Abstract: External computer-supported collaboration scripts may support learners in collaboratively using new technologies. This contribution introduces a framework that offers an object-oriented package of classes and methods that supports eXtremely Simple Scripting (XSS), i.e., the rapid implementation of CSCL scripts for the use with new technologies. We report two examples on how this framework has been used to implement computer-supported collaboration scripts in mobile learning with tablet-PCs and with an interactive table.

Increasingly, new technologies like interactive tables, ambient displays, handheld devices, smart boards, or tablet PCs can be found in classrooms. However, neither teachers nor students seem to have the adequate knowledge (i.e., internal scripts; cf. Schank & Abelson, 1977) to cope with these new possibilities in an effective way. Interaction and learning processes in scenarios using new technologies may differ significantly from their collaboration experiences. Learners may thus lack effective internal scripts to use such new technologies for collaborative learning. In these situations, learners need additional instruction (i.e., an external script). External computer-supported collaboration scripts may support learners using innovative collaboration technologies productively.

However, the development (e.g., the programming of the software) of new computer-supported collaboration scripts has been requiring considerable effort so far, because the design of the script requires expertise in both cognitive psychology of learning and in computer science. Collaboration scripts can be seen as consisting of a number of components and mechanisms that are highly interrelated. Usually, “the first year” of collaboration between educational scientists and computer scientist is spent to find a common language with respect to these components and mechanisms.

However, recently a group of Finnish, French, German, Greek, and Swiss researchers in the European Research Team CoSSICLE developed such a common specification of computer-supported collaboration scripts (Kobbe et al., 2007). This contribution introduces a framework that transferred this specification into an object-oriented package of classes and methods that support rapid implementation of scripts for the use with innovative technologies: the eXtremely Simple Scripting (XSS). Furthermore, we report two examples to show how this framework can be used to implement computer-supported collaboration scripts on different devices.

Specification of Collaboration Scripts
Kobbe and colleagues (2007) distinguished components and mechanisms of computer-supported collaboration scripts. Components can be seen as the objects that can be manipulated by specific mechanisms. The components are participants, activities, roles, resources, and groups. Important attribute of participants is their number, due to the fact that some scripts require a fixed number of participants per group. These requirements often result from a specific number of other components like roles or resources. Activities are the main component with respect learning, because the actual learning processes are triggered. In case of a peer review script, learners may be asked to explain or to provide constructive critique. Roles, e.g. the role of an analyst or a constructive critic, are often used to define clusters of a set of activities. Within the script, learners use, manipulate, and/or create resources. Learners may get a problem case and theory to solve a problem as initial resource. Their task is to create a new case analysis and/or to discuss their analysis until they agreed on a joint solution. The problem case, the theory, and all contribution of learners in the discourse are regarded as resources. While roles specify a set of activities, groups are used to specify learners who directly interact.

The mechanisms of computer-supported collaboration scripts to manipulate the components are task distribution, group formation, and sequencing (Kobbe et al., 2007). The task distribution within a group of learners is a central feature of collaboration scripts. The activities, roles, resources described in a script may be distributed within a group. Some scripts use specific group formation mechanisms to facilitate learning processes, i.e., instead of self-organized groups, an algorithm is used to compose groups. Through sequencing,
scripts often provide a temporal structure of components and mechanisms, e.g., scripts may specify the sequence of activities, roles, resources, and/or group formations.

**The eXtremely Simple Scripting (XSS) framework**
The XSS framework is based on the specification developed by Kobbe and colleagues (2007). The aim is to provide a functional framework that allows for the rapid development of computer-supported collaboration scripts for innovative technologies. Therefore, the XSS framework provides a number of classes and methods to realize scripts. However, all classes can be extended or overwritten if needed to implement a specific script. The framework consists of three main components: (1) The **Script class**, which specify the overall structure of any collaborative learning application that is based on CSCL scripts. Frequent operations, such as the group formation and sequencing, are pre-implemented to support the developers. (2) The **GroupBuilder class** provides group composition features. (3) The **Sequence class** organise and manage the activities of all participants. In the following, we describe the procedure how the framework can be utilized to implement computer-supported collaboration scripts.

**Initialising the script.** To realise a script, the **Script class** has to be filled with information on the script first. A list of possible activities as well as a list of possible roles has to be provided. The roles comprise the information about the sequence of activities with regard to this role. Furthermore, phases are defined, which serve to coordinate the participants within one group. To synchronise the phases, the framework offers different options: (a) the phases of each participant can be independent, (b) the phases can be synchronised, i.e. the phase does not start before the last member of a group reached the beginning of that phase, or (3) when the first member reaches the phase the system will move all other members of a group to the same phase.

**Participants.** After initialising the script, participants can enrol (can be enrolled). For each participant an instance of the class **Participant** is created. Furthermore, the participants are assigned to a group. The assignment of participants to groups can, with regard to the script, be done right after enrolment or until all participants are enrolled. Algorithms that can be defined in the **GroupBuilder class** perform the group composition and the assignment of roles. To trace changes with respect to the assigned activities, each **Participant class** is observed by the view of the participant.

