Adaptation in the Context of Explanatory Visualization

Tomasz D. Loboda and Peter Brusilovsky

School of Information Sciences
University of Pittsburgh
135 North Bellefield Avenue Pittsburgh, PA 15260, USA

Abstract. Explanatory visualization is a promising approach that has been used in many tutoring systems. This paper presents an attempt to assess the value of adaptation in the context of explanatory visualization. It shows that a system employing a user model and tracking users' progress gives students an opportunity to interact with larger amount of material in the same amount of time.

Keywords: adaptation, adaptive explanatory visualization, evaluation, systems comparison, user study, working memory.

1 Introduction

For more than thirty years, many researchers have explored the use of computers in education. Many of the endeavors focused on developing various kinds of educational tools to facilitate teaching and learning programming. Intelligent tutoring systems (ITS) for programming and program visualization tools [1,2,9,15] were amongst the earliest examples of such applications. ITSs have employed artificial intelligence techniques to model the state and dynamics of the student's programming skills and provide them with individualized guidance in the problem-solving process. In contrast, program visualization systems have supported exploration approach instead of tutoring. They attempted to scaffold students' own intelligence by helping them to better understand the behavior of complex programming constructs. For over ten years those two directions evolved independently. At the end of the 1980s, the pioneer work of Reiser et al. [13] has initiated a new trend in this field of research. An increasing number of researchers have attempted to build systems combining both tutoring and visualization components. This translation from classic ITSs to intelligent learning environments seeks to combine both guided tutoring and learning by exploration.

Many intelligent and adaptive systems for programming, that include both tutoring and visualization functionalities [10,12,17,19], have been created to date. However, in most cases those functionalities are independent and do not affect or reinforce each other. One potential approach to achieve true integration of tutoring and visualization making it more than a sum of its parts is adaptive visualization, originally suggested in [3]. The idea of adaptive visualization is to apply the model of student knowledge maintained by the ITS component to produce visualization adapted to the current level of student's knowledge. It has been argued, that the properly adapted visualization can help to focus student's attention on the least understood programming constructs and thus improve learning outcomes [4].

While being implemented in several systems, the benefits of adaptive visualization have yet to been properly evaluated. To date, some implementation attempts show no evaluation [3], employ a simulated study [4], or report a classroom study with no control group [5]. This paper presents our attempt to advance the adaptive visualization research agenda by exploring the added value of this technology in a controlled study.

The reminder of this paper is structured as follows. Section 2 introduces the system used in the evaluation. Section 3 provides details of the experiment. Section 4 presents the results. Section 5 provides discussion on the findings. A summary and discussion of future plans conclude the paper.

2 The Object of Visualization

cWADEIn[1] [6] is a Web-based tool for students in the introductory C programming language oriented courses[2]. The system addresses the problem of explaining the evaluation of expressions (excluding pointer arithmetics). This problem is both relatively complicated and rarely addressed by visualization tools. cWADEIn supports twenty four operators covering simple arithmetic, comparison, increment/decrement, logic, and assignment operations. The system tracks the progress of the student and uses adaptive visualization and adaptive textual explanations.

In cWADEIn, a visualization of a single operation can have one or more of the following five stages (examples given in parentheses):

- Reading a variable (`A % 3`)
- Production of the value (`0 || -4`)
- Writing a variable (`A = 10`)
- Pre inc/dec (`++A`)
- Post inc/dec (`A--`)

Each stage includes several steps, many of which are animated. The system adapts the speed of animations to the progress the student has done so far with the operator in question. The more progress the higher the pace of animations. Eventually, animations get collapsed into single-step actions.

