Olsen, J. K., Belenky, D. M., Aleven, V., Rummel, N., Sewall, J., & Ringenberg, M. (2013). Authoring collaborative intelligent tutoring systems. In R. Kumar & J. Kim (Eds.), Proceedings 2nd Workshop on Intelligent Support for Learning in Groups at the 16th International Conference on Artificial Intelligent in Education (pp. 1-10). Memphis, TN: International AIED Society.

Authoring Collaborative Intelligent Tutoring Systems

Jennifer K. Olsen[1], Daniel M. Belenky[1], Vincent Aleven[1], Nikol Rummel[12], Jonathan Sewall[1], and Michael Ringenberg[1],

[1]Human Computer Interaction Institute, Carnegie Mellon University, Pittsburgh, PA, USA
jkolsen@cs.cmu.edu, dbelenky@andrew.cmu.edu,
{aleven,sewall,mringenb}@cs.cmu.edu
[2]Institute of Educational Research, Ruhr-Universität Bochum, Germany
nikol.rummel@rub.de

Abstract. Authoring tools for Intelligent Tutoring System (ITS) have been shown to decrease the amount of time that it takes to develop an ITS. However, most of these tools currently do not extend to collaborative ITSs. In this paper, we illustrate an extension to the Cognitive Tutor Authoring Tools (CTAT) to allow for development of collaborative ITSs that can support a range of collaboration scripts. Authoring tools for collaborative ITSs must be flexible enough to allow for different learning goals and different collaboration scripts. We discuss how two collaboration scripts that we are using in our research on fractions learning are implemented in CTAT. The examples illustrate how CTAT flexibly supports collaborative tutors by running synchronized tutor engines for each student, and how it supports the development of collaborative tutors through the use of multiple behavior graphs that use no programming to develop.

Keywords: Problem solving, collaborative learning, intelligent tutoring system, authoring tools

1 Introduction

Collaborative learning has been shown to be effective for student's knowledge acquisition in some computer-supported settings [9]. However, there is a lack of effective and flexible authoring tools for collaborative learning activities. Authoring tools for Intelligent Tutoring Systems (ITSs) are often geared towards *individual* learning and typically do not have support for the components that make collaborative learning effective [11]. Within Computer Supported Collaborative Learning, collaboration scripts are often used to support collaborative learning, but are often either developed specifically for a particular application [8] [17] or, at best, are provided through a tool that can be used for reuse of the *same* script across multiple subject areas [1], [3], [7], [10], [13-14], [16]. In both approaches, the development tailored for particular domains and learning goals is not straightforward and may not even be feasible. A tool that can be used to flexibly author a range of collaboration scripts for a range of subject areas would bridge this gap. We are working on creating such a tool, by extending an existing ITS authoring tool, the Cognitive Tutor Authoring Tools (CTAT) [2],

so it aids in the development of tutors that integrate a range of collaboration scripts. An earlier attempt to extend CTAT [4] focused on log data, not scripting.

Collaboration scripts are used to structure the tasks and interactions within a group. According to Kollar, Fischer, and Hesse [6], a collaboration script within the educational domain consists of at least five components: the learning objectives, the types of learning activities, the sequencing of the activities, role distribution, and how the script is represented. These components are a way to compare collaboration scripts across platforms, such as face-to-face and computer-supported settings and provide a guideline for the coverage that is needed in authoring tools that wish to support collaborative learning.

There has been work to make collaboration scripts generalizable across learning domains. One example of an authoring tool that can be used across different learning domains is the work done with conversational agents, which monitor a group conversation and can intervene when needed [1], [7]. Although this authoring tool supports multiple learning domains, it supports only the development of collaboration scripts that rely on the use of conversational agents and not a more general class of collaboration scripts. Other tools aim to reuse existing collaboration scripts for new scenarios [3], [10], [13], [16]. These tools are dependent on the learning goals that the existing collaboration script supports instead of customizing the collaboration script for the desired learning goals. On the other hand, the tool, XSS, which is a framework for rapidly developing computer-supported collaboration scripts for new technologies, does support the creation of collaboration scripts to meet specific learning goals [14]. However, XSS does not have support for authoring scripts through an interface, so it may be difficult for users with less programming experience.