**Sequence.** A **Sequence class** is assigned to each group. This class has access to all participants and the list of phases provided in the **Script class**. Using this list, the current phase is handed over to the **Participant class**. Knowing it’s own role, the **Participant class** use this information to start the next activity. At the same time, the **Participant class** informs the respective view about the change of the activity. With regard to the activity, the view can provide the required user interface including all necessary functions. If the **Participant class** finished the current activity, the **Sequence class** is informed. If the activity has an output (e.g., a short essay), this output is saved and made available as resource for all other activities. The **Sequence class** further checks the status of the phases of all group members. If a participant reaches a phase that has to be reached by all participants, the participant has to hold out for the other participants. If a participant gets to a phase that is defined to force all participants into this phase, the phases of all participants of one group are set to this phase. If the phase has no specific flag with respect to synchronisation, the **Participant class** moves forward to the next regular class. This will cause that the procedure described above is restarted with the phase. This procedure is performed until the last phase in the list of phases is finished.

**Two Examples Using the XSS Framework**
In this section, we describe two examples of computer-supported collaboration scripts that were adapted for innovative devices. The first example is the MURDER script (cf. O’Donnell & Dansereau, 1992) that is implemented for the use on TabletPCs or handheld devices. The second example is an argumentative collaboration script (cf. Stegmann, Weinberger, & Fischer, 2007) that was implemented for an interactive tabletop display. For both examples, specific features of the XSS framework will be focused. The evaluations of both scripts are currently running. The outcomes of these evaluations will be presented at the conference.

**The MURDER Script in mobile learning**
The MURDER script was developed to support dyads in face-to-face settings with respect to text comprehension. First of all, a text of several pages will be segmented into smaller sections. For each section, the MURDER script specifies several activities for two learners (cf. O’Donnell & Dansereau, 1992): (1) **Mood** – the learners relax and concentrate on the task, (2) **Understand** – both partners read the first section of the text, (3) **Recall** – learner A reiterates the text section without looking at the text, (4) **Detect** – partner B provides feedback without looking at the text, (5) **Elaborate** – both learners elaborate on the information, and (5) **Review** – both partners look through the learning material once again. The learning partners are supposed to engage in these activities for each text segment, switching roles regarding recall and detection for each segment, until they have completed the text. The implementation of the MURDER script for the use on a TabletPC should especially
enhance the fourth and fifth phase by providing special features to trigger deep elaboration and transactive discussions.

The MURDER script heavily relies on two main features of the XSS framework: (1) the synchronisation of phases and (2) the feature to use the output of one phase as resource for another phase. The synchronisation of phases is especially important with respect to the switch from phase “Understand” to phase “Recall”. The computer-supported implementation of the MURDER script has to ensure that both learners finished the phase “Understand” before learner A starts to describe the text section in the phase “Recall”. The possibility to use the output of one phase as resource of another phase is crucial in the computer-supported version of the MURDER script to support phase “Elaborate” and phase “Review”. The learners are supported with specific visualisations regarding the similarity of their keywords made in phase “Understand”.

The Argument Construction Script
The argument construction script aims to facilitate the quality of argumentation during discussions. So far, the script was only implemented to support online discussions. The script focuses on the quality of single arguments (according to a simplified model of Toulmin, 1958) as well as the quality of argumentation sequences (according to Leitão, 2000). There, the script consists of input text boxes for a claim, grounds and qualifications. Each text box of the interface is to be filled out by the learners to construct a completely explicit argument. Furthermore, the script aims to facilitate specific argumentation sequences of argument-counterargument-integration (following Leitão, 2000). Hence, the script triggers learners to answer argumentations with counter arguments and tries to integrate the argumentations in the end in an integration.

The implementation presented here transfers the argument construction script to face-to-face scenarios at an interactive table. Learners are asked to apply a specific theory to a problem. While they argue, they should put their arguments on the table and relate them with the arguments that are already posted. In this scenario, the XSS framework provides the methods to trigger the construction of single arguments (i.e., writing the claim is the first phase, adding the grounds is the second phase, and so on) as well as the construction of argumentation sequences (trigger to reply to an argumentation with an counter argumentation).

Further potential, limitations and open issues
The XSS framework has the potential to ease the development of computer-supported collaboration scripts for innovative technologies significantly. By providing an easy to use implementation of the script specification of Kobbe and colleagues (2007), the complexity of interacting components and mechanisms can be handled more easily. However, so far the XSS framework does not include an interface to author scripts. A possible solution would be to integrate the framework with other available tools like the script modelling tool (that produces an IMS-LD file; Harrer & Maizlahn, 2006) that already produce script description on the base of the specification of Kobbe and colleagues (2007). Another open issue is, whether the XSS framework can cope with the large variety of scripts. So far, we developed and evaluated only the scripts described above. Many more scripts need to be tested to find an answer to this issue.

References