

Posttest
A quiz on the topic of Loops was given to both groups. Questions types remained the same as during pretest.

Questionnaire
The students were asked to answer a 15-question questionnaire using a 5-point Likert scale (*strongly disagree*, *disagree*, *no strong opinion*, *agree*, *strongly agree*). Free text remarks could also be added.

The main results

The main goal of our study was to assess the effect of the peer review-based example annotation process on the annotation results and student knowledge. The following dependent variables were used to investigate the main effects of the study:

- *Annotation rate:* the ratio of annotated lines to total program lines.
- *Annotation quality:* the quality of annotation was measured by expert ratings.
- *Student knowledge:* the knowledge was measured by the pre and posttest scores, ranging from 0 to 10. A score of 10 meant that all test questions were answered correctly.

Table 1: Summary of the control and experimental groups

	Annotation		*Re-annotation*	
Group	*Control*	*Experimental*	*Control*	*Experimental*
Annotated lines	8.3	9.2	11.3	18.8
Annotation rate	27.03%	31.32%	34.28%	56.92%
Expert ratings	1.99	2.40	2.11	3.69
Standard deviation (σ) of expert ratings	1.11	1.38	1.02	0.57

Based on the pretest scores, no significant differences were found between the knowledge levels of the control and experiment groups before the start of the experiment ($p = 0.22$).

For both the control and experimental groups, the annotations collected after phase 1 and the re-annotations collected after phase 3 were passed through Expert Review to be rated for quality examination. Each annotation was rated by two experts. Both were PhD students. One was from the National Taiwan Normal University Computer Science and Information Engineering Department and had 2 years experience as a teaching assistant for an Introduction to Programming course. The other was from the School of Information Science, University of Pittsburgh, with 6 years of professional Java programming experience. The data summary, including before and after the re-annotation for both the control and experimental groups, is provided in Table 1. The following methods were used to analyse this data to determine the main effects of the experiment.

Community feedback increases the annotation rate

As shown in Table 1, the number of annotated lines and the annotation rate increased for both groups after re-annotation. However, without community feedback in the control group, the increase was very modest (7.25%) and not significant ($p < 0.1$). On the other hand, after re-annotation based on community feedback in the experimental group, the annotation rate increased from 31.32 to 56.92%, and the increase was significant ($p < 0.05$). For the experimental group, more than half of the example codes were augmented with explanations after re-annotation. Another interesting influence of the peer-review process was a strong decrease in the variability of the expert ratings. While the standard deviation of the expert ratings changed very moderately for the control group, it decreased more than twice from 1.38 to 0.57 for the experimental group. A deeper investigation of the mechanism of this change is presented in later section.

Community feedback improved annotation quality

At the end of the study, the annotation quality (as measured by expert ratings) increased for both groups (Table 1). However, for the control group, the increase was very small (0.12) and not significant ($p = 0.764$). In fact, 6 out of 15 examples actually scored lower than before re-annotation (Figure 7). For the experimental group,

Table 2: Student performance, before and after the annotation process

Knowledge test scores	*Control group*	*Experimental group*
Pretest	9.28	8.73
Posttest	9.57	9.60
Significance (*p*-value)	0.24	0.12

the increase was very sizeable (1.29) and significant ($p < 0.01$). The results allow us to attribute the enhanced quality of the annotations to the community feedback mechanism.

Students' knowledge increase was not significant

Following the insight we gained from previous research on student authoring and self-explanations cited in the introduction (Chi *et al*, 1989; Jonassen & Reeves, 1996), we hypothesised that example authoring would be a meaningful educational activity for the students themselves, increasing their understanding of the subject. To compare the students' knowledge before and after the example annotating process, we analysed their pre and posttest scores. The maximum number of points that could be gained was 10. As shown in Table 2, the test score of the experimental group increased 0.87 points (9.9%) after their work with examples. The score of the control group also increased; however, the increase was smaller (0.29 points). However, both increases were not significant, probably due to the very short length of the study and small sample sizes. While subjective data analysed below provided some evidence in favour knowledge increase, a longer or/and larger study may be required to reliably demonstrate using pretest/posttest approach that annotation process benefits student learning.