The enhancement to CTAT described in this paper allows authoring of collaborative ITSs without programming, and the collaboration script can be specific to the learning goals of the tutor being developed. In this paper we provide collaboration script examples that support cognitive group awareness [4] and sharing of unique information, illustrating the flexibility of the CTAT authoring tool for collaboration. The enhancement to the CTAT system allow students to collaborate through synchronized tutor engines and we will describe how it supports collaborative tutor problems.

2 Collaboration Examples Using CTAT for Collaboration

2.1 An Example of Support for Cognitive Group Awareness

Before we describe how we modified CTAT so it supports authoring of collaborative tutoring, we describe two examples of collaborative tutoring behavior authored with this tool. Specifically, building on our prior work on the Fractions Tutor [12], we are creating a collaborative tutoring system to help elementary students learn fractions. The current prototype includes four conceptual problems and four procedural problems focused on equivalent fractions, each with embedded collaboration scripts. The prototype tutor has been pilot tested with four dyads so far. As students use the tutor, they talk to each other via Skype. The two examples illustrate the types of collabora-

tion scripts can be implemented using the collaborative version of CTAT. In the next section, we extended CTAT to support the collaborative features of these tutors.

The first example features a collaborative fractions problem with a script that supports cognitive group awareness, in which the student is learning conceptual knowledge about equivalent fractions. Cognitive group awareness is the awareness that comes from having information about group members' knowledge, information, or opinions and has been shown to be effective for the collaboration process [5]. This awareness can be supported through tools such as skill meters or by using an interactive interface to display a partner's answers. In our tutor, cognitive group awareness during problem solving is structured as follows: First, the collaborating partners each answer the same question separately. The tutor then displays both partners' answers to promote discussion, and the partners provide a final answer endorsed by both. Each student is given a pair of contrasting attributes (see Figure 1, panel B2) about the fractions. The students are not given feedback on their individual answer but are shown what their partner selected. This allows each student to see their partner's understanding of the fractions. The students are then asked to discuss their answers and decide as a pair what the correct answer will be. Having each student display his or her knowledge of the given fractions before discussing the question together supports the cognitive group awareness. This discussion can lead to a mutual understanding of the fraction attributes, which supports a better understanding of the conceptual knowledge for equivalent fractions. As may be clear, to support cognitive group awareness, the collaborative tutor provides different views of the same problem to the collaborating partners, using two synchronized tutor engines as described below.

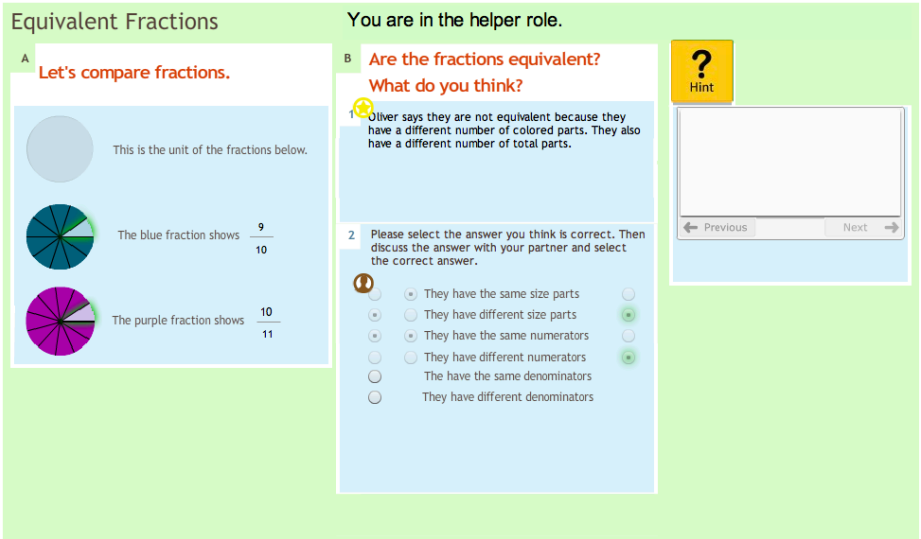

Fig. 1. Panel B2 displays an example of support for cognitive group awareness through the use of multiple radio buttons where each student first selects an answer based on their knowledge before the group makes a group selection that is tutored.

