Behavior Research Methods
2007, 39 (2), 248-258

DEWEX: A system for designing and conducting Web-based experiments

Anja Naumann
Deutsche Telekom Laboratories, Berlin University of Technology, Berlin, Germany

Angela Brunstein
Carnegie Mellon University, Pittsburgh, Pennsylvania

and

Josef F. Krems
Chemnitz University of Technology, Chemnitz, Germany

DEWEX is a server-based environment for developing Web-based experiments. It provides many features for creating and running complex experimental designs on a local server. It is freeware and allows for both using default features, for which only text input is necessary, and easy configurations that can be set up by the experimenter. The tool also provides log files on the local server that can be interpreted and analyzed very easily. As an illustration of how DEWEX can be used, a recent study is presented that demonstrates the system's most important features. This study investigated learning from multiple hypertext sources and shows the influences of task, source of information, and hypertext presentation format on the construction of mental representations of a hypertext about a historical event.

DEWEX: AN OVERVIEW

The *D*evelopment *E*nvironment for *W*eb-Based *Ex*periments (DEWEX) is an environment for generating and conducting Web-based experiments either on the Internet or in the laboratory. It has been developed since 2001 within the hypertext research project "User-Oriented Presentation of Information on the Internet," together with the Chemnitz LogAnalyzer (Brunstein, Naumann, & Krems, 2005). The Chemnitz LogAnalyzer is a free tool for analyzing log files from Web-based experiments (e.g., those conducted with DEWEX). The "heart" of DEWEX is the CGI program nm.cgi, which interprets the folder and document structure of the environment for conducting Web-based experiments. These experiments can be generated with DEWEX even by those with little expertise in programming or in using the Internet and HTML. With the help of DEWEX, the materials for the experiment (i.e., instructions, questionnaires for participants' data, pretests, text or picture materials for different experimental conditions, posttests, and additional questionnaires) can be created, and the order of the presented materials and the assignment of participants to different experimental conditions can be defined. Experimental designs with one or more factors, whether within subjects, between subjects, or mixed, are possible. Those experiments can then be made available on the Web (e.g., by being linked to on an index.html page) and data can subsequently be collected. Log files are generated by DEWEX and automatically stored on the local Web server in an *Experiment* folder. Users' data can be collected in the form of answers input via radio buttons, pull-down menus, or interactive text fields; of response times; and of navigation patterns, all recorded in the logs. The well-structured log files can then be analyzed and visualized very quickly and easily, for example with the Chemnitz LogAnalyzer. Some information (e.g., number of correct answers) can also be seen directly in the raw log file data, without additional analyses.

In addition to text, DEWEX allows the following features to be implemented in an experimental design:

- graphics and pictures;
- text structure overviews or text menus, either passive (i.e., nonclickable) or active (with navigation possible by clicking on a menu item);
- feedback functions (e.g., number of correct answers);
- JavaScripts for dynamic displays (as used in, e.g., www.tu-chemnitz.de/projekt/elearning/tutor_engl);
- the option of assigning participants to different experimental conditions (automatically or in a predetermined manner).

DEWEX basically consists of one main folder with several subfolders, with files that can be adapted to user

A. Naumann, anja.naumann@telekom.de

needs. The main folder, *Experiment*, can be downloaded from www.tu-chemnitz.de/projekt/elearning/DEWEX/ as a .zip archive and can be installed on any Web server that supports the CGI program nm.cgi. This program is used to connect the other files in the *Experiment* folder and mediate between them. It is written in C++, and its source code is available on our Web site (www.tu-chemnitz.de/projekt/elearning/DEWEX/cgi%20source%20code/). The configuration of the environment—that is, the creation of question/answer formats—takes place by entering text components and commands in different text files. In the directory www.tu-chemnitz.de/projekt/elearning/DEWEX/, a user manual is provided that describes all the details of our tool.

We have used DEWEX for conducting Web-based studies that can be classified into four categories. The first type is a simple forced choice task, as can be seen in the example at www.tu-chemnitz.de/projekt/elearning/Potter_engl. In that study, participants received instructions, a questionnaire, and some picture–name pairs for characters out of the Harry Potter story. These characters had to be rated as male or female. This study investigated how learners integrate different sources of information for their choices.

The second type of study—and the main reason for developing this environment—is hypertext navigation studies, such as the one at www.tu-chemnitz.de/projekt/elearning/Mq (please fill in "A001" for the ID). In this study, we investigated the influences of task, presentation format, and source of information on learning from a hypertext. Here, participants received instructions, pre- and posttest questionnaires, and one of several versions of a hypertext. The hypertext versions had different structures according to different sources of information that described, for example, a historic event. These structures were defined by a single assignment file so that the text components were kept constant for all participants. In addition, several navigation and orientation aids were presented in parallel with the text components. These aids were also defined by a single assignment file. This is the kind of study we will use later to describe the environment in detail, and at that time we will explain at length the example study at www.tu-chemnitz.de/projekt/elearning/Mq.

The third type of study is represented by a survey-like study in which participants explored the home pages of several banking companies. The participants looked at a number of external Web pages, and after a predefined amount of time they had to answer questions about those particular sites. In this study, hundreds of questionnaire components had to be handled.

The fourth and last type of study utilizes dynamic displays, as can be seen at www.tu-chemnitz.de/projekt/elearning/tutor_engl. In this example, participants had to construct a virtual racing car for a simulated race. Specific to this kind of study is the combination of questionnaires, tests, and text, on the one hand, with JavaScript-based applets, on the other; data are collected for all of these components. This kind of study relates to the most recent development of the environment, for creating intelligent tutoring systems capable of teaching complex tasks (such as designing experiments) to undergraduate psychology students.

DEWEX IN DETAIL

The first step in running an experiment with DEWEX is to store the *Experiment* folder in the working directory of the local server (in most cases, this will be an Apache server). The address for the experiment then will be http://servername/experiment. In order to create an experiment and use it later to collect data, it is important that the researcher have reading and writing privileges in that directory and its subfolders. The main *Experiment* folder contains a subfolder *cgi-bin*, which has the following contents.

