The Cost of Authoring with a Knowledge Layer

Lichao LI, Judy KAY
School of Information Technologies,
The University of Sydney, NSW 2006, Australia
{lli1, judy}@it.usyd.edu.au

Abstract. We have recently added a knowledge layer to a learning tool, Assess, developed to help students develop programming skills. This paper describes the authors' view of the new Assess and an evaluation of the authoring interface. We discuss the advantages of adding a knowledge layer and report the study of the effort of authoring.

1 Introduction

Many adaptive hypermedia (AH) [1] educational systems exist today, such as ELM-ART [2], WebTutor [3], INSPIRE [4] and TANGOW [5]. An advantage of an adaptive teaching system over others is that if offers a personalized learning environment and/or learning experience. Moreover, such systems are relatively simple to construct compared to traditional Intelligent Tutoring Systems. However, from an authoring perspective, an efficient AH is not at all simple to design [6]. There has been considerable work towards the support of powerful and flexible authoring for authors of AH, such as the LAOS [6] and MetaLinks [7].

In this paper, we present Assess [8], a programming education system that facilitates student self-evaluation and provide adapted learner feedback. Our research is primarily concerned with the addition of a knowledge layer to the system so that students' knowledge can be modeled and more intelligent feedback on learner's progress can be provided. In this paper, we discuss in detail the authoring of teaching material in the system. For more details of the full system and evaluations, from a student perspective, see [8].

The following section provides a brief overview of Assess from the student perspective. Section 3 describes the process of authoring of exercises in the system. Section 4 describes evaluations and Section 5 is conclusion.

2 Assess: A Self-evaluation Tool

The work reported here was conducted in the context of teaching/learning programming in C and Java in our undergraduate subject, Software Development Methods I, which is a second year programming course that aims to teach programming in C in a UNIX environment.

In Assess, exercises for self-evaluation take the form of tasks. Each task has a programming problem that students need to answer. The system allows students to provide solutions to these problems and self-evaluate their solutions against a set of marking criteria provided by authors. More importantly, it provides students with example solutions to assess.

They are normally not the 'perfect' solutions, but they do provide students some ideas to think about and evaluate. These example solutions have been pre-assessed by task authors. The student is meant to evaluate them as she did for her own solutions. Her assessment is then compared to the tutor's assessment of the same example solution. The comparison gives the student feedback on: how the teacher marked the example, the difference between the teacher's assessment and the student's assessment and why the teacher assessed it that way. This information is used to update the system's belief of students' learning progress, which can be viewed in the student's user profile. Our system's current approach to student assessment is quite unique, different from other existing systems, such as CourseMaster [9] and InSTEP [10], both of which automatically evaluate students' codes and provide instant feedback. The whole design was intended to help students reflect on code, taking the perspective defined by the criteria.

The process of student self-evaluation is shown in Figure 1. This system was used in 2004 in the Software Development Method I course. We recognised that there was a lack of knowledge representation in the system, thereby preventing intelligent and informative feedback to students. To overcome this limitation, we added a knowledge layer to Assess (See Figure 2), so that all the ad-hoc elements were replaced by a systematic knowledge layer that defines the learning objectives, user model components and domain ontology. To accommodate this new layer, the process of task authoring in Assess was changed considerably. In this paper, we present the new task creation process and its evaluation.