A deeper analysis

This section attempts to reveal the mechanism behind the beneficial effects of the peer review-based annotation and learning, and to discover potential ways to improve this process even more. One of the issues examined in this section is the comparison of good versus poor annotation performance. As shown on Figure 6, the quality of the original annotations (as rated by experts) differed widely from student to student. Some annotations were very good even in their original form, while others were very poor. Below, we analyse why good and bad annotations have been affected differently by the peer-review process. In addition, we discuss how good and poor performance can be distinguished. The ability to recognise good performance can decrease the overhead of the annotation process: it opens up the option of running only poorly annotated examples through the intense, resource-consuming re-annotation process while accepting good annotations as they are.

The community successfully distinguished between good and bad annotations

As can be observed in Figure 6, the community ratings of examples before re-annotation are very close to the expert's ratings. Formal analysis shows that the

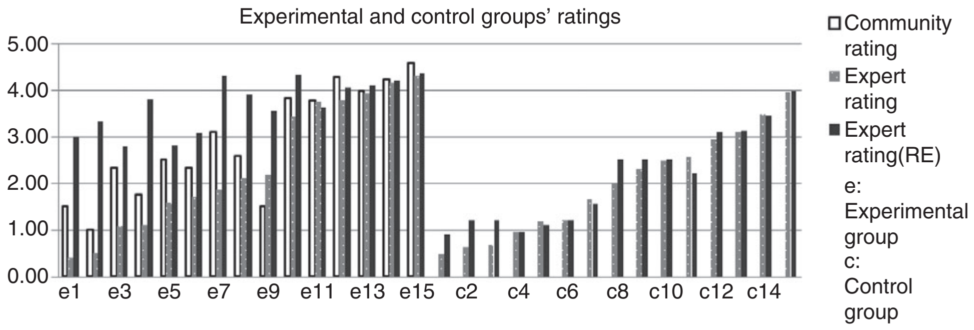


Figure 6: A sorted expert rating figure for each annotated example's from both groups: community rating, expert rating and re-annotation expert rating

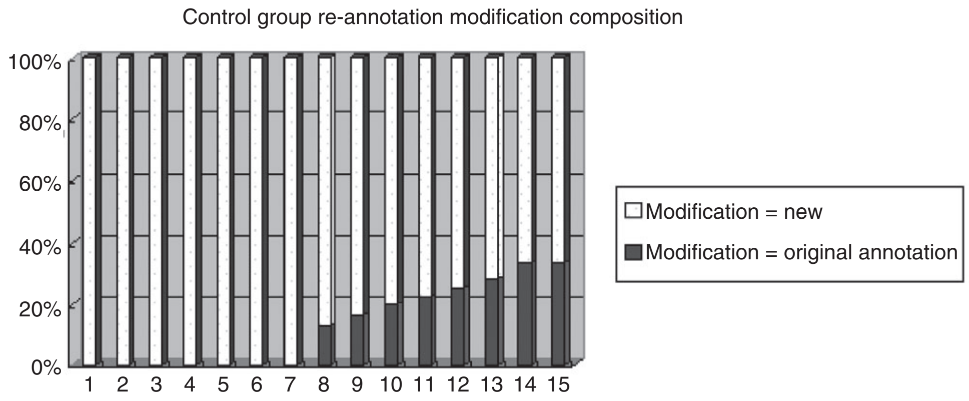


Figure 7: The types of re-annotation in the control group

correlation between them is high ($r = 0.93$). It indicates that the community is capable of providing high-quality judgments, and, most importantly, can successfully distinguish between good and bad annotations. Although the average expert ratings tend to be slightly lower than the community ratings, especially for the lower-rated annotations, the data hints that with a proper quality threshold, the community rating alone may be able to serve as a quality control for the results of the annotation process. In addition, it suggests that the community rating can be used to separate examples, which can be released immediately as educational content without re-annotation from those, which require re-annotation.

The community-based re-annotation produces a more coherent outcome by significantly influencing weaker annotations

The decrease in the standard deviation from 1.38 to 0.57 (Table 1) in the experimental group suggests that the quality of the annotation results becomes more uniform after community feedback and re-annotation. On the contrary, we do not see this effect in the control group. Figure 6 helps us to understand the mechanism of this process. As the

Table 3: Students' perceived skillfulness and knowledge test scores of the control and experimental groups

Perceived skillfulness	*Control group*	*Experimental group*
Bad	7	4
Very bad	8	11

figure indicates, the student group can be split into *stronger* and *weaker* students. Here and below, *stronger students* are those who receive expert ratings above 2.5 on their original annotations, and *weaker students* are those who receive expert ratings below 2.5 on their original annotations. The results show that community feedback affects stronger and weaker students differently. The left side (experimental group subjects) of the Figure 6 shows, stronger students (numbered e10–e15) produced very good original annotations, which were only marginally improved after re-annotation. In contrast, annotations by weaker students were impacted rather dramatically by the community feedback and re-annotation. The rated quality of annotations produced by weaker students more than doubled, growing from 1.40 to 3.40. The growth was also significant ($p < 0.01$).