Files

HTML pages needed for entrance into the experiment. These include the following:

Index.html. This file is the entry point for the experiment. It was created to give researchers the option of hiding the addresses of later pages in the experiment (see Features of DEWEX below), and automatically forwards participants to Start.html.

Start.html. This file may contain input fields if required—for example, for participant number and participant code (i.e., an individual combination of letters—e.g., four, which could consist of the first letter of a participant's town of birth, the last letter of their first name, etc.). This information could be used for assigning participants to experimental conditions and for compensating for drop-outs in some of the provided conditions. By default, however, assigning participants to conditions and generating participant IDs are executed automatically by the program. In that case, participants would not see the start page, but instead proceed to the page for one of the conditions when the experiment is started.

Templates for the design and appearance of the experimental interface. These style sheets include Start_Notnetscape4.css and Start_Allbrowsers.css and define the layout of the experiment (the frames, colors, etc.; for details, see the readme file online).

Files for error messages. These files are WrongId.html, SystemError.html, and Start_check.js. They become active when an error of some type occurs (e.g., missing or wrong login information). For example, WrongId.html informs the experimenter and the participant when a wrong participant number is typed in, which is only necessary when participants are not automatically assigned to experimental conditions. Start_check.js checks which functions are defined in Assignment.txt (e.g., "allow any ID").

nm.cgi. In some cases, the source code of this file (available, once again, at www.tu-chemnitz.de/projekt/elearning/DEWEX/cgi%20source%20code) may have to be compiled to run on a server's operating system. Compiling the program will only be necessary only once per system, and then the program will run for all follow-up studies. For more detailed information on compiling, see www.tu-chemnitz.de/projekt/elearning/DEWEX/

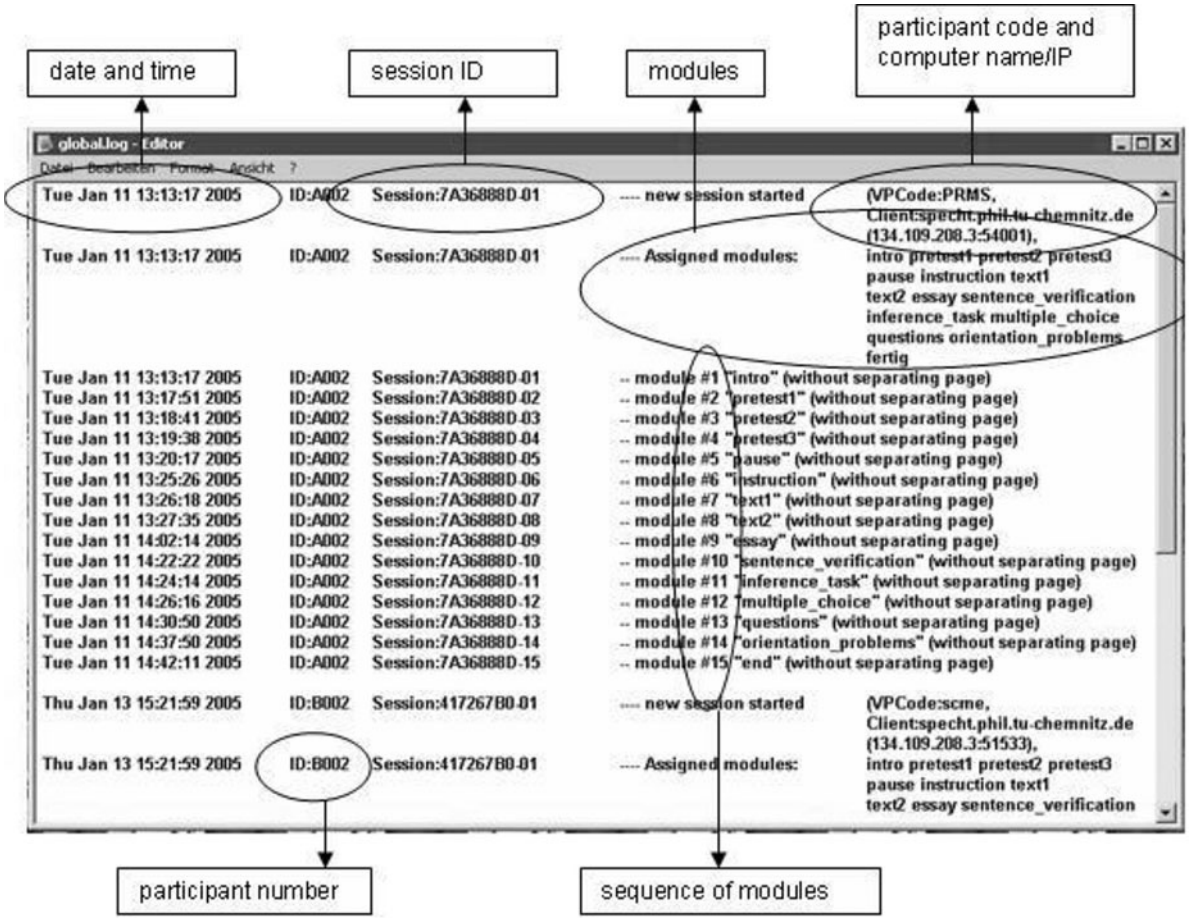

Figure 1. The global.log file, showing which modules were provided and when.

cgi%20source%20code/readme_compiling. For current operating systems, such as Windows (2000/2003/NT4.0/XP), Mac OS X, and Linux, the program is already compiled.

The *Data* Folder

Besides the program nm.cgi, the *cgi-bin* folder also contains the *data* folder. Within the latter folder, experimenters can make many of the changes necessary to adapt an experiment to their own needs. This is only the first part that needs to be changed for running a new experiment, however. This folder mirrors the experimental design, with all components of the study and all of the variations between participants, and consists of subfolders for log files and modules. If more than one experiment is planned for this environment, either the experiments could be stored in separate experiment folders, each containing all required components, or their components could all be stored together in one *data* folder. In that case, the experiments would be distinguished only by the assignment file. That configuration is especially recommended if experiments share components such as questionnaires, texts, or tests.