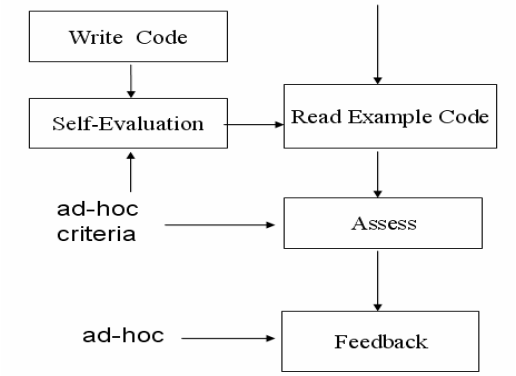

Figure 1 The Old Student Self-evaluation Process

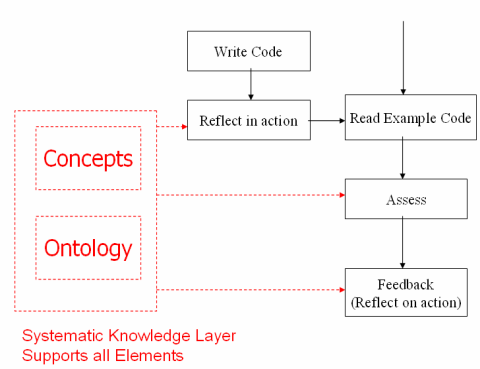

Figure 2 The New Student Self-evaluation Process

3 Task Creation with a Knowledge Layer

With the inclusion of a knowledge layer, we need to set up the learning objectives in the system. These objectives are concepts to be taught/learned, defined by teachers according to the learning outcome requirements. They are what the system attempts to teach and are used to specify teaching/learning goals of individual tasks. Moreover, they also define the knowledge that the system tries to model each student as knowing or not in the student's learner model. The addition and removal of the objectives is achieved using a web interface, as shown in Figure 3. After defining these objectives, there are three stages to follow to author a task in Assess:

Stage One. Create task statement and associate learning concepts;
Stage Two. Edit marking criteria and;
Stage Three. Create example solutions.

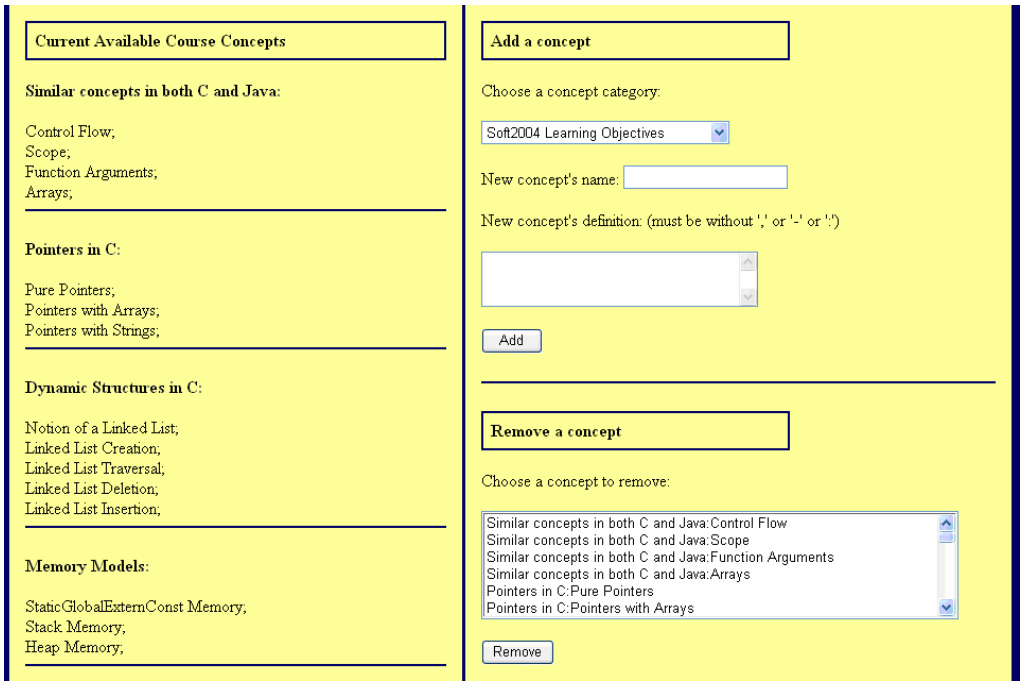

Figure 3 Concept Editing Interface (The left side of the interface shows the concepts that already exist in the system. They are organised into different predefined categories. The top right part of the screen allows new concepts to be added. Each new concept must be selected from a predefined category. The bottom right part of the interface allows concepts to be removed from the system.)

In the first stage (see Figure 4), an author provides the problem statement (i.e. the task that student needs to solve) and, optionally, a skeleton answer to help the student. They also need to indicate the difficulty of the problem and specify whether to force the student to save and assess their own answer before viewing the example solutions. More importantly, the author must relate the task to relevant learning objectives. Each chosen learning objective represents a teaching/learning goal of the task. A task must have at least one goal associated with it to denote what it aims to teach.

