Over-scripting CSCL: The risks of blending collaborative learning with instructional design.

Prof. Pierre Dillenbourg pierre.dillenbourg@epfl.ch

CRAFT - Centre for Research and Support of Training and its Technology

Swiss Federal Institute of Technology, Lausanne

Abstract. Free collaboration does not systematically produce learning. One way to enhance the effectiveness of collaborative learning is to structure interactions by engaging students in well-defined scripts. A collaboration script is a set of instructions prescribing how students should form groups, how they should interact and collaborate and how they should solve the problem. In computer-supported collaborative learning (CSCL), the script is reified in the interface of the learning environment. This contribution dismantles the concept of script. Syntactically, a script is sequence of phases and each phase can be described by five attributes. The grammatical combination of these elements may however produce any kind of pedagogical method, even those that have nothing to do with the idea of collaborative learning. On the one hand, the definition of scripts constitutes a promising convergence between educational engineering and socio-cultural approaches but, on the other hand, it drifts away from the genuine notion of collaborative learning. Will the fun and the richness of group interactions survive to this quest for effectiveness? The answer depends on the semantics of collaborative scripts: what is the design rationale, what is the core mechanism in the script through which the script designer expects to foster productive interactions and learning?

1. Introduction

The idea of constraining collaborative learning results from the empirical findings on the effectiveness of collaborative learning. These studies show that the effectiveness of collaborative learning depends upon multiple conditions such as the group composition (size, age, gender, heterogeneity, ...), the task features and the communication media. However, these conditions are multiple and interact with each other in such a complex way that is not possible to guarantee learning effects. Hence, effectiveness control migrated

from outside to inside, from pre-conditions to the actual collaborative processes (Dillenbourg, Baker, Blaye & O'Malley, 1995). Instead of tuning the conditions that (indirectly) determine the group interactions, scholars attempt to (directly) influence the interactions: augmenting the frequency of conflicts, fostering elaborated explanations, supporting mutual understanding, ... Collaboration can be influenced anticipatively, by *structuring* the collaborative process in order to favour the emergence of productive interactions, or retroactively, by *regulating* interactions, as tutors do. These two approaches are complementary.

Regulating collaborative learning is a subtle art. The tutor has to provide prompts or cues without interfering with the social dynamics of the group. Light human tutoring is a necessary, but expensive resource for computer-supported collaborative learning (hereafter CSCL). There have also been some attempts to design computerized tutors (Inaba & Okamoto, 1996; Barros & Verdejo, 2000; Constantino-González & Suthers, 2002). Another alternative approach consists in helping the group to regulate itself by providing it with some representation of its own process (Jermann, 2002; Dillenbourg et al., 2002) or with a trace of their interactions (Zumbach et al, 2002).

Structuring collaborative learning is achieved by *semi-structured communication interfaces* and/or by the application of *scripts for collaborative learning*. A collaboration script[1] (O'Donnell & Dansereau, 1992) is a set of instructions regarding to how the group members should interact, how they should collaborate and how they should solve the problem. When teachers engage students in collaborative learning, they usually provide them with global instructions such a "do this task by group of 3". These instructions usually come with implicit expectations with respect to the way students should work together. The teacher's way of grading collaborative work strengthens this implicit contract. A script is a more detailed and more explicit didactic contract between the teacher and the group of students regarding to their mode of collaboration. This contract may be conveyed through initial instructions or encompassed in the CSCL environment.

This contribution focuses on scripts for collaborative learning, especially for computer-supported collaborative learning. I focus on scripts for two reasons.

---

[1] I previously used to term 'scenario' to refer to what is now more commonly referred to as a script. Some colleagues, namel Hoppe, use the term 'scripting' to refer to the analysis, by the student, of the log file of their own interactions (Zumbach et al., 2002)

First, I was invited to Paul Kirschner inaugural address with the mission of presenting the Geneva school of CSCL and that scripts constitute one part of our applied research, namely applied in my own teaching. Second, the design of scripts is currently a convergent focus of the CSCL community (at least in Europe) and some critical thinking is always required whenever a research community converges on something. This critique is expressed in the title: do our efforts to make collaborative learning effective drift us way from the genuine idea of collaborative learning. Intrinsically, collaborative learning is an optimistic view, à la Rousseau: two learners, neither of them being very knowledgeable in the domain of study, would naturally gain knowledge by engaging in miraculous interactions. As Glachan & Light (1981) wrote, "can two wrongs make a right?" The recent evolution of CSCL leads collaborative learning scripts that are quite far away from this natural process and get closer to teaching methods. These pedagogical methods include social interaction episodes but can they still be described as collaborative? *Is it possible to blend two pedagogical traditions, collaborative learning and traditional instructional design à la Gagné, without losing that which makes 'natural' collaborative learning different from other teaching methods?*

2. Examples of CSCL scripts

Each author has his or her own understanding of what a CSCL script or scenario can be. I hence start by illustrating with a few examples of scripts I have used either in my own courses or in projects in which I was involved.