Log folders. The subfolders for log files include the following:

!-global-logs. In this folder, there is one file for all sessions, global.log. It contains data for all sessions of the experiment, including date, time, participant number, the modules provided and which were accessed, participant code (see the description above for the Start.html page), name of the computer, and computer IP address. The advantage of this setup is that the experimenter can see at a glance which modules were assigned, as well as which were accessed, when, and in what order (see Figure 1).

!-local-logs (all,copy). This folder contains log files for each module (see Figure 2), each named with an ID and module name. For the modules in which data are collected, there is one .txt file and one .log file. One of these files is more complete (the .log file), and the other is easier to process (the .txt file). Both contain the computer name/IP, participant number/ID, access time, and module name. The text file gives a good overview because it only shows participants' answers, which are ordered according to the modules. In contrast, the log file shows additional information as well—for example, answering times, example questions, questions not answered yet, and so forth. The log files are text files containing data separated by tabs. These files can be analyzed directly using software such as the Chemnitz LogAnalyzer (Brunstein et al., 2005).

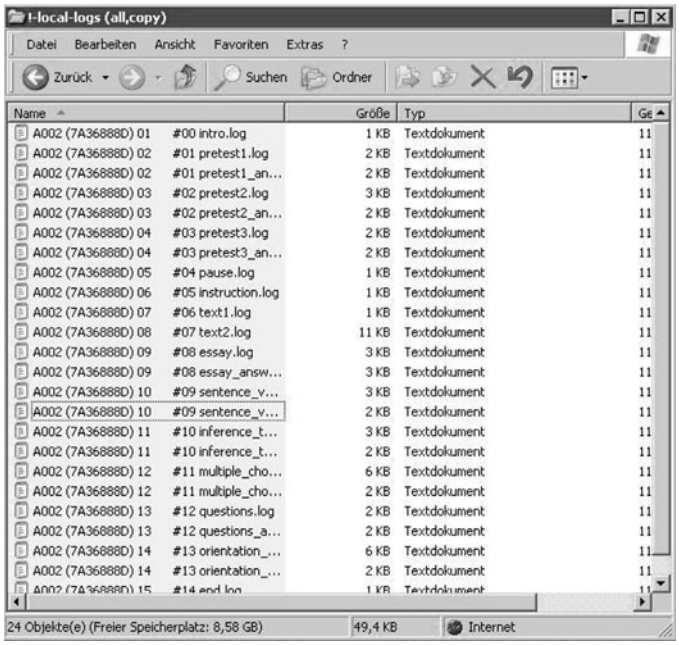

Figure 2. ***!-local-logs (all,copy)*** **folder, containing log files for each module.**

However, they can also be imported into other software (e.g., Excel or SPSS) for further processing. In this case, restructuring of the table might be necessary in order to create the favored format (e.g., all the data for 1 participant arranged in one column).

!-sessions. This folder gives an overview of all sessions and contains one .dat file per session. The advantage here is that all sessions listed are named after the session name. In this folder would be recorded, for example, which session was visited by which host.

!-counter. This folder shows the number of accesses to the experiment, or more precisely, how many participants went further than actually accessing the start page. Counters could also be created for number of entries, number of generated IDs, and number of completed runs. Each of these counters would be stored in a separate text file containing only these numbers, for a very quick overview during the running of a Web-based experiment.

Modules. Module folders (e.g., *Module1*, *Module2*, . . .) are the places where the pages of the experiment(s) are defined. They can also be named individually (e.g., *ReadingTask*). As an example, the module depicted in Figure 3 contains both templates and text files.

Templates. These relate to a questionnaire and its questions. All of the HTML templates here can be adapted individually or left in the default format provided.

Questionnaire templates (here, TemplateQuestionnaire .html) provide the frame for the questionnaire and define the appearance of the interface (e.g., the maximum number of questions per page, numbers printed before the questions).

TemplateQuestion.html defines the layout of the questions (e.g., presented in a separate frame), and TemplateQuestionDone.html is used to change questions that have already been correctly answered to passive mode, in which they are no longer clickable.

Files. The questions are stored as text files (e.g., Question001.txt) that fill the template. They have to be named with the word "Question" plus a number so that the StructureQuestions.txt file can access them properly, as described below. Each file defines a question, the possible answers, and the format of the answers (radio buttons, pull down menus, etc.; see Figure 4). They are presented in the experiment according to the specifications in the TemplateQuestion.html template.

StructureQuestions.txt defines the order of the questions and their distribution over several pages (with one or more questions per page). If the questions are arranged on one single page, a scroll bar appears automatically (see Figure 4).

The file StructureTexts.txt (not present in *Module1*) defines the text and format of text structure overviews.

StructureAssociations.txt (also not present in *Module1*) defines, for example, combinations of graphics and buttons.

Example. As an example of creating a module, we shall describe *Module1* in detail (see Figure 3). The goal of this module is to present 20 multiple-choice questions in a linear order. In order to do this, the 20 questions need to be created as 20 separate text files, and the file StructureQuestions.txt has to be adapted. Figure 4 shows examples. Question001 .txt defines the first question ("Which is the right answer?"), the answer options (A, B, C, and D), and the format of the answers ("#RADIO" means radio buttons with the options listed one below the other; "#LRADIO" would mean radio buttons with the options presented side by side). A minus sign ("-") prior to an option means that

that answer is wrong, and a plus sign ("+") means that it is right. The text "LINEARTEST" in StructureQuestions .txt defines the linear order of the questions. The empty line after each group of four entries defines a page break, resulting in four questions per page. The numeration of the entries (e.g., 001) needs to match the text files containing the questions (e.g., Question001.txt). However, the program needs only the numbers, not the complete file names, in order to run the experiment.

Files. In addition to its subfolders, the *data* folder also contains the following files:

Assignment.txt. This file assigns the modules and their order for each participant. The modules need to be listed in the desired order. Two versions of an assignment are attainable, corresponding to the two versions of Start.html: Either the program automatically assigns participants to experimental conditions and runs the experiment according to the corresponding line of Assignment.txt, or the experimenter defines the IDs and assignments by hand. In the latter case, participants have to type the correct ID into the start page, and the program then runs the experiment by executing the corresponding line of the file. In Figure 5, an example of Assignment.txt using a by-hand-defined ID is shown. The participant with the number VP_A001 first sees *Module1*, then *Module6*, *Module5*, *Module4*, and *Module3*. The experiment ends after that participant is presented *Module2*. In the case of randomized assignment of modules to the participants, the entry ";AllowEmptyID" must appear in the Assignment .txt file. In this case, the order of components for the different conditions still has to be defined in the file, but no ID number is assigned to each sequence of modules (see also the user manual).