The second stage in task creation is the editing of marking schemes. See Figure 5. A default set of marking schemes is automatically generated based on the teaching goals for the task. As a result, when students assess their own solutions and our example solutions with them, they should concentrate on the task's learning goals. However, as the automatically generated marking schemes' criteria are not always meaningful, we allow them to be edited.

Stage Three involves creating, editing and assessing example solutions. A new example can be created at this stage. The author can edit the example solution's marking schemes. All example solutions of a task share a same set of marking schemes that were created in Stage One and Two. However, it is possible for each solution to have additional marking schemes. This means that each task has core teaching goals but each example solution can have additional elements. Figure 6 exhibits how additional marking schemes can be created and removed for example solutions. However, the task's main learning concepts cannot be edited here; as noted above, these are core to the whole task and can only be altered at the task level, i.e. at Stage One and Two.

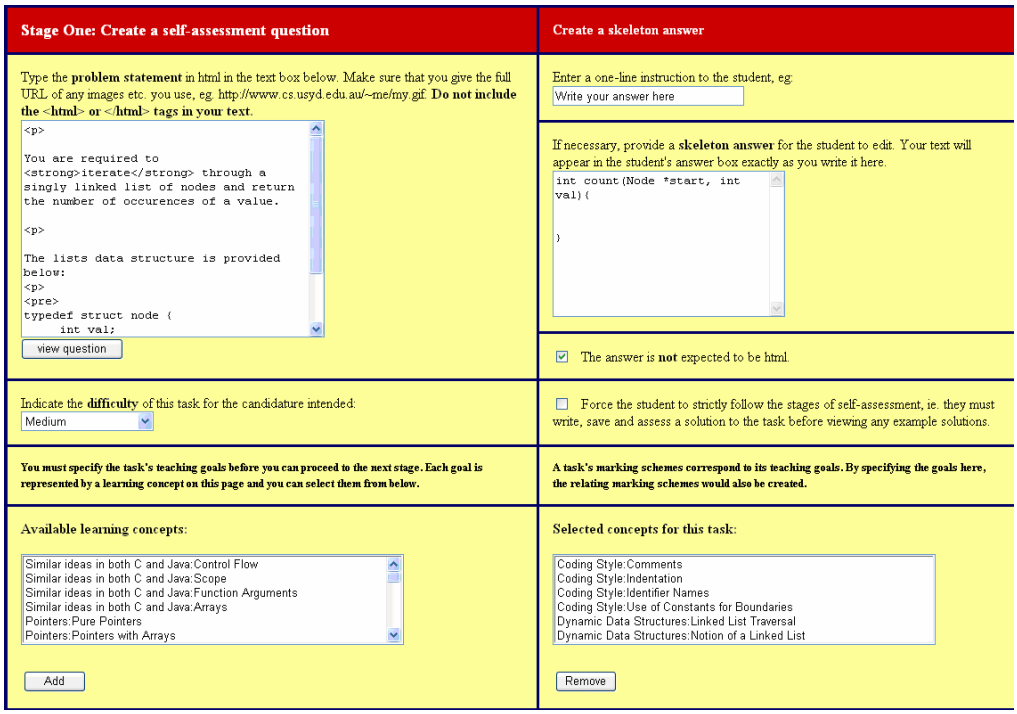

Figure 4 Stage One of the New Task Creation Process with Teaching Goals Selection (The top left part of the interface provides a text box for authors to type the problem statement. To its right, the authors can provide an optional skeleton answer. At the bottom of the interface, authors can select learning goals of the task. The bottom left box contains all the learning objectives available in the system that have not been associated with the task. The box to its right contains the learning goals of this task. A task cannot have duplicated learning goals.)