Objective and subjective skill measures do not predict the quality of annotations
As we found in our study, annotations produced by strong students (with a quality rating over 2.5) received little improvement after the community peer review. Moreover, the original quality of the strong annotations was sufficient to be used as learning material. This suggests the idea that there is no need to pass strong annotations through the feedback process, instead using them 'as is' while focusing the scarce community feedback resource on improving weaker annotations. The problem, however, is that the annotation ranking that allowed us to distinguish good from bad annotations, is produced as an outcome of the very annotation process that we seek to avoid. Is there a way we can reliably predict the quality of the annotations before the community annotation process takes place?

In our study, we tried to assess whether either an objective or subjective skill measure taken before the annotation process could be used to predict annotation quality. The results of the knowledge pretest were used as an objective measure, and the students' *perceived skillfulness* was used as a subjective measure. Information about student-perceived skillfulness was collected (among other data) through the questionnaire filled in before the annotation process began. Two of the questions asked the students to report their experience with Java and to rate their programming skills using 5-point scale (ranging from *very good* to *very bad*). The average experience with Java was 14 months for both groups. The students were rather modest in rating their programming skills (Table 3), yet we were able to split both groups approximately in half by dividing them on their perceived skillfulness.

It could be hypothesised that the students with better pretest scores or students with higher perceived skillfulness will produce better annotations. If at least one of these

Table 4: Re-annotation types and composition statistics (average re-annotation per example)

Type of modification	*(a) new*	*(b) based on comments*	*(c) exactly as comments*	*(d) based on original annotation*	*Total*
Control	4.2			0.8	5
Experimental	1.4	4.6	3.67	2.4	12.07

hypotheses is true, strong annotators can be pre-selected before the start of the annotation process, and the annotations produced by this group may not require any peer reviewing. To assess this hypothesis, we tried to correlate the perceived skillfulness and the pretest scores with the annotation performance. However, no strong correlation was found for both measures. The correlation between pretest and annotation quality was −0.29142, and the correlation between perceived skillfulness and annotation quality was 0.1277. Therefore, we concluded that neither pretest, nor perceived skillfulness (if in the bad/very bad range) can serve as good indicators to predict annotation quality. In brief, the community peer-review is so far the only reliable mechanism to select good quality annotations.

Annotations and comment analyses

In order to investigate the quality of the community comments and how this is associated with the final ratings after re-annotation, the annotations were categorised into four types: (a) completely new annotations; (b) modified ones, based on comments; (c) modified to be exactly the same as comments; and (d) the re-annotation is modified from the original annotation; however, either no comments were available for the given line or the modified annotation bears no clear connection to the comment.

For the control group, there was only type (a) and type (d). As shown in Figure 7 and Table 4, in the control group, 84% of the re-annotation increase in quality came from type (a). As explained above, both groups were able to see everyone's example and annotations, although only the experimental group had access to the community feedback. In other words, subjects from the control group were impacted by the community's work, but not the peer feedback. Observing annotation produced by others may give the students of the control group some new ideas about how to improve their own annotations. It is interesting that the addition of type (a) annotation highly correlated with an increase in the ratings ($r = 0.93$). A more detailed examination of the new annotations demonstrated that they primary fall into two categories: declaration and block statement (Table 5). These two categories are relatively simple in terms of annotating and are mostly context-independent. As a result, they are highly transferable to new examples. At the same time, these annotations are not the most valuable either, so the improvement of rating is rather small. This data suggests that the 'community wisdom' can be transferred even when the students are simply observing each others' work, although leaving out the peer reviewing process means that the community impact is very moderate.