AssignmentAuto.txt. This file defines the order and assignment of participants to experimental conditions in the case of automatic assignment. The program assigns each participant entering the system into one of the predefined experimental conditions. Along with our algorithm, this guarantees that all conditions are counterbalanced between participants; the first participant entering the system is run on the first condition, the second participant on the second version, and so on. Of course, there is no guarantee that all participants will complete the study, but there is no systematic bias in assignment of conditions.

As already noted, there are several possibilities for answer forms. The type of form is defined by the entry in the question file (e.g., Question001.txt). In addition to radio buttons (see Figure 4), text fields can also be generated. For example, "#TEXT 100,20" is a text field with a width of 100 characters and a height of 20. Pull-down menus can also be implemented (e.g., #-A1#-A2#-A3#+A4), with "+" marking the correct answer of a question and "-" an incorrect answer for later analyses. The log files then automatically show the percentages of right answers. The "forward," "back," or "next" buttons have to be defined in the questionnaire templates (HTML files).

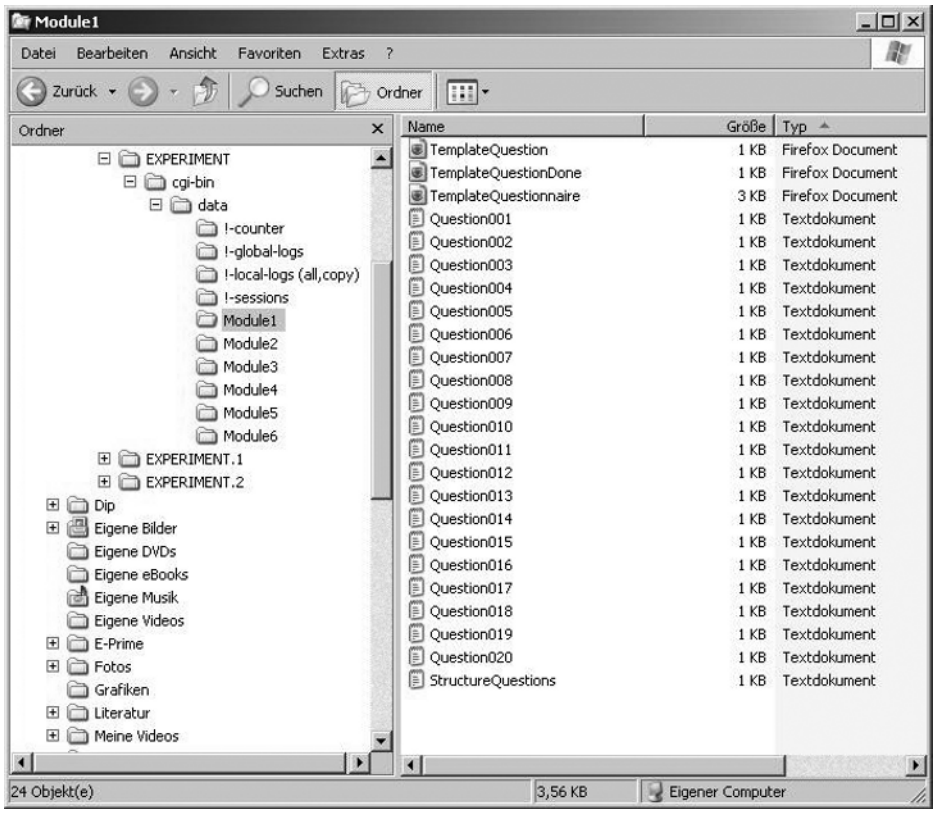

Figure 3. ***Module1*** **folder and its components.**

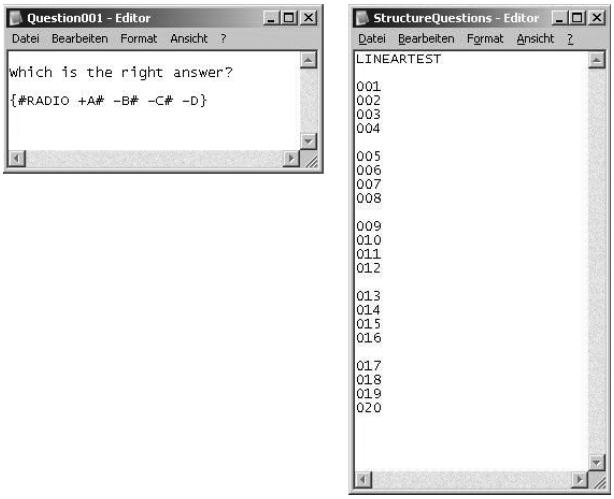

Figure 4. (Left) Question001.txt file, showing the question and the answer options in a radio button format. (Right) StructureQuestions.txt file, showing the order of questions.

Features of DEWEX

DEWEX also has a number of additional functions, such as aggregation of questions into categories, different feedback functions (feedback promptly after a question is answered or later), and automatic forwarding to the next page after a predefined period of time. A special feature is the option of generating text structure overviews. These are text menus that can be made clickable for direct navigation within a hypertext system and that are defined in the file Menuframe_template.html in the folder for a specific module. If the StructureTexts.txt file contains no specific command, the text structure overview will by default allow direct and active navigation to pages of interest within the module, but a ";PassiveMenu" entry will define an overview without any navigation option. The text menus can also be split into separate chapters. In DEWEX, it is also possible to hide page names in the address bar, which is advisable in order to avoid problems with experimental results due to the visibility of page names. When the page name is hidden, "Index.html" always appears in the address bar. Other features of DEWEX are the possibility of more than one link on one page and the implementation of a warm-up technique (see Reips, 2002b), which is especially relevant for Web experiments. Reips (2002a) also described several other very useful techniques and automatic safeguards to guide Web experimenters and help them avoid errors. These techniques are implemented in our example experiments and are automatically considered when copying the experiment and using it for adaptation to a newer experiment. For example, drop-outs (i.e., participants who do not complete all requirements) may become a problem in Web experiments. In DEWEX, drop-outs are detected by finding incomplete log files. In the laboratory, it is not problematic if some participants pause during the experiment and continue a little while later, but on the Web it might be, for then participants' computer IDs will need to be matched.