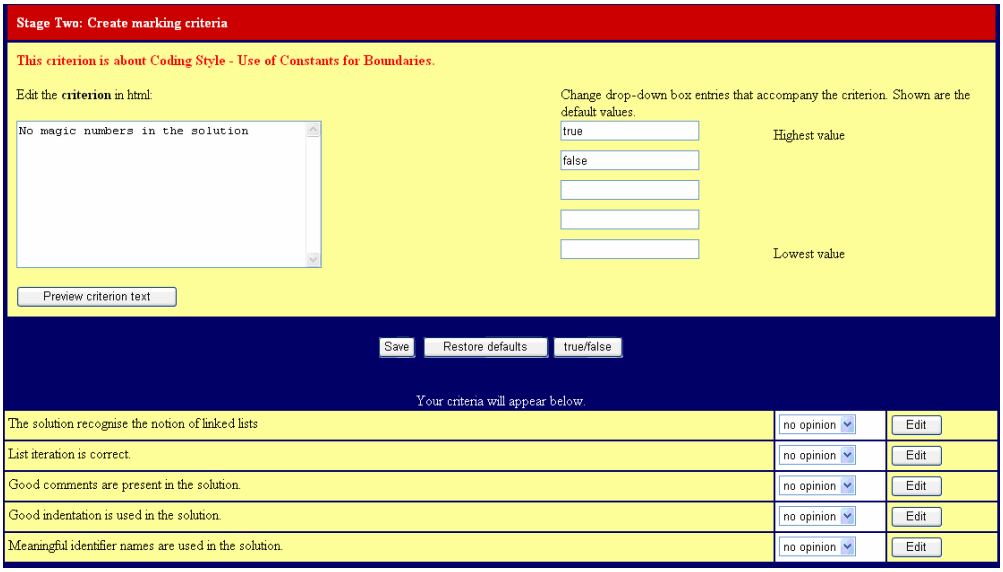

Figure 5 New Task Creation Stage Two, Editing of Marking Schemes (All marking schemes are displayed at the bottom of the page, with their criteria to the left and their marking options to the right in the pop-up menus. When an author *edits* a scheme, the scheme will be taken off from the bottom section and appear at the top section of the interface. Its criterion will be displayed in the text box at the left and its marking options are displayed in the text boxes to the right, so they can be edited. Once the author finishes authoring, she can *save* the changes.)

After the marking schemes are updated for an example solution, an author can assess the solution with the complete set of marking schemes and provide an explanation for the assessment (Figure 7). When a student assesses this example solution, her

assessment is compared with the author's assessment. The discrepancy indicates how well she understands the learning concepts associated with the marking schemes, and is recorded in her individual learner model to provide adaptive learning feedback. We can illustrate this with an example: A concept, *Flow of Control – While Loop*, is selected to be a learning objective of a task and its marking scheme is created automatically. The marking scheme's criterion is "The while loop used in the solution code is correct" and its marking options are *true* and *false*. When a student assesses an example solution with this scheme, she thinks the loop used in the code is correct but the author of the task thinks otherwise, this shows the student cannot recognise the elements that make up a correct while loop and so does not understand this learning objective yet. This information is recorded in her learner model.

