Sustainable Script and Scaffold Development for Collaboration on Varying Web Content: The S-COL Technological Approach

Christof Wecker, Karsten Stegmann, University of Munich, Department of Psychology, Leopoldstr. 13, 80802 Munich,
Email: christof.wecker@psy.lmu.de, karsten.stegmann@psy.lmu.de
Florian Bernstein, Michael J. Huber, Georg Kalus, Sabine Rathmayer, Technical University of Munich,
Department of Informatics, Boltzmannstr. 3, 85748 Garching,
Email: florian.bernstein@in.tum.de, hubernic@in.tum.de, sabine.rathmayer@in.tum.de
Ingo Kollar, Frank Fischer, University of Munich, Department of Psychology, Leopoldstr. 13, 80802 Munich,
Email: ingo.kollar@psy.lmu.de, frank.fischer@psy.lmu.de

Abstract: We present a single solution for the following two problems: (1) to provide just-in-time support for collaborative learning tasks on top of arbitrary web pages (e.g. in order to foster online search competence) and (2) to transfer collaboration scripts implemented on a particular platform to other platforms. S-COL solves both by using fixed browser-side scripts and scaffolds and triggering them by recognizing types of functionally equivalent web pages, and combines this with collaborative web-browsing.

Two seemingly unrelated problems and current attempts at solutions

Collaborative learning tasks involving authentic web content may require support on top of existing web pages that cannot be anticipated by instructional designers. For example, collaborative search tasks for fostering online search competence will lead groups of learners to a broad variety of different web pages. Several approaches to support learners during online search tasks have been developed. One is to provide search engines developed specifically for particular target groups (e.g. children) that closely resemble general search engines, but restrict results to information appropriate for their target group, such as „Yahooigans!“ (now „Yahoo! KIDS“; see Bilal, 2002), or sophisticated search environments enriched with tools that help users organize their search activities, such as „ARTEMIS“ (Wallace et al., 1998). However, these tools are not so much directed at increasing the learners’ competence, but rather at permanently decreasing task difficulty. Furthermore, while rather unsupported search tasks, including WebQuests (Ikpeze & Boyd, 2007), may be insufficient for increasing learners’ online search competence, so-called “guided pages” developed specifically for a particular learning tasks (Berry, 1998) are both less authentic and not transferrable to different learning tasks.

A more promising approach to foster online search competence could be collaborative online search, which has been demonstrated to have beneficial effects on the strategies employed, although not yet on learning (Lazonder, 1995). Beneficial effects on the acquisition of online search competence might be expected from collaboration scripts that structure learners’ interactions in the various phases of a collaborative online search task (cf. Kollar, Fischer & Hesse, 2006). On the cognitive and instructional side, this approach requires the identification of a set of general strategies of (collaborative) online search that can be applied to and accordingly should be prompted during the interaction with a broad variety of thematically diverse web pages. Although we addressed this topic in an ongoing research project, we will not pursue this issue here. On the technical side, support for the application of these strategies in different phases of a collaborative online search task has to be provided in the context of the web page currently displayed, based on some idea of where the learners are in the search process. Tools for changing existing web pages for display in a browser (e.g. enriching them with script prompts) have already been developed (e.g. Greasemonkey, 2009). The main problem, however, is to determine automatically, which script prompts should be displayed along with a specific web page encountered by the learners. Accordingly, a desideratum for supporting online search activities would be a tool to implement collaboration scripts with specific prompts for the different stages of collaborative online search tasks.