Some visualizations can be difficult to understand when presented on their own. To address that problem, cWADEIn uses natural language explanations associated with most of the visual events. Each explanation is constructed from one or

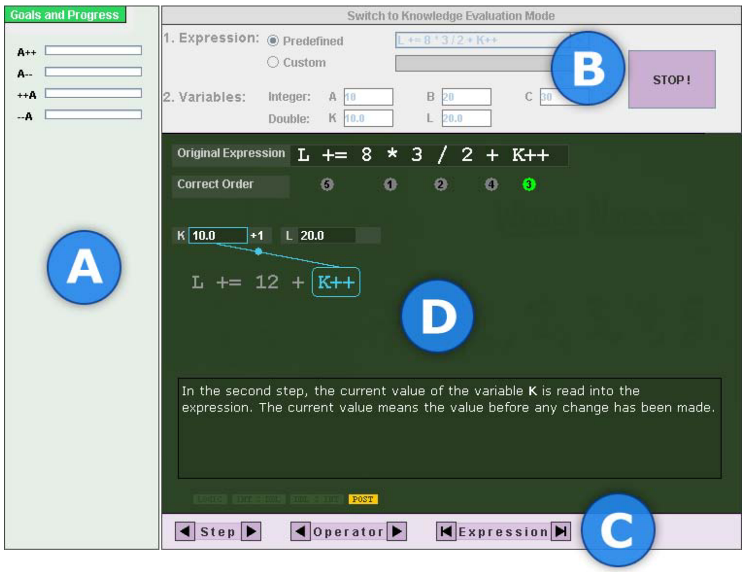
[1] The previous name of the system was WADEIn II. It has been renamed to cWADEIn since a new version, named jWADEIn and supporting the evaluation of expressions in the Java programming language, has been developed.

[2] E.g. *Data Structures and Programming Techniques* course offered by the School of Information Sciences at the University of Pittsburgh.

more fragments of text. Each fragment addresses a different idea. The system decides which fragments to present based on its knowledge of the student's progress.

cWADEIn models two types of concepts: explicit and implicit. The system visualizes the progress the student makes with explicit concepts. The progress made with implicit concepts is tracked, but not visualized. All operators are modeled as explicit concepts. Implicit concepts include (1) reading a variable, (2) implicit casting, and (3) logical value representation.

The user interface of cWADEIn (Figure 1) is divided into four areas: Goals and Progress (A), Settings (B), Navigation (C), and Blackboard (D). Goals and Progress area contains a list of explicit concepts, along with progress indicators (skillometers) which allow users to monitor their progress. The Settings area allows user to select the expression to be evaluated and set the initial values of variables. The Navigation area allows users to go through the evaluation process on a step-by-step or operator-by-operator basis, either forward or backward. Finally, the Blackboard area is where the evaluation takes place. All visualizations and explanations are presented there.



1. Expression: [✓ Predefined] Custom L += 8 * 3 / 2 + K++

2. Variables: Integer: A 10 B 20 C 30 Double: K 10.0 L 20.0

Original Expression L += 8 * 3 / 2 + K++

Correct Order 5 1 2 4 3

K 10.0 +1 L 20.0

L += 12 + K++

In the second step, the current value of the variable **K** is read into the expression. The current value means the value before any change has been made.

Fig. 1. The user interface of the cWADEIn system is divided into four areas: Goals and Progress (A), Settings (B), Navigation (C), and Blackboard (D)

The system features two mutually exclusive modes: Exploration and Knowledge Evaluation. In the Exploration mode the student can step-through the evaluation, observe the visualizations, and read the associated explanations. In the Knowledge Evaluation mode the student is asked questions at every step of the evaluation – starting with a question about the order of evaluation, checking their knowledge of the operators precedence.

3 Experiment

3.1 The System

For the purpose of the experiment, cWADEIn could be launched with adaptive mechanisms enabled or disabled. As described in Section 2, the adaptive version attempted to tailor its behavior to the student's progress. The non-adaptive did not modify its behavior. Animations were always played using the same speed and fragments of explanations were never hidden. Additionally, the progress indicators were not showing the student's progress. They were still displayed, but only as a reminder of current learning goal (i.e. concepts to be mastered). Only the Exploration mode was employed in the experiment.

3.2 Subjects

Fifteen subjects were recruited for the cWADEIn study at the University of Pittsburgh. Nine of the subjects were students in different Bachelor's of Science programs; five of the subjects were students in different Bachelor's of Art programs; one of the subjects was in the Masters of Information Science and Telecommunications program. The only graduate student was subsequently excluded from the statistical analysis as an outlier – their gain score was more than three standard deviations below the mean gain score in both of the two experimental trials (i.e. they learnt very little). Cook's distance depicted in Figure 2 shows the influence that the subject would have had on the analysis results (subject 2) and provides support for the exclusion (Cook's D > 0.8). Nine of the subjects were female and the age range of all subjects was 18–25 ($M = 20.5$, $SD = 1.7$).