Table 5: The types of new annotations in the control group

	The composition of new annotations	
Category	Declaration	Block statement
Concepts	Class, type, method and variable declaration	Closing bracket functions
Control group	53.33% (8/15)	86.67% (13/15)

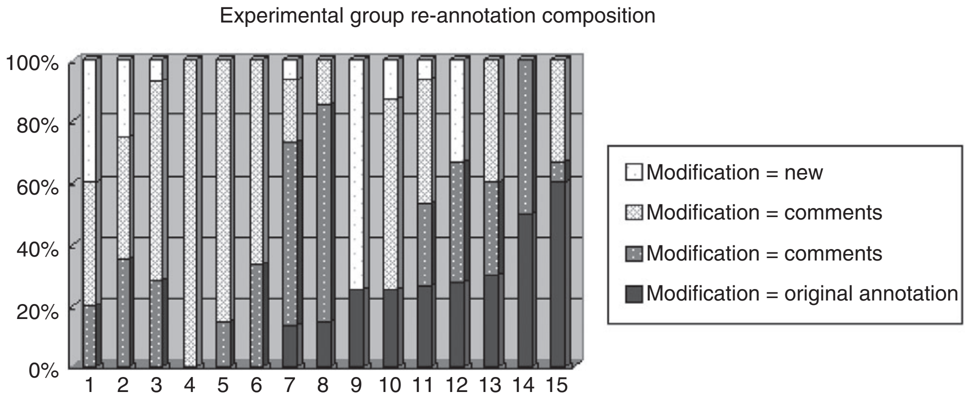


Figure 8: The types of re-annotation in the experimental group

In the experimental group, 68.5% of re-annotations were modified from community comments (types (b) and (c)) leaving only 11.6% for new (type (a)) annotations (Figure 8). The number of type (d) annotation also increased in comparison with the control group, which emphasises that these changes may be also influenced by comments, although indirectly. The number of re-annotations directly influenced by comments (type (b), (c)) is highly correlated with an increase in the ratings ($r = 0.94$). It confirms the role of the impact of the community, comments on the growth of annotation quality and uncovers its mechanism.

Subjective data analysis

Opinions and suggestions on the features of the system were collected through questionnaires at the end of the experiment. The questions attempted to assess both sides of the annotation process and its contribution to students learning. As you can see from Figure 9, 80% of the students considered peer annotation as helpful, and even a larger fraction of students (86.67%) agreed or strongly agreed that working on annotations contributed to their understanding of the subject. The whole process was considered to be a meaningful educational experience by 70% of students. Least positive (although still quite positive in absolute value), the students hesitated to credit the re-annotation process. Only 60% of students specifically credited the additional help of the re-annotation part. This, however, is not surprising, since it corresponds to the fraction

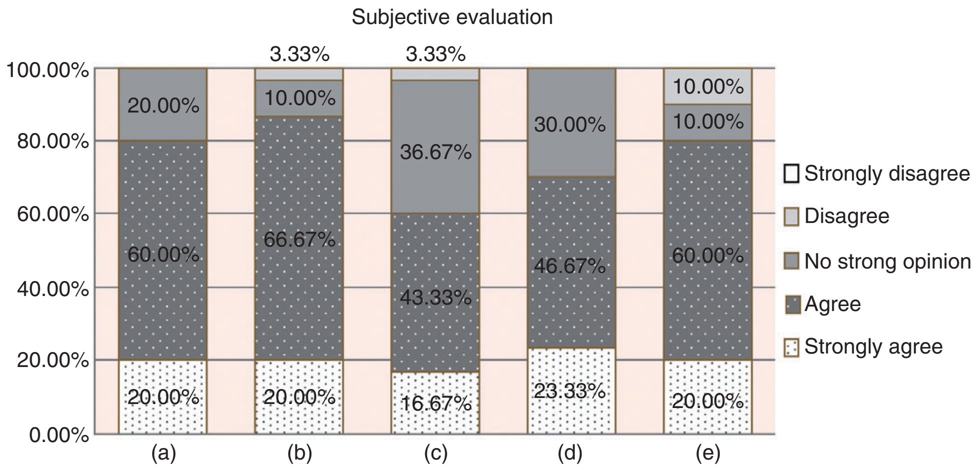


Figure 9: Questions and answers in the subjective evaluation of AnnotEx. (a) Overall, the annotations that my fellow students provided for the examples were helpful. (b) Providing my own annotations contributed to my understanding of the subject. (c) Re-annotating the examples helped me to clear some difficult concepts. (d) Annotating the examples provided me a meaningful experience to practice my programming skills. (e) Annotating the examples was challenging.

of students who significantly improved their work during re-annotation stage. As the analysis above showed, about one-third of the students produced good original annotations and only marginally improved it. For these students, the re-annotation process hardly added any value.