In this paper we argue that the technical solution to this problem also constitutes the missing jigsaw piece in a solution for a further problem that has received substantial attention by CSCL researchers in recent years: the re-use of collaboration scripts on different learning platforms. Typically collaboration scripts, e.g. for online discussion, have been implemented and tested as part and parcel of particular learning platforms (e.g. Stegmann, Weinerberger & Fischer, 2007). This approach makes the investigation of the generalizability of findings across learning environments as well as broad use of empirically evaluated collaboration scripts rather difficult. To overcome these problems, several proposals have been made. A prominent approach is the development of a common script language or a unified framework for the description of scripts such as the one developed by the European Research Team “CoSSICLE”. This framework (Kobbe et al., 2007) defines a small number of components and mechanisms of computer-supported collaboration scripts. The components are participants, activities, roles, resources, and groups; the mechanisms comprise task distribution, group formation, and sequencing. On the basis of this descriptive framework a graphical modelling tool for the
development of new collaboration scripts was built (Harrer & Malzahn, 2006). As an output, this modelling tool produces an IMS-LD file, i.e. a file that can be read by all learning platforms that support the IMS Global Learning Consortium Standards. However, currently we are not aware of any free available learning platform that supports this “dialect” of IMS-LD and accordingly can import a script description as an IMS-LD file.

Another, more practitioner-oriented approach is the “manyscripts” approach (Dillenbourg & Hong, 2008): The manyscripts tool offers teachers an environment to adapt a set of specific scripts with regard to their own needs, especially their own learning material. At the moment only the Concept Grid, Argue Graph, and Ice (Dillenbourg & Hong, 2008) are available. The manyscripts environment is a standalone learning platform. A native integration into other learning platforms has not been a goal and is thus not supported yet.

Currently, neither the common script language and graphical modelling approach nor the manyscripts approach are suitable to develop new scripts and implement them in a broad range of different learning environments. A framework that effectively supports the re-usability of technology-based collaboration scripts is currently not available. The transfer of scripts from one learning environment to another is still hampered by the need to adapt and integrate the collaboration script into the new learning platform. Therefore, a solution for using scripts developed and evaluated in one learning environment on other platforms is also still a desideratum. If different learning platforms are considered as varying web content, this problem becomes a special case of the problem of the development of scripts for learning activities involving varying web content. In the next section, we will present a comprehensive solution for this problem.

A comprehensive solution

The basic idea

In an interdisciplinary collaboration involving educational psychologists and computer scientists, our goal was to develop a tool that provides a common solution for the two problems described by providing content- and role-specific support for collaborative learning tasks to each individual learner. The basic idea to achieve this goal is very simple: Rather than trying to enrich pre-selected web-pages with scaffolds or embed scripts into particular learning platforms, our approach endows the browser with instructional support by means of a library of hard-coded scripts and scaffolds that are invoked based on the recognition of functionally equivalent web pages or components of different collaborative learning platforms. Therefore we regard this as a kind of two-fold “Copernican Turn” in sustainable script and scaffold development. We will now describe this approach in more detail using the scenarios of the two problems described before as examples.

Practically any search engine such as Google, Yahoo! or Live Search consists of a form for entering a search query that leads to a series of results pages with a common structure. From here, the user can reach the web pages that may contain the information he or she is looking for. Accordingly, there are three types of functionally equivalent pages users have to traverse whatever web search engine they use: the search query form, the results page, and the pages reached from the results page. The first two of these page types usually have a structure of functionally equivalent elements that is typically identical across different search engines: For example, a search query form usually contains one (or sometimes several) text field(s) for entering the search terms and a button for starting the query. As each of these page types corresponds to a specific phase during an online search, specific cognitive processes that may require support are associated with each type of page. For example, while a user is on the search query form, he or she needs to generate a set of search terms yielding relevant results and precluding irrelevant results. At the results page, hits need to be selected based on an evaluation of the information provided along with them. At the pages reached from there, search strategies have to be applied to locate relevant information on the website. Based on this correspondence between the types of pages traversed during an online search and the cognitive processes required in different phases of the search, the capacity of recognizing these page types and their component objects allows for the development of scaffolds that are specific for each site type, but apply to any web search engine that has this kind of structure.