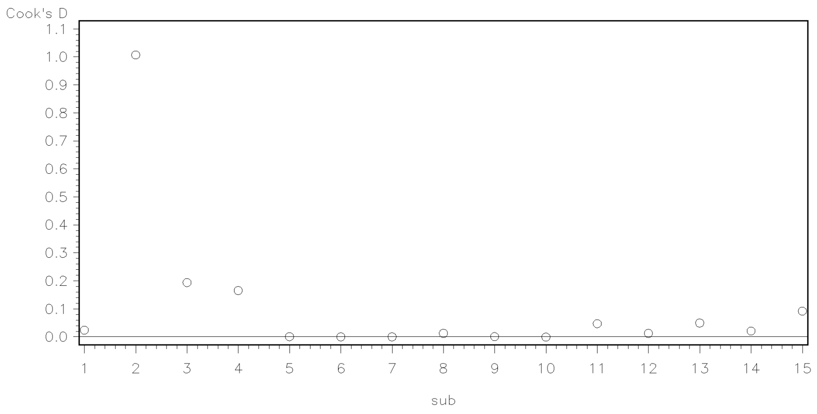


Fig. 2. Cook's distance showing the influence of every subject's scores

The majority of the regular users of the cWADEIn system come from a Data Structures and Programming Techniques course at the School of Information Sciences. The school receives applications from people with different backgrounds and no assumption on the level of technical proficiency are to be made. Because of that the following two recruitment requirements were enforced: (a) none of the subjects could be a student in the Computer Science program and (b) all subjects could have at most *very limited* programming skills (1 on a scale of 0-5). Effectively, eleven subjects never programmed before and three had very limited programming experience.

3.3 Protocol

Figure 3 shows the timeline of the experiment (the anticipated durations of each phase are given in minutes). After filling in the entry questionnaire and performing the MODS task (see Section 3.4), each subject was given an introduction to the study and a short training to minimize the effect of the unfamiliarity with the system. During the training, subjects were asked to attend the evaluation of several expressions to know what to expect with regard to visualization and explanations. Ten expressions with three simple arithmetics operators (+, −, and ∗) were used. Subjects were free to finish the training when they felt ready, but not before they attended the first three expressions and at least one of the more structurally complex ones.

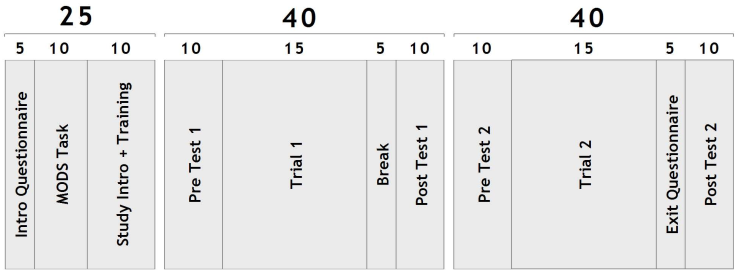


Fig. 3. The timeline of the experiment (time shown in minutes)

After training, subjects were asked to perform two 15-minute learning tials using the cWADEIn system – one with the adaptive version and the other with the non-adaptive version. Initially, the trials were scheduled to be 20-minute long, but after a few pilot subjects we decided to cut them 5 minutes shorter. Each trial was framed as preparation for an in-class test on the C programming language expressions. Subjects were made aware of the dual nature of the problem involving semantics and precedence of the operators. Six subjects completed their first trial using the non-adaptive version, with the other eight starting with the adaptive version, to control for any possible learning effects. A total of twelve operations from the twenty four supported by cWADEIn were used in the experiment. The selected operations were divided into two sets to accommodate the two trials (Table 1). The operator sets were designed to be equal in overall difficulty and orthogonal with respect to knowledge transfer. For instance, assignment operators were cumulated in one set. Distributing them between the two sets would increase the likelihood of inter-trial dependency. Thirty expressions were associated with each operation set. They were available as elements of a drop down list and ordered by difficulty, with the easier ones at the top. Subjects were aware of the ordering but were not forced to follow it.