2.2 An Example of Support for Sharing Unique Information

We also used the collaborative version of CTAT to implement a second type of fractions problem, in which students learn how to procedurally evaluate equivalent fractions. As in the previous example, the collaborative tutor provides a different view on the same problem for each collaborating partner, although this time the collaboration is scripted differently for the different learning objective. Specifically, we implemented a script that distributes unique information between the partners and supports the sharing of this information. Students are shown a fraction expressed in symbols (see Figure 2) that their partner does not see as indicated by the star icon. Each partner is also given a circle diagram that they can interact with; their partner can see this diagram but cannot interact with it as indicated by the silhouette icon. One student is first asked to share their fraction with their partner (i.e., by telling their partner about it) while the second student is asked to make this fraction using their circle diagram. The students then switch roles and one student shares their fraction while the other student makes this fraction. Each student sees the feedback from the tutor, so if a student is struggling to correctly make the fraction, their partner, who can see the fraction and the tutor feedback, can provide support and help. By providing each student with different information, the students need to start a dialogue and share. This activity makes the students aware of the fractions as a first step to supporting procedural knowledge for evaluating equivalent fractions.

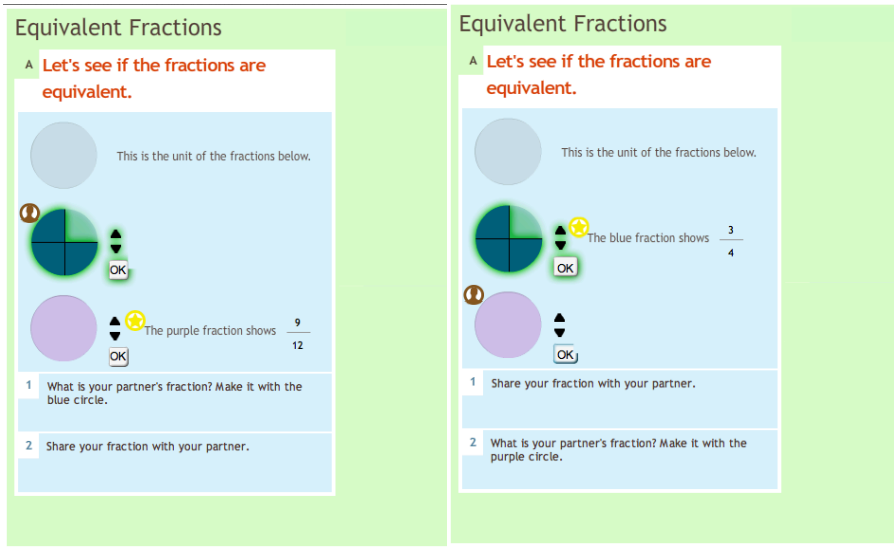

Fig. 2. Panel A displays an example of individual information that needs to be shared between participants. The top blue fraction was made by the student on the left screen using the information shared by the student on the right screen. The purple fraction will be made with the student on the right screen with the information from the student on the left screen.

Both examples illustrate a range of collaborative activities that can be supported using CTAT for collaboration. Kollar, Fischer, and Hesse specify collaboration scripts by focusing on five components [6]. These five attributes provide a guideline for the coverage that is needed in authoring tools. Both examples use different *learning activities* to support the learning goals of the problems. The sharing of unique information uses activities such as sharing and problem solving where as the script that supports cognitive group awareness uses activities such as sharing knowledge and mutual explanations. Within these activities the students are also assigned to very different *roles* where in the unique information scenario they are asked to be a sharer or to be a problem solver and then switch roles. In the support for cognitive group awareness, both students are responsible for sharing their knowledge and then discussing the answers.