DEWEX can run several experiments in parallel. In this case, each experiment should be set up via either new folders in the *data* folder or separate *Experiment* folders. The assignment of participant codes to particular experiments is defined in Assignment.txt. When using more than one portal, the CGI and operating files need to be copied into each folder.

For example, an actual experiment on name–picture pair judgments can be downloaded from www.tu-chemnitz.de/projekt/elearning/DEWEX/ in order to see all the components of DEWEX. For installation on a Windows-based machine, one must download the file Potter-Local-V21.exe. This file already includes a local server application. Then the program has to be run, a folder name has to be given, the .cmd file has to be started in this new folder, and finally the Web browser should be started and pointed at http://localhost/. To see the experiment from the participant's perspective, one can also visit it at www.tu-chemnitz.de/projekt/elearning/Potter_engl.

DISCUSSION AND COMPARISON WITH OTHER TOOLS

DEWEX is a tool that provides many features for creating and conducting complex experiments on the Web. In addition to DEWEX, there are some other tools for generating surveys on the Web and Web-based experiments. For example, SurveyWiz (Birnbaum, 2000) is a tool for creating surveys on the Web. It uses JavaScript-based Web pages for the development of HTML forms and allows researchers to use such features as radio buttons, description fields, and an automatic generation of a short demographic questionnaire. FactorWiz (Birnbaum, 2000) is a tool for generating Web experiments. It assists researchers in building 1×1 to 9×9 factorial experimental designs by producing HTML source code that can then be processed. Either of these applications can be loaded on a Web server together with a matching CGI script and used for data acquisition. SurveyWiz and FactorWiz allow the user to set up surveys and experiments quickly, but without substantial experience in HTML programming, users can only create simple designs with those tools. In contrast, DEWEX has the advantage of requiring no substantial experience in HTML programming in order to set up complex designs. All important information can be input via plain text. However, basic HTML knowledge could be

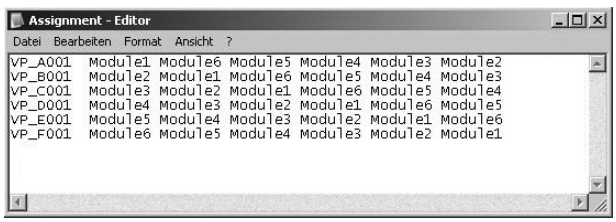

Figure 5. Assignment.txt file, defining the order of modules within the experiment for different participants.

helpful for the individualization of certain questions, for instance by formatting a word with bold letters.

Schmidt (2002) developed a server-side program for delivering experiments with animations, and it can also deliver CGI programs. This tool is very useful for simple graphical experiments, but it does not have many features and does not provide an experimental framework for larger experiments, as DEWEX does. WEXTOR (Reips & Neuhaus, 2002) is a Web-based tool that allows easy generation of Web-based experiments in 10 steps and also serves as a teaching tool. WEXTOR has the advantages of being easy to use, not requiring a great deal of practice, and having a coherent user interface. Also, it requires little or no knowledge of HTML programming and allows the possibility of visualizing the experimental design. However, our DEWEX tool incorporates some improvements even over WEXTOR. First, our tool uses the "post" method, which means that text input is not limited (with WEXTOR, the "post" method is possible, but it is not the default). In addition, with DEWEX more than one button per page can be implemented, the randomization of stimuli is not limited, and supplementary features (e.g., interactive graphics or text menus) can be integrated.

To summarize, DEWEX provides an environment in which complex designs can be generated, and many features can be implemented. It is also possible to construct an experimental framework that can be modified by the user. For an experienced user, it would take only 15–30 min to create an experiment like the Potter study mentioned above. The times required for the DEWEX download itself and for data analysis are not included in this estimate, but the download will only need to be done once, and a subsequent installation step is not necessary. The system is also relatively easy for novices to learn, because many of the features can be set to their default settings at first, and then adapted later to suit personal preferences. In addition, well-structured log files are automatically stored on the local server and can be analyzed and visualized with the Chemnitz LogAnalyzer (Brunstein et al., 2005). Even without using the Chemnitz LogAnalyzer, the log files can be imported as text files into any statistics software (e.g., Excel or SPSS) and then analyzed. When using other tools for analyzing the log files, one could also choose to analyze server-generated log files, if the experimenter has access to these files. Of course, DEWEX also has some limitations: It has no eye-catching user interface and visualizes the experimental design by the folder and document structure only. Therefore, it cannot serve as a teaching tool (as WEXTOR can). It also takes some practice for the user to customize all of the possible features.

EXAMPLE STUDY
Learning From Multiple Hypertext Sources

As an illustration of how DEWEX is used, we provide an account of one study on learning from multiple hypertext sources (available at www.tu-chemnitz.de/projekt/elearning/Mq; please fill in "A001" for the ID). In this context, some basic features of DEWEX are described. This study was chosen because it demonstrates the most important features of DEWEX that are necessary for most studies (e.g., instructions, questionnaires, and stimuli with different corresponding response options). The study is presented at length in Naumann, Wechsung, and Krems (2006).

Studies on learning with multiple texts (on the same topic from different authors or sources) have identified readers' comprehension goals, the presentation format, and the types of documents available as main predictors of the learning outcome and of processing success. Naumann et al. (2006) investigated how these factors interact when constructing a mental representation of the content of different text formats.