Table 1. Operators used in each of the set along with the operation group they belong to

Set	Group	Operator
1	comparison	<, >=, !=
	modulo	%
	increment	++A, A−−
2	parenthesis	()
	assignment	=, +=, *=
	logical	||, &&

Prior to starting each task, subjects were given a pretest. After the task was finished subjects had to take a break of an approximate length of five minutes. That time lag was introduced to control for some portion of recency effect. In the case of the first trial that break was a scheduled break in the experiment. In the case of the second trial the break was in a form of the exit questionnaire. A posttest was administered after the break. The corresponding questions on pretest and posttest checked the understanding of the same operator. The tests were designed not to give away the answers. Both the semantics and the precedence of operators was tested, with a greater emphasis on the former.

Apart from questionnaire and test responses, user interface events and gaze data protocols were collected. The Tobii 1750 [16] remote eye tracker was used. The eye tracker calibration routines were part of the experiment, but constituted only a minor portion of the whole experiment time. The discussion of eye tracking results is beyond the scope of this paper. The whole experiment took between 1.5 and 2 hours. That variation was due to the difference in the time it took subjects to solve the tests and fill out the questionnaires (those were not time constrained) and the fact that subjects could finish both learning tasks when they felt ready (before fifteen minutes passed).

3.4 Working Memory Capacity

The task of understanding evaluation of expression evaluation is symbolic in nature. Working memory capacity may have an impact on performance. To measure subjects' sensitivity to the increase in the memory load we used the modified digit span task (MODS) [7]. This task emphasizes individual differences in working memory and reduces impact of other individual differences, e.g. prior knowledge and usage of compensatory strategies. It was administered right after the entry questionnaire (demographics) and before study introduction and the first pretest.

In each trial of the MODS task, a subject was presented with a string of letters and digits. Their task was to remember the digits for later recall. To suppress subvocal rehearsal subjects were asked to read all characters aloud as they appeared on the screen. A metronome tick sound was used to help time the articulation. Each letter was presented for 0.5 sec. The presentation time for digits was lengthened to 0.91 sec to help to encode them in the memory. The number of letters preceding each digit was three or four (selected randomly). This variation was introduced to mask the position of the next digit.

Each of the subjects started with three digits and went through stimuli with four, five, and six of them, which yields a total of four conditions. All subjects went through four trials per condition, which yields a total of 16 trials (not counting the three training ones). The total length of the stimulus was between 12 and 30 characters.

Each subjects started each trial with a screen showing 30 empty boxes (Figure 4). The stimuli presentation begun after they acknowledged their readiness by clicking the "Ready" button. After the entire stimulus was presented all boxes were cleared and recall prompt was presented. This prompt highlighted boxes previously occupied by digits. The subjects had to provide the digits in the order they were originally presented. Backtracking was not possible. They could skip a digit they didn't remember by using the letter X.

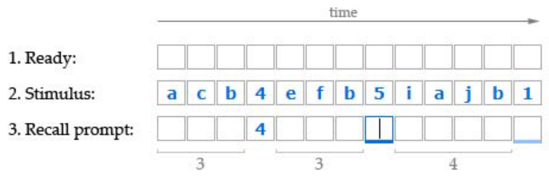


Fig. 4. A sample trial in the MODS task (set size 3)

4 Results

We used an alpha level of .05 for all statistical tests. All p-values reported are two-tailed. All results were obtained using SAS System version 9.1 [14].

4.1 Pretest-Posttest Difference

The response variable in our comparison was the difference between the posttest and pretest scores, that we will be referring to as *gain score* and denote as δ. The two independent variables were the system version and the trial order. Both variables were within-subject and dichotomous.

The system version (non-adaptive and adaptive) was counterbalanced. Subjects were randomly assigned to the two possible orderings. In order to check if that assignment generated equivalent groups we used a paired t-test to compare the pretest scores. We found no significant difference between the mean score of the first group ($M = 2.57$, $SD = 2.24$) and the second group ($M = 4.57$, $SD = 3.13$), $t(13) = .48$, $p = .642$, $\hat{g} = 0.24$.