3 Authoring Tool Extensions to Support Collaboration

Until recently CTAT only supported tutors for individual use. We focus on one type of tutor that can be authored with CTAT, namely, example-tracing tutors [2]. To develop such a tutor, an author creates two key components, both without programming: a user interface designed specifically for the problem type being tutored (the interface lays out the problem steps) and a generalized behavior graph, which stores all of the acceptable solution paths along with commonly-occurring incorrect steps. The tutor uses the behavior graph to monitor student problem solving and provide guidance to students. Each behavior graphs consists of a set of links that correspond to steps that can be taken in the problem, such as typing in the numerator to a fraction. Some steps (explicitly marked as such) represent *tutor-performed actions*, such as showing a component in the tutor interface that was hidden before. To evaluate student input, the tutor compares the student's problem-solving steps against those in the behavior graph, testing whether the student is on one of the paths in the graph. An author may specify constraints on the order of steps. Behaviorally, example-tracing tutors are similar to other types of ITSs, providing all the key functionality singled out by VanLehn [15] as typical of ITSs.

3.1 Authoring Collaborative Tutors

To expand CTAT so it supports *collaborative example-tracing tutors*, we added the capability to run *multiple synchronized tutor engines,* one for each student in a collaborating group. This set up allows for great flexibility in authoring tutors with embedded collaboration scripts. Specifically, each student in a group has their own behavior graph file and interface file for the given problem. The collaborative version of CTAT synchronizes the tutors so that when one of the collaborating students takes an action, this input is sent to both that student's tutor engine and their partner's tutor engine. Similarly, tutor output is shared among the members of a collaborating group (i.e., all output from the two synched tutor engines, such as hints and feedback, is sent to each student interface separately). One result of this output sharing is that student actions taken on one interface will be "mirrored" on the other interface in the corre-

sponding interface component, together with the associated tutor feedback. As we extended CTAT, we updated the interface tool components to include new actions that better support collaborative learning activities. As an example, we updated the existing components to allow students to view the options of a component without being able to take action on the component, as illustrated in the examples above. We are also adding a highlighting functionality so each student can easily reference a component.