In previous studies, it has been demonstrated that the comprehension goal of the reader has an essential influence on the comprehension process (Britt, Rouet, & Perfetti, 1996; Hemmerich & Wiley, 2002; Voss & Van Dyke, 2001; Voss & Wiley, 2000; Wiley, 2001; Wiley & Voss, 1996, 1999): A goal requiring an argumentative procedure results in deeper processing as well as in better comprehension than does a goal more oriented toward a narrative processing. Argumentative goals would be (for example) writing a discussion or forming an opinion, whereas narrative goals would be writing a description or a summary of the text. In our study, we hypothesized that argumentative tasks support better handling and comprehension of multiple texts and sources.

Presentation format is another factor that influences comprehension processes for multiple texts. Most previous studies have compared linear text with a single text that was segmented into hypertext nodes (e.g., Britt et al., 1996), and the results showed the superiority of static, linear presentation for text comprehension (Gerdes, 1997; Naumann, 2003; Naumann, Waniek, & Krems, 2001). Static presentation therefore results in better performance for learning facts from a single text than does hypertext format. In contrast, analyses of multiple texts show a different picture. Hypertext environments can support the construction of a mental representation, but this is only possible when the task demands that an integrative approach be taken and when the browser design is supportive of that process (Wiley, 2001). Perhaps, hypertext environments reduce the effort required for comparing relevant information from text to text (Britt et al., 1996). Therefore, we assumed that hypertext format would help the reader in constructing a mental representation of multiple texts when an argumentative task (e.g., form an opinion) was given.

The third main factor influencing learning from multiple texts is the type of document. In this study, we focused on primary sources for historical events. Primary sources are documents that derive directly from the event (e.g., contracts or chronicles). We hypothesized that a better mental representation of the hypertext can be constructed by participants using primary sources (Rouet, Britt, Mason, & Perfetti, 1996; Voss & Wiley, 2000). This effect should then be enhanced when an argumentative task is given, because results have shown that primary sources are taken into account more with argumentative tasks (Rouet et al., 1996).

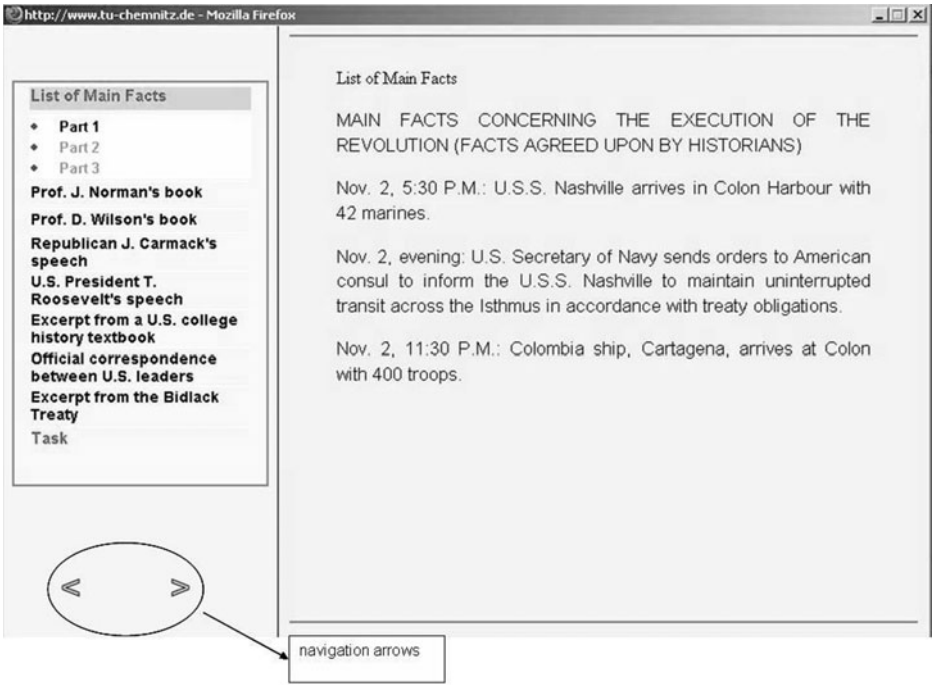

Figure 6. An example page. The navigation arrows at the lower left were present in the static text menu condition but not in the active text menu condition.

Method and Construction of the Experiment With DEWEX

The study was carried out in a computer lab of Chemnitz University of Technology, which allowed up to 20 participants at a time to take part in a session. A total of 122 students participated. All of the instructions, questionnaires, texts, and tests were constructed with DEWEX and presented on a computer screen. The participants interacted with a computer with a mouse and keyboard. They attended one session, lasting about 2 h.

Ten text excerpts served as text materials about the beginning of the revolution in Panama in 1903. These texts were adapted from Rouet et al. (1996) and translated into German. The text materials consisted of

- one list of facts,
- two authoritative primary sources,
- two reports from persons involved in the event,
- one textbook passage,
- four essays from historians, and
- a short introduction to the topic.

The eight texts and the introduction were presented one at a time. The number of essays from historians (two or four) and the number of primary sources (two or none) were varied.

In a 2 × 2 × 2 design, the factors task (argumentative vs. narrative), presentation format (static vs. active), and type of available documents (with primary sources vs. only secondary sources) were manipulated. In the argumentative task condition, participants were instructed to form an opinion about a controversy on the Panama topic. In contrast, in the narrative task condition, they had to write a description of the events. The active presentation format allowed navigation through the text via links within a text menu (a text structure overview; see the left frame in Figure 6). No additional navigation aid was provided. In the active condition, participants were able to navigate through the text in a self-directed way. In the static presentation condition, they could navigate through the texts only via the "back" and "forward" buttons (see Figure 6). Their navigation paths were therefore basically predetermined. The text menu was always presented on the left side of the screen. On the right side was the text frame. In both presentation format conditions, the actual position within the hypertext was marked with a color background. In both conditions, the list of facts and the task instruction could be reached by clicking on the corresponding links.