In an attempt to differentiate subjects with respect to their working memory capacity we calculated an index w for each of them. We did that by averaging the partial recall proportions from the MODS task for set sizes four, five, and six. We excluded set size three due to its small influence (almost no variability in the partial recall proportions; Table 2). We used w as a between-subject covariate ($\rho_{\delta,w} = 0.36$).

Table 2. Partial recall proportions for all set sizes used in the MODS task. Standard deviations express the distinguishing power of each set.

Set size	3	4	5	6	w
Mean (SD)	0.99 (0.02)	0.93 (0.08)	0.77 (0.16)	0.65 (0.15)	0.79 (0.08)

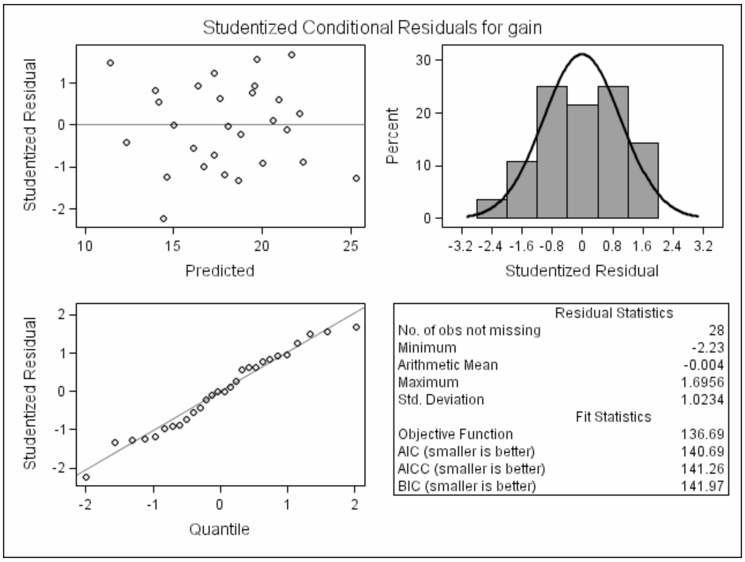
To test the effect of both factors on the gain score we fitted the following linear mixed model

$$y_{ijk} = \mu + \alpha_i + \beta_j + \gamma_k + b_{ik} + e_{ijk} \qquad (1)$$

where μ is the overall mean, α_i is the effect of system i, β_j is the effect of trial j, γ_k is the working memory for subject k, b_{ij} is the random effect of subject j assigned to system i, and e_{ijk} is the random measurement error. We assumed random effects to be normally distributed and independent of each other. We used the variance component structure for the random effect covariance matrix. We used the Kenward-Roger correction for degrees of freedom and standard errors, a method suggested for repeated measures data with small sample sizes [8]. Figure 5 shows studentized conditional residuals diagnostic panel indicating no clear departures from the assumptions of normality. The model above is the most parsimonious one; all higher order terms were not significant.

Working memory index w was a significant covariate, $F(1, 12) = 14.27$, $p = .003$. The trial order explained a significant amount of variability in δ. Subjects achieved higher gain scores on the second trial ($M = 20.43$, $SD = 4.18$) than they did on the first one ($M = 15.50$, $SD = 4.20$), $F(1, 12) = 10.46$, $p = .007$. There was no difference between the mean gain score achieved by subjects with the adaptive version of the system ($M = 17.35$, $SD = 5.05$) as compared to the non-adaptive version ($M = 18.57$, $SD = 4.69$), $F(1, 12) = 1.48$, $p = .247$. If the difference between the two systems existed it might had been masked by the ceiling effect. It is possible that fifteen minutes was still too much for the learning task in the case of our, quite simple domain. Some evidence of ceiling effect is provided by the fact, that eleven out of the total of twenty eight trials were finished before time by subjects themselves.