In their free-form feedback, students commented on their experiences with the process and the AnnotEx system. In their comments, the students stressed that it was valuable to them to review others' annotations: '*I was happy with the exercise ... It is interesting to see the different styles that each person used ...*,' '*This is an excellent learning environment ... The AnnotEx system is a great tool in exposing students to how others would view your code explanations*', '*[AnnotEx] gives us better insight on how to annotate more appropriately*'. Students emphasised that the whole process was important not only as a chance to learn the annotation styles, but also as a chance to examine how correct was their understanding of code reflected on their annotations.

Summary and future work

The study reported in this paper evaluated the impact of the community-based peer-reviewing process supported by AnnotEx authoring system on the quality of example annotation and student learning. The results demonstrated that community feedback positively affects both the quality and the quantity of produced content. Peer comments made by the community members provided efficient guidelines for improving annotations and caused a significant increase in quality. Most remarkably, the community feedback affects weaker annotations, resulting in a more coherent annotation quality.

We also discovered that through the peer-rating process, the community is capable of distinguishing good and bad annotations almost as well as the experts did, while other sources such as objective or subjective skill measures fail to predict annotation quality. Students very positively evaluated their experience in the community annotation process: at least 80% of them stated that both reading peer annotations and authoring their own annotations helped them to understand the subject they were working on. While test-based evaluations failed to reliably confirm this part of the subjective feedback and demonstrate any significant knowledge growth, the subjective data shows some evidence that the knowledge was, indeed, improved during the process.

In the future, we want to continue our exploration of the community authoring process and the AnnotEx systems. In addition to exploring alternative settings for the peer-annotation process, we are also interested in exploring whether students are able to create valuable examples, not just to explain the code examples provided by teachers.

Acknowledgements
This material is based upon work supported by the National Science Foundation under Grant No. 0447083.

References

Abad, C. L. (2008). Learning through creating learning objects: experiences with a class project in a distributed systems course. In Proceedings of *13th Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE'2008*. Madrid, Spain (pp. 255–259). ACM New York, NY, USA.

Aleven, V. & Koedinger, K. R. (2002). An effective metacognitive strategy: learning by doing and explaining with a computer-based Cognitive Tutor. *Cognitive Science*, *26*, 2, 147–179.

Brna, P. (1998). Searcing for examples with a programming techniques editor. *Journal of Computing and Information Technology*, *6*, 1, 13–26.

Brusilovsky, P. (2001). WebEx: learning from examples in a programming course. In W. Fowler & J. Hasebrook (Eds), *Proceedings of WebNet'2001, World Conference of the WWW and Internet* (pp. 124–129). Orlando, FL: AACE.

Brusilovsky, P., Grant, N., Hsiao, S., Moore, K. & Sosnovsky, S. (2007). Personalized E-Learning for distance courses in community colleges. In T. Bastiaens & S. Carliner (Eds), *Proceedings of World Conference on E-Learning, E-Learn 2007* (pp. 226–231). Quebec City: AACE.

Brusilovsky, P. & Yudelson, M. (2008). From WebEx to NavEx: interactive access to annotated program examples. *Proceedings of the IEEE*, *96*, 6, 990–999.

Burow, R. & Weber, G. (1996). Example explanation in learning environments. In C. Frasson, G. Gauthier & A. Lesgold (Eds), *Proceedings of Third International Conference on Intelligent Tutoring Systems, ITS-96* (pp. 457–465). Berlin: Springer-Verlag London, UK.

Cafolla, R. (2006). Project MERLOT: bringing peer review to web-based educational resources. *Journal of Technology and Teacher Education*, *14*, 2, 313–323.

Chi M. T. H., Bassok, M., Lewis, M. W., Reimann, P. & Glaser, R. (1989). Self-explanations: how students study and use examples to solve problems. *Cognitive Science*, *13*,145–182.

Cho, K. & Schunn, C. D. (2006). Commenting on writing: typology and perceived helpfulness of comments from novice peer reviewers and subject matter experts. *Written Communication*, *23*, 3, 260–294.

Cho, K., Schunn, C. D. & Wilson, R. W. (2006). Validity and reliability of scaffolded peer assessment of writing from instructor and student perspectives. *Journal of Educational Psychology*, *98*, 4, 891–901.