This solution can be transferred to the problem of re-usable collaboration scripts for online discussion. In our own prior research (e.g. Stegmann, Weinberger & Fischer, 2007) a script for the construction of single arguments was implemented in an online discussion board of a collaborative learning environment by means of prompts and separate text boxes for the parts of an elaborated argument. These were embedded in the form for entering messages and their contents were composed to one continuous message before posting the message. The advantage of this approach lies in the fact that it allows for the
use of a library of hard-coded collaboration scripts contained in the browser that can be used with a broad variety of web-based collaboration tools. This solution works for any kind of “micro” script that applies to a single component page of a collaborative learning environment. As our solution for implementing re-usable collaboration scripts can also navigate automatically to different component pages of a collaborative learning environment, it also allows for the implementation of guidance for larger sequences of learning activities in “macro” scripts.

Main features of S-COL

We now turn to the specification of these ideas in the S-COL (Scripting for Collaborative Online Learning) technological approach, which provided the basis for the implementation in the tool we developed to solve the two interconnected problems described before.

The graphical user interface

As mentioned before, the tool was implemented as a browser plug-in. Accordingly, the main part of its graphical user interface is the browser itself. The area of the browser used for displaying web pages is split in two parts (see figure 1). The area on the right hand side is called the “browsing area”. It exhibits exactly the same behaviour as a standard web browser: It can present HTML pages, and the user can navigate through them by using links and menu elements of the browser such as the home, forward, and backward buttons. The part on the left hand side is called the “scaffolding area”. Its size is flexibly adaptable both by the user dragging its border as well as by JavaScript functions. Furthermore, it can be invoked and hidden by a function key. Its content can be flexibly designed using HTML. The content of the scaffolding area (text boxes, buttons etc.) can “interact” with objects in the browsing area. For instance, information from the browsing area such as the URL of the actually displayed Webpage can be read out. Furthermore, the browsing area can be manipulated by posting text into forms, activating buttons or even by navigating to an arbitrary URL. The scaffolding area moreover contains a menu bar providing functionalities such as loading scripts or scaffolds and configuring the navigation behaviour of the tool (see below).

Tool functions

This section describes the functions of the S-COL tool. For each function first the behaviour of the tool is characterized, then the methods for achieving this behaviour are described.

Content- and role-specific support. The display of content- and role-specific support for learners collaborating with this tool requires the recognition of types of web pages and their component objects. For example, to support the writing of arguments, a script needs the information whether the current page is a page...
for the composition of a new message or not. Based on this, scaffolds and scripts are displayed in the scaffolding area of the tool.

The recognition of the pages is achieved by means of a template file that contains a description of each variant of every page type as well as its components. To identify a page type, both the URL and the Document Object Model (DOM) of the page can be used. For example, it can contain the URL of the Google variant of the search query form to identify this page as the page type “search query form”, and the XPath expression or the ID of the text box for the search terms to identify the text box on the Google search query form as this type of object. It can contain the same information for other search engines as well as similar information for the other page types traversed during an online search. Based on the information contained in the template file, a JavaScript function yields the type of the page currently displayed. Using the values returned by this function (and potentially also the assigned role of the person using the computer), the contents of the scaffolding area are selected by JavaScript code contained in the HTML file loaded into the scaffolding area. Thereby, the contents of the scaffolding area can be adapted in a content-specific way according to the type of web page recognized, and in a role-specific way according to the role a person may have been assigned before. This includes that the scaffolding area can be configured to disappear if no scripts or scaffolds should be provided.

Collaborative web browsing. The tool furthermore allows for collaborative web browsing. This is to say that all learners belonging to the same group can automatically see the same web pages in their browser. The assignment to groups is done via a dialog window for group formation. Each participant of a group has the opportunity to navigate the whole group to a different web page by simply using his or her browser the usual way, i.e. by clicking on links, menu elements, or entering a new URL. This brings the page he or she navigates to on the screen of all members of the group. If a learner opens a new tab, new tabs will be opened in all way, i.e. by clicking on links, menu elements, or entering a new URL. This brings the page he or she navigates to on the screen of all members of the group. If a learner opens a new tab, new tabs will be opened in all connected browsers.