We also looked at the difference between posttest and pretest scores to assess if cWADEIn helped subjects in improving their understanding of the domain. The results of a paired t-test indicate that they got a significantly higher score



Residual Statistics	
No. of obs not missing	28
Minimum	-2.23
Arithmetic Mean	-0.004
Maximum	1.6956
Std. Deviation	1.0234
Fit Statistics	
Objective Function	136.69
AIC (smaller is better)	140.69
AICC (smaller is better)	141.26
BIC (smaller is better)	141.97

Fig. 5. Model 1 fit assessment. No apparent patterns in the residuals cloud. The distribution of the residuals reasonably well follows the shape of the normal distribution.

on the posttest ($M = 21.54$, $SD = 5.46$) than they did on the pretest ($M = 3.57$, $SD = 2.86$), $t(27) = 19.73$, $p < .001$, $\hat{g} = 8.67$.

4.2 Material Exposition

In addition to investigating the gain score difference, we checked the amount of material subjects were exposed to. For that purpose, we used the interaction logs collected during the experiment. Due to space limitation, we present only one metric of material exposition: the rate of expression evaluations exploration. If we treat the events of exploring the evaluation of an expression as independent, the total number of them will be Poisson distributed. We choose to compare rates instead of total numbers to control for the total session time.

We fitted the following generalized linear mixed model

$$\log y_{ijk} = \mu + \alpha_i + \beta_j + \gamma_k + \log \tau_{jk} + b_{ik} + e_{ijk} \tag{2}$$

where μ is the overall mean, α_i is the effect of system i, β_j is the effect of trial j, γ_k is working memory for subject k, τ_{jk} is the session time for subject k in trial j and entered the model as an offset variable, b_{ij} is the random effect of subject j assigned to system i, and e_{ijk} is the random measurement error. We assumed random effects to be normally distributed and independent of each other. We used the variance component structure for the random effect covariance matrix. We used Kenward-Roger degrees of freedom approximation. Studentized conditional residual plots showed a good fit of the model. The variance of the Pearson residuals was 0.71 indicating no problems with overdispersion [11], a common problem with count data. Again, that is the most parsimonious model; all higher order terms were not significant.

Subjects working with the adaptive version of the system explored expressions with a rate significantly higher ($M = 1.94$ per min., $SD = .46$) as compared to the non-adaptive version ($M = 1.53$ per min., $SD = .68$), $F(1, 24) = 7.65$, $p = .010$. Subjects were also exploring expressions significantly faster in the second trial ($M = 2.03$ per min., $SD = .69$) than they did in the first one ($M = 1.44$ per min., $SD = .33$), $F(1, 24) = 13.66$, $p = .001$. The effect of working memory index w was not reliable, $F(1, 14.21) = 3.31$, $p = .090$.

5 Discussion

As we have shown above, the rate of expression evaluations explored has a potential of telling the difference between a non-adaptive and an adaptive version of a system. When working with the adaptive version of cWADEIn, subjects were able to explore more expressions. Other studies have also demonstrated that adaptive systems could cause a significant increase of the volume of learning content explored by students [19]. Looked at in another way, the adaptive version has a potential of allowing for a material exposition comparable with a non-adaptive version, but in a shorter amount of time.

However, we did not find the two versions of the system different with respect to the gain score. If we treat a performance on a task to be positively correlated with the amount of work done towards that task we can see that those two results provide some evidence for the existence of the ceiling effect in our study. If the learning sessions were shorter, the adaptive version of the system could have allowed subjects to explore material beyond what could be explored in the non-adaptive version.

Because of that, we were unable to check whether subjects' more efficient work resulted in gaining broader or stronger knowledge. Resolving this problem will guide our future work.

6 Conclusions

We presented an evaluation of the adaptation features of the cWADEIn system that supports students in learning about expression evaluation in the C programming language. We did so by comparing it to another version of the system, deprived of the adaptive capabilities. We have found the two versions indistinguishable with respect to the pretest-posttest difference. We argue that this may be an artifact of a ceiling effect.

We have found that the adaptive version of the system allowed subjects to explore expressions significantly faster. That shows that adaptation has the potential of improving material exposition. It can also provide time savings for students by allowing them to explore the domain of interest in shorter time.