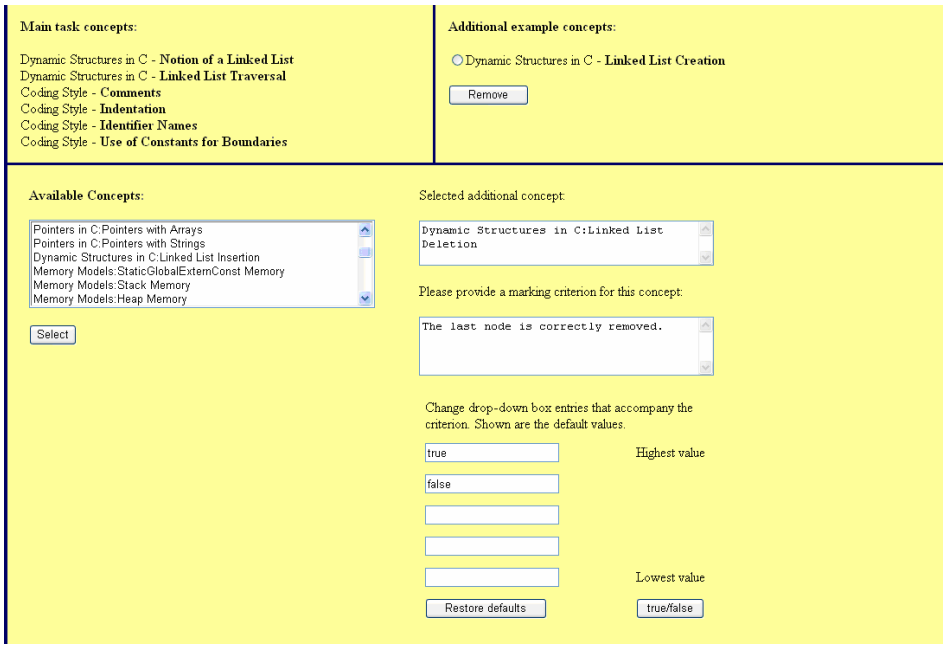

Figure 6 Editing Example Learning Concepts for One Example Solution for One Task (The top left section shows the main learning/teaching objectives of the task. The top right section displays the learning goals of this particular example solution, in this case, *Dynamic Structures in C – Linked List Creation*. They can also be removed from here. The bottom part of the interface allows additional learning goals be added and corresponding marking schemes to be created.)

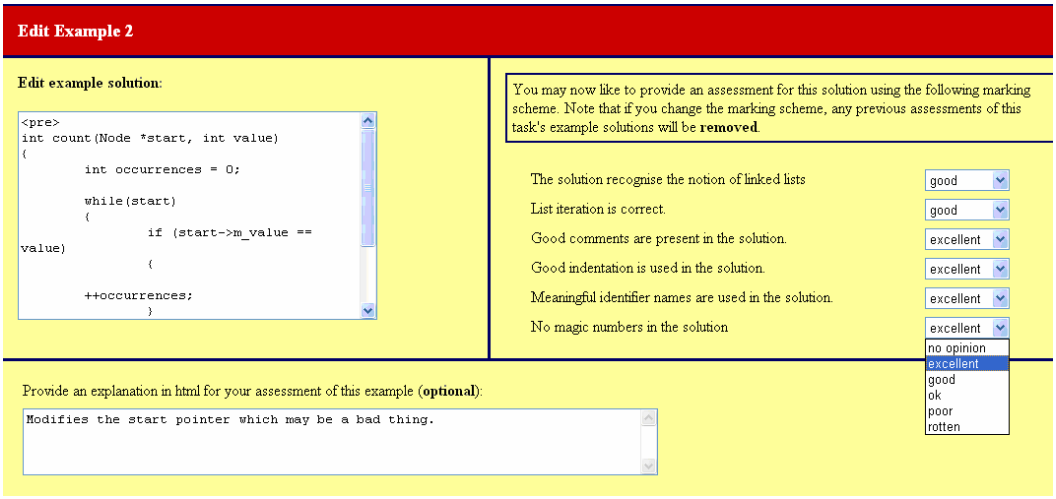

Figure 7 Assessing Example Solutions (The interface allows the example solution in the top left text box to be edited. In the top right section, the author can rate the code against the marking schemes. She can also provide an optional explanation in the text box at the bottom.)

When the author finishes creating and assessing example solutions, and is content with the entire task, she can publish it for students to view.

4 Evaluation

We have conducted a preliminary experiment on the author's perspective of Assess. It was designed to assess the intellectual effort and time involved in the creation of tasks in Assess as well as the usability of interfaces. In particular, we want to ask "What effort and time is involved in entering a new task into Assess?" To answer this question, we evaluated:

- How effective and usable are the interfaces?
- How quickly and accurately first-time authors can create a task?

We selected five[1] participants from our computer science honours and fourth year students to take part in this user trial. They were from different backgrounds. Participant 1 was an experienced tutor, but had not tutored Software Development Methods I before. Participants 2 and 3 were tutors of the subject in 2004. Participant 4 has never tutored and Participant 5 was a tutor for only a brief period in 2003. Participant 3 was the teaching assistant for SDM in 2004. All participants were in the top 15% of the class when they completed the course, so they were all familiar with it. Though this is clearly a highly qualified, technically elite group, it represents the class of users qualified to define and enter tasks for the subject.

At the start of a session, we demonstrated Assess from a student's perspective to show how the system works. This allowed participants to get some insight into Assess, but did not bias the experiment by letting them see what they would need to do. We also supplied the participants with materials they need to put into the system. This included a problem statement and two example solutions that were taken from the original Assess system. They were asked to re-create this task in Assess. They were also told the task's and example solutions' teaching goals. We did not ask the participants to create new tasks because creation of teaching materials is always a time consuming process and requires deep understanding of the big picture course goals. We were not trying to determine how people tackle the more intellectually demanding task of choosing a task and providing solutions. We had also pre-typed the teaching materials on a text editor, so participants could simply cut and paste them into the interface. This reflects a typical scenario of Assess task creation where the lecturer has set an exam question, graded student answers and then developed and tested example solutions. In such a case, all the materials made available to participants would have been complete at the time the lecturer added the task to Assess.