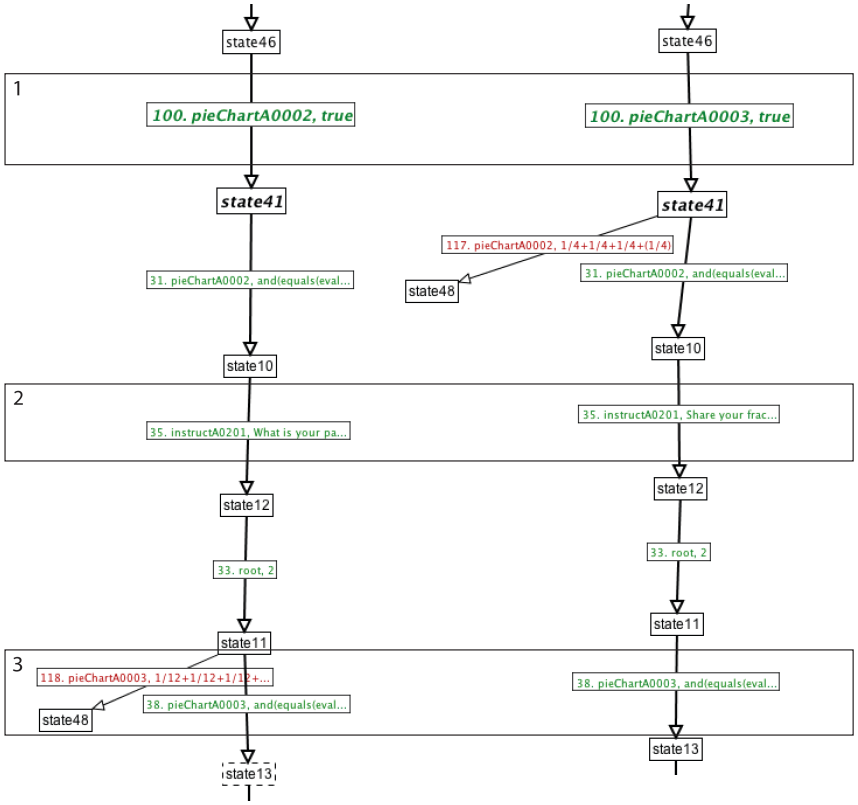

Fig. 3. Excerpts from two behavior graphs working for a single problem. Together both behavior graphs capture the first step to be completed by the students for the problem in Fig 2. Box 1 demonstrates the different locking of components for each student, Box 2 demonstrates different instructions for each student, and Box 3 demonstrates the use of student-performed actions to advance the state of the problem where the partnering student can only take the action.

With these collaborative extensions to CTAT, an author can create tutors that do not differ for the collaborating partners - simply by supplying the same behavior graph and interface for each collaborating partner. The result would be a tutor with which two students interact simultaneously and synchronously while each sitting at their own computer. They would each see the changes that their partner makes. This kind of collaboration may not be terribly useful, however. The power of the approach

comes from being able to craft tutors in which the collaborators have different views on the same problem and have different sets of actions available to them. There are many collaboration activities, such as the jigsaw and the tutee/tutor paradigm, where the benefit of the activity comes from the students having different roles and responsibilities in the problem-solving task. The CTAT authoring tool supports this kind of differentiation, as an author can create separate behavior graphs, one for each student, that display different instructions or capture different student problem-solving actions, dependent on the role of each student, as is used in the cognitive awareness activity. For example, Figure 3 shows, side-by-side, two behavior graphs for the support of unique information example illustrated in Figure 2. These two behavior graphs share common structure, but also differ so as to support different interactions for the two collaborating students.

To show different instructions for each student, an author can use a different tutor-performed action at the corresponding link in the two behavior graphs. An example is shown in Box 2 of Figure 3 where each student receives different directions from the behavior graph at the same point in time. (The label on the link shows the message displayed to the student in truncated form.) Similarly, by providing different behavior graphs for each member, the actions taken by the users can differ. One way to make different sets of actions available to each collaborating partner is by locking certain components in the interface, a different set for each partner. This allows both students to see the action on their respective interfaces while only allowing one student to be able to take the action. An example is shown in Box 1 of Figure 1 where different components (the two circle components, pieChartA0002 and pieChartA0003) are locked for the students through a tutor-performed action, preventing them from interacting with that component. The result of this link in the behavior graph is seen in Figure 2 where the circle that corresponds to the fraction shown on the screen is locked for that student, so that each student can perform his/her own role but not his/her partner's role. Though the student cannot act on the component that is locked, a step to solve the component is in the behavior graph (see Box 3 of Figure 3) so that the problem will not advance until their partner has completed the step. An author can also make the tutor accept different actions from each student by recording different actions in each student's behavior graph. In this case, the student without the action recorded would not have to wait for this action to take place to continue working on the next step of the problem.

Another way to provide different interface elements to the members of each dyad is through an interface file. This file is a SWF file created in Flash. The author can select the components, control their placement on the interface, set basic parameters, and use custom code if necessary. In this way, an author can tailor the interface for the different roles that the collaborators have in the collaboration script that is being supported. An author can also determine what feedback each student receives during the problem by setting an initial tutor feedback parameter for each interface component. This parameter controls whether or not there will be tutor feedback on actions on that component. For example, in the cognitive awareness task in Figure 1, the radio buttons that correspond to the student's *individual* answers provide no feedback, as they serve mainly to support the partners' mutual awareness of each other's reason-

ing. On the other hand, the radio buttons for the *group* answer (on the right in Figure 1) provide correct or incorrect feedback.

The steps to develop a tutor using CTAT consist of developing a user interface, creating a behavior graph, and annotating the behavior graph [2]. Within CTAT, an interface is built using an interface builder and the different components of the interface are adding using a drag-and-drop method. Each component has a set of parameters that can be set allowing the developer to customize the look and feel of the parameter to match their tutor layout. This allows a developer to create a tutor interface without the need for any coding on the part of the developer. Once an interface is created, a behavior graph can be created that maps out the tutor steps through correct and incorrect actions. The behavior graph can be created through demonstrating the actions to be taken on the interface. While having the CTAT Behavior Recorder in demonstration mode, any action that is taken on the interface will be recorded on a behavior graph. By starting at different points in the behavior graph, different branches can be created. This allows a developer to create a behavior graph without the need of programming. After the behavior graph is created through demonstration, the graph can be annotated. Annotation includes adding hints to the links and identifying knowledge components.