Davidovic, A., Warren, J. & Trichina, E. (2003). Learning benefits of structural example-based adaptive tutoring systems. *IEEE Transactions on Education*, *46*, 2, 241–251.

Denny, P., Luxton-Reilly, A. & Hamer, J. (2008). Student use of the peerwise system. In Proceedings of *13th Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE'2008*. Madrid, Spain (pp. 73–77). ACM New York, NY, USA.

Fujihara, Y., Ohnishi, H. & Kato, H. (2006). A practice of peer evaluation in ICT education using a report evaluating support system. In T. C. Reeves & S. F. Yamashita (Eds), *Proceedings of World Conference on E-Learning, E-Learn 2006* (pp. 687–694). Honolulu, HI: AACE.

Gómez Albarrán, M. (2005). Teaching and learning of programming: a survey of supporting software tools. *The Computer Journal*, *48*, 2, 130–144.

Gotel, O., Scharff, C. & Wildenberg, A. (2008). Teaching software quality assurance by encouraging student contributions to an open source web-based system for the assessment of programming assignments. In Proceedings of *13th Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE'2008*. Madrid, Spain (pp. 214–218).

Hsiao, I-H & Brusilovsky, P. (2007). Collaborative example authoring system: the value of re-annotation based on community feedback. In T. Bastiaens & S. Carliner (Eds), *Proceedings of World Conference on E-Learning, E-Learn 2007* (pp. 7122–7131). Quebec City: AACE.

Jonassen, D. H. & Reeves, T. C. (1996). Learning with technology: using computers as cognitive tools. In David H. Jonassen (Ed.), *Handbook of Research for Educational Communications and Technology* (pp. 693–719). New York: Mc Millan.

Linn, M. C. (1992). How can hypermedia tools help teach programming. *Learning and Instruction*, *2*, 119–139.

Masters, J., Madhyastha, T. & Shakouri, A. (2008). ExplaNet: a collaborative learning tool and hybrid recommender system for student-authored explanations. *Journal of Interactive Learning Research*, *19*, 1, 51–74.

Pinkwart, N., Aleven, V., Ashley, K. & Lync, C. (2006). Using collaborative filtering in an intelligent tutoring System for legal argumentation. In P. Brusilovsky, J. Dron & J. Kurhila (Eds), *Proceedings of Workshop on the Social Navigation and Community-Based Adaptation Technologies at the 4th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems* (pp. 542–551). Dublin.

Pirolli, P. L. & Anderson, J. R. (1985). The role of learning from examples in the acquisition of recursive programming skills. *Canadian Journal of Psychology*, *39*, 240–272.

Pressley, M., Wood, E., Woloshyn, V. E., Martin, V., King, A. & Menke, D. (1992). Encouraging mindful use of prior knowledge: attempting to construct explanatory answers facilitates learning. *Educational Psychologist*, *27*, 1, 91–109.

Recker, M. M. & Pirolli, P. (1990). *A model of self-explanation strategies of instructional text and examples in the acquisition of programming skills*. Paper presented at the Annual Meeting of the American Educational Research Association, Boston, MA, April 16–20, 1990.

Robinson, R. (2001). Calibrated peer review: an application to increase student reading and writing skills. *The American Biology Teacher*, *63*, 7, 474–480.

Rucker, M. L. & Thomson, S. (2003). Assessing student learning outcomes: an investigation of the relationship among feedback measures. *College Student Journal*, *37*, September, 400–404.

Sitthiworachart, J. & Joy, M. (2004). Effective peer assessment for learning computer programming. In Proceedings of *9th Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE'2004*. Leeds, UK (pp. 122–126).

van den Berg, I., Admiraal, W. & Pilot, A. (2006). Peer assessment in university teaching: evaluating seven course designs. *Assessment & Evaluation in Higher Education*, *31*, 1, 19–36.

Weber, G. (1996). Individual selection of examples in an intelligent learning environment. *Journal of Artificial Intelligence in Education*, *7*, 1, 3–31.

Yudelson, M. & Brusilovsky, P. (2005). NavEx: providing navigation support for adaptive browsing of annotated code examples. In C.-K. Looi, G. McCalla, B. Bredeweg & J. Breuker (Eds), *Proceedings of 12th International Conference on Artificial Intelligence in Education, AI-Ed'2005* (pp. 710–717). Amsterdam: IOS Press Amsterdam, The Netherlands.