For the experiment, participants were asked to create a task in the system with the supplied material and grade the teaching material creation process in terms of its usability separating the intellectual effort of it from the ease of use of the interfaces. Participants were asked to think aloud so we could take note of any difficulty they encountered.

All participants completed the user trial successfully. The intellectual effort required for each stage of the task creation was rated from 1 (minimal effort) to 6 (a lot of effort). Participants' ratings are illustrated in Figure 8, which shows only modest intellectual effort is required. Participant 4 considered Stage One and Two required a lot of effort. Participants tended to consider Stage Two required the highest (or equal highest) level of intellectual effort as creating questions that can properly evaluate students' understanding of a marking criterion's corresponding learning concept is not an easy process. In terms of

[1] As Recommended for think-aloud usability evaluations: How many users to test (Jakob Nielson's Alterbox), http://www.useit.com/alertbox/20040719.html, 22 Oct, 2004

interface usability, most participants have positive opinions. Participant 3 rated the interface in Stage Two not easy to use because the participant thought there was not enough guidance for authors (See Figure 5). Participant 5 rated the interface of Stage One not very usable and suggested that there was too much information presented (See Figure 4). Overall, the new functionalities required by the knowledge layer did not increase the intellectual effort required to create a task greatly or introduce too much complexity to the interfaces.

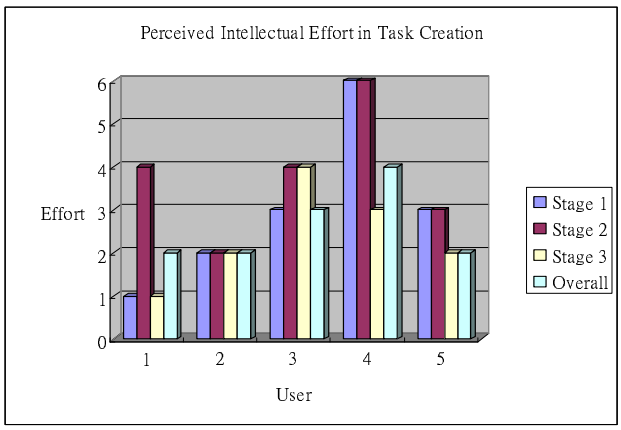

Figure 8 Illustration of Participants' Rating of Intellectual Effort Required for Each Stage of the Task Creation Process (1 = minimal, 6 = a lot of effort)

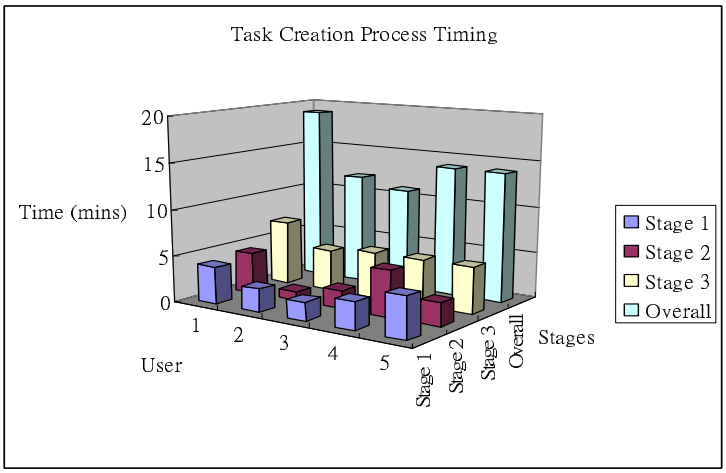

Figure 9 Illustration of Different Participants' Timing of Each Stage of the Task Creation Process

Since this was a think-aloud evaluation, timing data must be interpreted cautiously. However, it gives an indication of the effort and time involved and, taken with the observations of participants, is valuable for establishing indicative times for task creation. The time each participant spent in different stages of the task creation process is listed in Figure 9. All but one participant finished creating the task within 15 minutes. It took the first participant 19 minutes to finish as it was the first user trial run and the participant gave considerable feedback on the experimental procedure during the user trial. Participants 2 and 3 took a shorter time than the others because they were familiar with the course concepts as they had been tutors of the subject as would be the norm for a typical task author. Stage Three slowed for all participants because there were more functionalities involved and more interfaces to view. It is to be noted here that the overall time is not only the sum of time the participants spent on the four stages. It includes also time that

participants spent reading instructions on the tutor home page and task home, which are not part of any specific stage. Of course, it also included any time talking to the observer.