The collaborative web browsing function can be adapted in several ways. In principle, each user can dissociate him- or herself from collaborative web browsing. This comprises an active and a passive component: On the one hand, a user may switch off the function that “sends” his or her navigation actions to the other group members. This has the effect that his or her navigation actions have no effects on what is displayed on the computer screens of the other members of the group, so he or she can no longer “lead” the group to other pages. On the other hand, he or she may switch off the function that “receives” the navigation actions of the other group members. This has the effect that navigation actions of other group members have no effects on what is displayed on the respective group member’s computer screen, so he or she no longer “follows” other group members to other pages. S-COL also offers a JavaScript function that allows script developers to switch these communication functions on and off. Furthermore, the rights to switch on and off the “sending” and “receiving” of navigation actions can be set globally to allow teachers to control their students’ options during collaborative learning tasks on the Internet.

The collaborative web browsing function is based on JavaScript functions that send messages to all group members as well as to all connected web browsers with activated S-COL plug-in. These messages can also be used to synchronize the scaffolds. These messages could also be used for a chat tool implemented in the scaffolding area.

Administration
Group settings. The tool contains a dialog window for the formation of groups that also allows to assign roles to individual members of the groups and to select scripts and scaffolds from the library to be displayed in the scaffolding area for individual users. This window is password-protected and can be used from each connected browser to change the group-related settings of any of the connected users.

Script and scaffold library. Furthermore, S-COL has a library that contains the different scripts and scaffolds that can be invoked in the scaffolding area. Currently this is implemented as a folder that contains all the files with the contents of the scaffolding area as well as the JavaScript code for adaptation and talking to the contents of the browsing area. For the future, we plan to either develop or integrate a script and scaffold editor. This will allow for easy configuration of the template file used for the recognition of site types and the organization of page types, subtypes and their component objects. It will also simplify the assignment of script prompts and scaffolds to page types and subtypes as well as roles and also states of counters to implement the fading of scripts or scaffolds based on the number of occasions the students have experienced to practice certain skills.

Further potentials, limitations and open problems
In this contribution we suggested an approach to the development of flexible and reusable collaboration scripts and scaffolds that draws on earlier conceptual and empirical research on scripts, but makes a big step forward by changing perspectives on the problem. While we think that S-COL is already a very helpful tool for research on technology-supported collaboration scripts, we are also quite sure that it has much more potential. As described above, important steps have been made with respect towards a common script language (cf. Harrer & Malzahn,
An implementation of an interpreter of this IMS-LD based language in S-COL would allow the graphical modelling of new scripts and their broad application in many web-based learning environments. Subsequently, the transfer of successful collaboration as well as systematic research on collaboration scripts would be much easier to conduct. Also approaches like “manyscripts” (Dillenbourg & Hong, 2008) could be integrated into the scaffolding area of S-COL. Thereby, the tool may bridge the gap between the development of new technology-based collaboration scripts and their systematic application in research and practice.

In the wake of the features of S-COL that allow for an easy implementation of collaboration scripts, several additional functions to support research were developed. For example, if it is necessary to analyze the search activities of learners, usually screen recordings have to be analysed. S-COL can log the browsing behaviour including all clicks, mouse movements, and the content (i.e. the DOM) of all web pages visited. S-COL may also help to transfer identification data from pre-test to post-test in field studies, even without awareness of the participants, thereby reducing the likelihood of mistakes and data loss. Furthermore, the tool can be used to administer process measurements (e.g. for flow experiences) by the Experience Sampling Method (ESM) during learning activities: By a JavaScript function using the communication functions of S-COL each connected browser can be individually asked to open a short questionnaire in a pop-up window.

Some limitations and open issues also have to be discussed. The main limitation is the restriction to HTML-based learning environments. A growing share of learning platforms implement Java- or Flash-based communication tools. These tools can hardly be scaffolded with S-COL. Besides, the possibility of the logging of all user events including the DOM also has its dangers: S-COL could easily be configured to trace all web activities of a user and to send this data to a specified server somewhere on the web. However, an S-COL version without unsafe tracing functions can easily be derived from the current version.

While these issues still need to be addressed and there is still further potential to be actualized by connecting S-COL to previous achievements in script and scaffold development, S-COL has probably been a major breakthrough in sustainable script and also scaffold development for changing web-content.

References