To author a collaborative tutor each of the steps to create an individual tutor are followed for each member of the collaboration. Depending on the type of collaboration activities and roles depends on if different tutor interfaces and behavior graphs need to be made for each student in the group or if the same files can be used. If the students are going to be seeing something visually similar then the same tutor interface can be used. If the students are going to take the same actions during the problem, then the same behavior graph can be used. When developing a collaborative tutor, if different interfaces are going to be used and an action that one student takes should be reflected in the view to the other students, then the components that are used for this activity need to be named the same in both interfaces. This is shown in Box 3 of Figure 3 where the same component name is referenced in both behavior graphs. This allows the tutors to reflect an action taken on one interface on the other interface as well. On the other hand, if the author wants particular actions within a tutor interface to be private to one of the collaborating students, one way to do so is to not provide a corresponding interface component in the interface for the other student. The enhancements to CTAT did not add a need for a developer to program to create a tutor. Currently, to test a collaborative tutor, the tutor must be run through the tutoring service. A different browser window can be opened for each student interface so the actions can be seen simultaneously. By assigning each interface and behavior graph to a "student" and then identifying those students as being in a class together, the different tutors are synced and allows communication between the tutors. This assignment can be done through filling out fields in a user interface and no special programming is needed on the part of the author.

4 Discussion, Conclusion and Future Work

Computer-supported collaboration has been shown to be an effective learning paradigm for knowledge acquisition [9], yet most tools that support collaboration do not allow for the authoring of a range of collaboration scripts. Authoring tools for ITSs have been used to address a wide range of domains, but we are not aware of any that support collaboration scripts, other than an early attempt to extend CTAT [4] so it builds collaborative tutors from log data. We extended CTAT so it supports the authoring of collaborative tutors while maintaining its advantages for individual tutoring without programming. With this new version of CTAT, authors can develop collaborative ITSs to meet a range of domains and collaboration scripts. The developer does not need to have a strong background in programming to make a functional tutor.

The extension to CTAT allows for a range of collaboration scripts to be developed. Two examples were provided in this paper, but we also created tailored collaboration scripts to match the learning objectives of six other fractions problems. The flexibility to develop these scripts is because the collaborative version of CTAT was not developed to implement a specific script but to remain open-ended. This design also allows flexibility while developing tutors. As we continue to develop our collaborative fractions tutor, we are taking an iterative approach in which we repeatedly test the collaboration script with students and then refine it to best support the learning goals based on the outcomes of the pilot studies. The collaborative version of CTAT allows for these changes to be made easily in a problem, as behavior graphs are relatively malleable.

Future work will consist of extending CTAT so it can support more than two students in the group. Other future work will be to allow the specifying of groups at runtime instead of needing to specify groups ahead of time. By being able to specify the members of a group at runtime, there would be more flexibility in grouping students in a classroom on any given day. Students would not be dependent on their partner also being there that day. Also to improve the authoring process, functionality is being added to allow an author to have multiple behavior graphs open so they can compare the steps and can copy and paste steps from one graph to another that are similar. The eventual goal of our project is to investigate how best to combine individual and collaborative modes of learning.

Acknowledgments. We thank the Cognitive Tutor Authoring Tools team for their help. This work was supported by Graduate Training Grant # R305B090023 and by Award # R305A120734 both from the US Department of Education (IES).