For implementing such a text menu, DEWEX requires the file Menuframe_template.html. To create an active, clickable menu, the StructureTexts.txt file in the specific folder does not need to be modified. If the menu should be static, the entry ";PassiveMenu" should be inserted into the file. Figure 7 shows the file StructureTexts.txt used in the active condition during the present study. The entries marked with a dot in the beginning appeared in the menu even when the participant was not in that particular section. The file names and the HTML tags used to change the color of items in the menu were hidden. In the Text*xx*.txt files, the actual text for each page was written. Several texts did not fit on one page, so they were distributed over multiple pages. This was done by dividing those texts into different files and putting them under one menu entry. With the entry ";TIMEOUT2100", the time allowed for reading all of the texts was determined. After

Figure 7. The StructureTexts.txt file for the active text menu condition.

that time (2,100 sec), DEWEX automatically forwarded each participant to the next assigned module. The HTML tags "" set the items "List of Main Facts" and "Task" to be displayed in red.

The participants had to read eight texts altogether. In the condition with primary sources, they received one list of facts, two authoritative sources, two reports of people involved in the event, a textbook passage, and two essays from historians. In the condition without primary sources, the primary sources were replaced by another two secondary sources (essays from historians). The number of texts was the same for both conditions. Including the task instruction page and list of facts, 15 text nodes were presented in all because most of the texts consisted of two text nodes.

The assignment of participants to experimental conditions resulted from definitions made in the Assignment.txt file in DEWEX. In this experiment, IDs were assigned to the participants by the experimenter. Therefore, every ID was typed into the Assignment.txt file (shown in Figure 5) in DEWEX before conducting the study (to assign participants randomly to conditions, ";AllowEmptyID" would have been included in the text file instead of each of the IDs). The names of the folders containing the pages each participant was supposed to see were written after the ID. As shown in Figure 5, the participant with the ID VP_A001 first saw the pages stored in folder *Module1*, followed by the pages stored in *Module6*, then folder *Module5*, and so on. If the entry ";AllowEmptyCode" had been included in Assignment.txt, no code to identify the study would have been required before starting the experiment.

On the first screen page of the study, participants had to type in their participant ID and the code for the study. The second page they saw showed the global instructions. In DEWEX, these instructions can be adapted beforehand by changing the text in the file Text001.txt in the instruction module folder.

Then, on a first questionnaire page, the participants answered questions concerning their age and gender and the subject of the study (see Figure 8). The questions and the answering format were defined in the DEWEX files Question001.txt, Question002.txt, and Question003.txt (see Figure 9, left). In Question001.txt and Question003.txt, text fields of different sizes are defined. In Question002.txt, radio buttons are instead defined for the answers; the "+" signs before the button descriptions mean that both "male" and "female" are right answers. The instruction shown at the top of the page was defined in the file TemplateQuestionnaire.html. The order of the questions was defined in StructureQuestions.txt (Figure 9, right). The empty line after the first three entries defines a page break. On the second page of the questionnaire, the contents of Question004.txt and Question005.txt (concerning

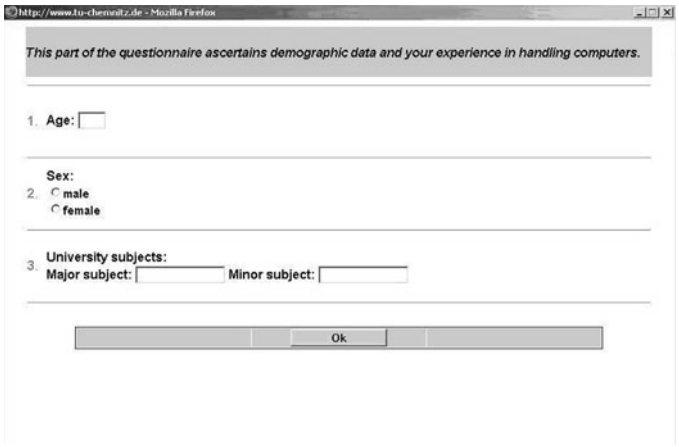

Figure 8. A sample questionnaire for demographic data.

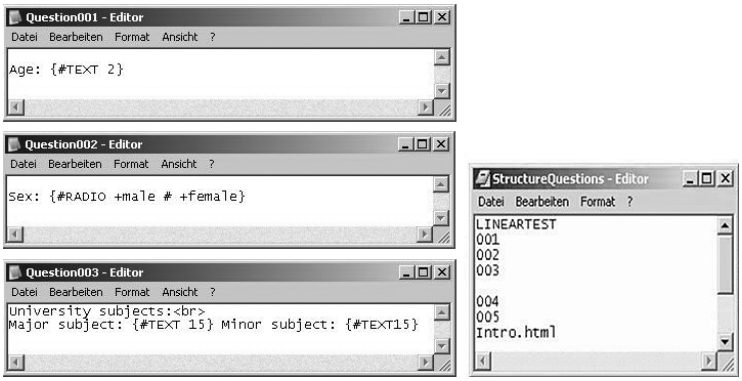

Figure 9. Construction of parts of the questionnaire for demographic data.

computer literacy) were presented. After that, Intro.html was accessed, which told the participants to await further instructions.

After filling out the questionnaires, participants had to perform a knowledge test about the history of Panama in order to check their previous knowledge about the topic. When all participants had done this first part of the study, they received instructions for reading the texts and doing the posttests. They then started to read the texts, beginning with the introduction text. After 35 min, they were automatically forwarded to the posttest by DEWEX (see the entry ";TIMEOUT2100" in structure.txt in Figure 7). The first task on the posttest was writing an essay. After that, participants had to carry out a sentence recognition task, an inference task, and a multiple-choice test. Then they had to answer open-ended questions about the texts. Finally, a questionnaire concerning any navigation or interface orientation problems they experienced was given. All questionnaires and knowledge tests were constructed similarly to the questionnaire for demographic data shown in Figure 8.

Results

In this study, we investigated the influence of the factors task, presentation format, and type of information (source) on learning from multiple sources. Detailed results are presented in Naumann et al. (2006); here, only a short summary is given.

As we had assumed, in the argumentative task condition participants were significantly better able to construct a mental representation of the hypertext than in the narrative task condition. Also in line with our predictions, participants acquired more factual knowledge in the narrative task condition. The best performance here was shown for the static presentation format, and no influence of the source of information was not found in this study.

The log files were analyzed with the Chemnitz LogAnalyzer (Brunstein et al., 2005). They showed that most participants reading the text in the active hypertext presentation format read the text in a linear way from the beginning to the end the first time. Then they started reading again, choosing individual reading sequences when reading a second time.