5 Conclusion

We have presented Assess, a student self-evaluation tool, and explained the exercise authoring process of the system with a consistent knowledge layer. We also described a user trial, with results showing that:

1. The intellectual effort required to enter a new task to Assess is modest and the interfaces that allow the creation of a new task are relatively usable;
2. The entire authoring process in Assess takes about 15 minutes, which again shows the effort and time involved, introduced by the knowledge layer, are minimum.

They indicate that the new knowledge layer adds only modest complexity to the task of authoring teaching materials in Assess. At first, one might imagine that the addition of a knowledge layer to a conventional learning tool might require more effort from task authors. From our experience, it seems that the knowledge layer may actually reduce the work of defining a new task and improve the quality of the task as the learning objectives are explicit so helping authors concentrate on what they want students to learn. Moreover, the marking criteria in the new Assess system are automatically generated and they correspond to the teaching/learning objectives of the task. This means that authors do not need to create the questions that test the learning objective for the task and avoids the risk of forgetting to include them. Furthermore, the knowledge layer makes it feasible to provide learners and authors with learner models that capture learning progress, supporting reflection.

References

[1] Brusilovsky, P. (2002) *Adaptive hypermedia*, User Modeling and User Adapted Interaction, Ten Year Anniversary Issue 11 (1/2), 2002. 87-110.

[2] Brusilovsky P., Schwarz E., and Weber G. (1996) *ELM-ART: An intelligent tutoring system on World Wide Web.* In Frasson, C., Gauthier, G., and Lesgold, A., eds., Proceedings of the Third International Conference on Intelligent Tutoring Systems, ITS-96. Berlin: Springer. 261-269.

[3] Pérez T.A., Gutiérrez J. (1996) *WebTutor*, Un sistema Hipermedia Adaptativo para la educación en WWW, Actas del V Congreso Iberoamericano de Inteligencia Artificial, IBERAMIA'96. Cholula, Puebla, MÉXICO, 1996.

[4] Grigoriadou, M., Papanikolaou, K., Kornilakis, H., & Magoulas, G. (2001) *INSPIRE: An INtelligent System for Personalized Instruction in a Remote Environment.* In P. D. Bra, P. Brusilovsky, & A. Kobsa (Eds.), Proceedings of Third workshop on Adaptive Hypertext and Hypermedia, July 14, 2001. Sonthofen, Germany, Technical University Eindhoven. 13-24.

[5] Carro R., Pulido E., Rodríguez P. (1999) *Task-based Adaptive learNer Guidance On the WWW: the TANGOW System*, Second Workshop on Adaptive Systems and User Modeling on the Web, en la Eighth International World Wide Web Conference. Toronto, Canadá. Mayo 1999.

[6] Cristea A. (2004) *Authoring of Adaptive Hypermedia; Adaptive Hypermedia and Learning Environments*. Book chapter to appear in "Advances in Web-based Education: Personalized Learning Environments", Sherry Y. Chen and Dr. George D. Magoulas (eds.). IDEA Publishing group.

[7] Murray T., (2002) *MetaLinks: Authoring and affordances for conceptual and narrative flow in adaptive hyperbooks.* International Journal of Artificial Intelligence in Education, 13 (1).

[8] Li L., Kay J., (2005) *Learner Reflection in Student Self-Assessment,* TR 568, School of Information Technologies, The University of Sydney, ISBN 1864877170.

[9] CourseMaster. (17 May, 2005). CourseMaster [Online]. School of Computer Science & IT, The University of Nottingham, UK., Available: http://www.cs.nott.ac.uk/CourseMaster/cm_com/index.html

[10] Odekirk-Hash, E. (2001). *Providing Automatic Feedback To Novice Programmers.* Unpublished MA, The University of Utah, Utah.