Our immediate future plans related to this project include the analysis of the gaze protocol and a more in-depth analysis of the interaction logs.

Acknowledgments. This material is based upon work supported by the National Science Foundation under Grant No. 0426021. The authors would like to thank (in alphabetical order) Roman Bednarik, Angela Brunstein, Rosta Farzan, Christian Shunn, and Sergey Sosnovsky for their valuable input. Sergey was our domain expert.

References

1. Baecker, R.: Two Systems which Produce Animated Representation of the Execution of Computer Programs. ACM SIGCSE Bulletin 7(1), 158–167 (1975)
2. Barr, A., Beard, M., Atkinson, R.C.: The Computer as Tutorial Laboratory: The Stanford BIP Project. International Journal on the Man-Machine Studies 8(5), 567–596 (1976)
3. Brusilovsky, P.: Intelligent Tutor, Environment and Manual for Introductory Programming. Educational and Training Technology International 29(1), 26–34 (1992)
4. Brusilovsky, P.: Explanatory Visualization in an Educational Programming Environment: Connecting Examples with General Knowledge. In: Blumenthal, B., Gornostaev, J., Unger, C. (eds.) EWHCI 1994. LNCS, vol. 876, pp. 202–212. Springer, Heidelberg (1994)
5. Brusilovsky, P., Su, H.-D.: Adaptive Visualization Component of a Distributed Web-based Adaptive Educational System. In: Cerri, S.A., Gouardéres, G., Paraguaçu, F. (eds.) ITS 2002. LNCS, vol. 2363, pp. 229–238. Springer, Heidelberg (2002)
6. Brusilovsky, P., Loboda, T.D.: WADEIn II: A Case for Adaptive Explanatory Visualization. In: 11th Annual Conference on Innovation Technology in Computer Science Education (ITiCSE), Bologna, Italy, pp. 48–52 (2006)
7. Daily, L.Z., Lovett, M.C., Reder, L.M.: Modeling Individual Differences in Working Memory Performance: A Source Activation Account. Cognitive Science: A Multidisciplinary Journal 25(3), 315–353 (2001)
8. Kenward, M.G., Roger, J.H.: Small Sample Inference for Fixed Effects from Restricted Maximum Likelihood. Biometrics 53, 983–997 (1997)
9. Koffman, E.B., Blount, S.E.: Artificial Intelligence and Automatic Programming in CAI. Artificial Intelligence 6, 215–234 (1975)
10. Kumar, A.: Generation of Problems, Answers, Grade and Feedback – Case Study of a Fully Automated Tutor. ACM Journal on Educational Resources n Computing 5(3), Article No. 3 (2005)
11. Littell, R.C., Milliken, G.A., Stroup, W.W., Wolfinger, R.D., Schabenberger, O.: SAS for Mixed Models, 2nd edn. SAS Publishing (2006)
12. Peylo, C., Thelen, T., Rollinger, C., Gust, H.: A Web-based intelligent educational system for PROLOG. In: Workshop on Adaptive and Intelligent Web-based Education Systems at 5th International Conference on Intelligent Tutoring Systems (ITS), Montreal, Canada (2001)
13. Reiser, B.J., Ranney, M., Lovett, M.C., Kimberg, D.Y.: Facilitating student's reasoning with causal explanations and visual representations. In: 4th International Conference on Artificial Intelligence and Education (AIED), pp. 228–235 (1989)
14. SAS Institute Inc. SAS 9.1.3 Help and Documentation (2007)
15. Shapiro, S.C., Witmer, D.P.: Interactive Visual Simulation for Beginning Programming Students. ACM SIGCSE Bulletin 6(1), 11–14 (1974)
16. Tobii, `http://www.tobii.com`

17. Vanneste, P., Olive, H.: Towards an intelligent environment to learn programming (in Pascal). In: CALISCE 1991, Lausanne, pp. 401–408 (1991)
18. Venables, W.N., Ripley, B.D.: Modern Applied Statistics with S. Springer, Heidelberg (2002)
19. Weber, G., Brusilovsky, P.: ELM-ART: An adaptive versatile system for Web-based instruction. International Journal of Artificial Intelligence in Education 12(4), 351–384 (2001)