As predicted, the active presentation format resulted in fewer reported orientation problems for the argumentative task. In contrast, the static presentation format supported orientation for the narrative task.

In summary, the most important factor determining learning with multiple texts and ease of constructing a mental representation of the content of a text seems to be the task (narrative vs. argument). The successful accomplishment of the task can also be supported by browser design or the format of text presentation.

FINAL REMARKS

DEWEX is a software tool that can be used to set up Web-based experiments. Even novices can use the system to construct materials (text, questionnaires, tests, etc.) for experimental studies and to implement the experiments on servers. DEWEX conducts the experiment and collects data in log files that are easy to interpret at a first glance and can also be analyzed quickly with the Chemnitz LogAnalyzer or imported for analysis by other processing software (e.g., Excel or SPSS). In our experience, student assistants needed about 1–2 h of practice time and were then able to reshape an example experiment into their own new experiments. For learning how to program a new experiment from scratch, they needed up to 2 days. By conducting the study with DEWEX, it was possible to implement experiments very economically, with up to 20 participants in one session. Currently, DEWEX is used at four universities in Germany, the U.K., and the U.S.

AUTHOR NOTE

This article was presented at the Society for Computers in Psychology (SCiP) conference in Toronto, Canada, on November 10, 2005. It results from recent work done within the project "User-Oriented Presentation of Information on the Internet" by the "New Media" research group at Chemnitz University of Technology. The study was supported by German Research Foundation Grant KR 1057. We thank Tobias Winkler for programming the DEWEX tool and Jacqueline Waniek for helpful comments on earlier versions of the tool. Correspondence concerning this article should be addressed to A. Naumann, Deutsche Telekom Laboratories, Ernst-Reuter-Platz 7, D-10587 Berlin, Germany (e-mail: anja.naumann@telekom.de).

REFERENCES

BIRNBAUM, M. H. (2000). SurveyWiz and FactorWiz: JavaScript Web pages that make HTML forms for research on the Internet. *Behavior Research Methods, Instruments, & Computers*, **32**, 339-346.

BRITT, M. A., ROUET, J.-F., & PERFETTI, C. A. (1996). Using hypertext to study and reason about historical evidence. In J.-F. Rouet, J. J. Levonen, A. Dillon, & R. J. Spiro (Eds.), *Hypertext and cognition* (pp. 43-72). Mahwah, NJ: Erlbaum.

BRUNSTEIN, A., NAUMANN, A., & KREMS, J. F. (2005). The Chemnitz LogAnalyzer: A tool for analyzing data from hypertext navigation research. *Behavior Research Methods*, **37**, 232-239.

GERDES, H. (1997). *Lernen mit Text und Hypertext* [Learning with text and hypertext]. Lengerich: Pabst.

HEMMERICH, J. A., & WILEY, J. (2002). Do argumentation tasks promote conceptual change about volcanoes? In W. D. Gray & C. Schunn (Eds.), *Proceedings of the Twenty-Fourth Annual Conference of the Cognitive Science Society* (pp. 453-458). Mahwah, NJ: Erlbaum.

NAUMANN, A. (2003). *Wissenserwerb und Informationssuche mit Hypertexten: Die Bedeutung von Strukturierung, Navigationshilfen und Arbeitsgedächtnisbelastung* [Knowledge acquisition and information retrieval with hypertext: The impact of structure, navigation aids and working memory load]. Doctoral dissertation, Technische Universität Chemnitz, Germany. Retrieved September 16, 2006, from archiv .tu-chemnitz.de/pub/2004/0117/index.html.

NAUMANN, A., WANIEK, J., & KREMS, J. F. (2001). Knowledge acquisition, navigation, and eye movements from text and hypertext. In U.-D. Reips & M. Bosnjak (Eds.), *Dimensions of Internet science* (pp. 293-304). Lengerich: Pabst.

NAUMANN, A., WECHSUNG, I., & KREMS, J. F. (2006). *Learning from multiple hypertext sources*. Manuscript in preparation.

REIPS, U.-D. (2002a). Internet-based psychological experimenting: Five dos and five don'ts. *Social Science Computer Review*, **20**, 241-249.

REIPS, U.-D. (2002b). Standards for Internet-based experimenting. *Experimental Psychology*, **49**, 243-256.

REIPS, U.-D., & NEUHAUS, C. (2002). WEXTOR: A Web-based tool for generating and visualizing experimental designs and procedures. *Behavior Research Methods, Instruments, & Computers*, **34**, 234-240.

ROUET, J.-F., BRITT, M. A., MASON, R. A., & PERFETTI, C. A. (1996). Using multiple sources of evidence to reason about history. *Journal of Educational Psychology*, **88**, 478-493.

SCHMIDT, W. C. (2002). A server-side program for delivering experiments with animations. *Behavior Research Methods, Instruments, & Computers*, **34**, 208-217.

VOSS, J. F., & VAN DYKE, J. A. (2001). Argumentation in psychology: Background comments. *Discourse Processes*, **32**, 89-111.

VOSS, J. F., & WILEY, J. (2000). A case study of developing historical understanding via instruction: The importance of integrating text components and constructing arguments. In P. N. Stearns, P. Seixas, & S. Wineburg (Eds.), *Knowing, teaching, and learning history: National and international perspectives* (pp. 375-389). New York: NYU Press.

WILEY, J. (2001). Supporting understanding through task and browser design. In J. D. Moore & K. Stenning (Eds.), *Proceedings of the Twenty-Third Annual Conference of the Cognitive Science Society* (pp. 1164-1169). Mahwah, NJ: Erlbaum.

WILEY, J., & VOSS, J. F. (1996). The effects of "playing historian" on learning in history. *Applied Cognitive Psychology*, **10**(Spec. Issue), S63-S72.

WILEY, J., & VOSS, J. F. (1999). Constructing arguments from multiple sources: Tasks that promote understanding and not just memory for text. *Journal of Educational Psychology*, **91**, 301-311.

(Manuscript received November 13, 2005;
revision accepted for publication October 5, 2006.)