5 References

1. Adamson, D., & Rosé, C. P.: Coordinating Multi-Dimensional Support in Collaborative Conversational Agents. In: Intelligent Tutoring Systems (pp. 346-351). Springer Berlin Heidelberg (2012)

2. Aleven, V., McLaren, B. M., Sewall, J., & Koedinger, K. R.: A New Paradigm for Intelligent Tutoring Systems: Example-tracing Tutors. International Journal of Artificial Intelligence in Education, 19, 105-154 (2009)
3. Harrer, A., Malzahn, N., & Wichmann, A.: The remote Control Approach-An architecture for Adaptive Scripting Across Collaborative Learning Environments. Journal of Universal Computer Science, 14(1), 148-173 (2008)
4. Harrer, A., McLaren, B. M., Walker, E., Bollen, L., & Sewall, J.: Creating cognitive tutors for collaborative learning: Steps toward realization. User Modeling and User-Adapted Interaction, 16(3-4), 175-209 (2006)
5. Janssen, J., & Bodemer, D.: Coordinated Computer-Supported Collaborative Learning: Awareness and Awareness Tools. Educational Psychologist, 48(1), 40-55 (2013)
6. Kollar, I., Fischer, F., & Hesse, F. W.: Collaboration scripts–a conceptual analysis. Educational Psychology Review, 18(2), 159-185 (2006)
7. Kumar, R., Rosé, C. P., Wang, Y., Joshi, M., & Robinson, A.: Tutorial dialogue as adaptive collaborative learning support. Frontiers in Artificial Intelligence and Applications, 158, 383 (2007)
8. Lesgold, A., Katz, S., Greenberg, L., Hughes, E., & Eggan, G.: Extensions of intelligent tutoring paradigms to support collaborative learning. In S. Dijkstra, H. P. M. Krammer, & J. J. G. van Merrienboer (Eds.), Instructional models in computer-based learning environments. (pp. 291-311). Berlin: Springer-Verlag (1992)
9. Lou, Y., Abrami, P. C., & d'Apollonia, S.: Small group and individual learning with technology: A meta-analysis. Review of educational research, 71(3), 449-521 (2001)
10. Miao, Y., Hoeksema, K., Hoppe, H. U., & Harrer, A.: CSCL Scripts: Modelling Features and Potential Use. In Proceedings of the 2005 conference on Computer support for collaborative learning: learning 2005: the next 10 years! (pp. 423-432). ISLS (2005)
11. Murray, T., Blessing, S., & Ainsworth, S.: Authoring tools for advanced technology learning environments: Toward cost-effective adaptive, interactive and intelligent educational software. Amsterdam: Kluwer Academic Publishers (2003)
12. Rau, M., Aleven, V., Rummel, N., & Rohrbach, S.: Sense Making Alone Doesn't Do It: Fluency Matters Too! ITS Support for Robust Learning with Multiple Representations. In: Intelligent Tutoring Systems, pp. 174-184. Springer Berlin/Heidelberg (2012)
13. Ronen, M., Kohen-Vacs, D., & Raz-Fogel, N.: Adopt & Adapt: Structuring, Sharing and Reusing Asynchronous Collaborative Pedagogy. In Proceedings of the 7th International Conference on Learning Sciences (pp. 599-605). ISLS (2006)
14. Stegmann, K., Streng, S., Halbinger, M., Koch, J., Fischer, F., & Hußmann, H.: eXtremely Simple Scripting (XSS): A framework to speed up the development of computer-supported collaboration scripts. In Proceedings of the 9th International Conference on Computer supported Collaborative Learning-Volume 2 (pp. 195-197). ISLS (2009)
15. VanLehn, K.: The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems. Educational Psychologist, 46, 197-221, (2011)
16. Wecker, C., Stegmann, K., Bernstein, F., Huber, M. J., Kalus, G., Rathmayer, S., Kollar, I., & Fischer, F.: Sustainable script and scaffold development for collaboration on varying web content: the S-COL technological approach. In Proceedings of the 9th international conference on Computer supported collaborative learning-Volume 1 (pp. 512-516). ISLS (2009)
17. Walker, E., Rummel, N., & Koedinger, K.: CTRL: A research framework for providing adaptive collaborative learning support. User Modeling and User-Adapted Interaction: The Journal of Personalization Research (UMUAI), 19(5), 387-431 (2009)