2.1. The Grid script

The best-known collaborative script is the Jigsaw: each group member has only access to a subset of the information necessary to solve the problem (Aronson et al, 1978). Therefore, no individual can solve the problem alone. Of course, group members could just forward information to each other, but the member who receives a body of information has to process this information, to become an 'expert' on that sub-domain, in order to use the information in the solution process. Thereby, information-sets define the role of each group member. There exists a broad range of variations of this script. In some cases, the one who plays role-X in a group sometimes meets those who play the same role in other groups and share experience. Hoppe and Ploetzner (1999) developed a kind of 'natural' Jigsaw in a CSCL environment. The environment includes a student-modelling component that categorizes students according to whether they rather apply qualitative or quantitative knowledge in physics

problem solving. Their environment then form pairs with one student from each category and provides them with problems that cannot be solved with only qualitative or with only quantitative knowledge. Another form of 'natural' Jigsaw can be obtained by grouping students from different backgrounds, for instance pairing a medical student with a student in psychology for constructing a therapy plan (Hermann, Rummel & Spada, 2001).

We implemented a variation of the Jigsaw, the Grid, in a master course on the theoretical bases of learning technologies (see figure 1). The course modules review different types of learning technologies: frame-based courseware, simulations, microworlds, …, For each module, students have to learn the key concepts of the domain and the underlying theoretical framework. The script runs as follows:

1. Groups of four students are formed, based on individual choices. They have to distribute four roles among themselves. Roles correspond to theoretical approaches and are defined by a notorious defender of this approach. For instance, in the first module on traditional computer-assisted learning, the roles are named Skinner, Bloom, Anderson and Saint-Thomas. The roles differ between each module except for the 'Saint-Thomas' role: his viewpoint is always to be sceptical with regards to the effectiveness of the educational software under study. To learn how to play a role, each student receives a few texts describing the related theory.

2. Each group receives a list of concepts to be defined. Examples of concepts appear in the cells of figure 1. They cover the key notions that teacher expects learners to acquire. The group distributes the concept definition work among its members. The teacher does not specify which role is knowledgeable for which concepts.

3. Each student writes a 10-20 line definition of the concepts that were allocated to him/her.

4. Groups have to assemble these concepts into a grid (see figure 1) and to define the relationship between grid neighbours. The often have to try many organisations of the concepts on the grid before are able to define all relationships. Two relationships are proposed: the symbol "<>" is used for dissociating between two similar concepts (namely 'false friends') and the symbol ">< " for relating to concepts that are apparently not related to each other.

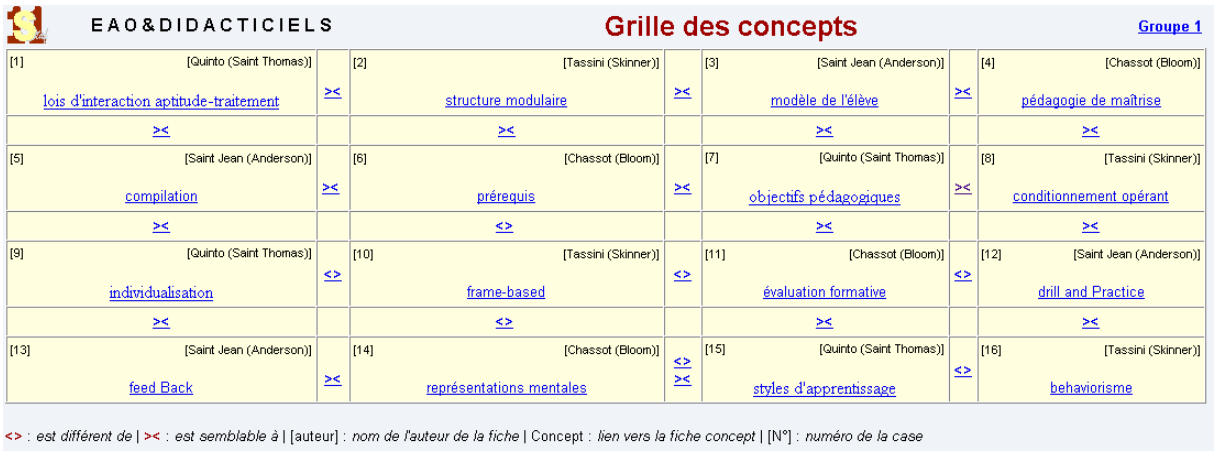

Figure 1: Interface of the GRID script (the students put two names in each cell, their own name and the name of the role their are playing).

Technically, the Grid is a simple html file in which each concept label and each relationship between two concepts refer to another file where the concept or the relation is explained. I did not yet carry a systematic evaluation of this collaborative script. Let me however make a few remarks that will be reused later on:

- The script is not fully collaborative: Phase 3 is cooperative (each student individually writes a text) while phase 4 requires for collaboratively building the grid.

- The design rationale of this script (and most Jigsaw scripts) is the complementarity of knowledge, i.e. that fact that no student can build the grid without collaborating with partners. When concepts A and B have been written by different students, writing the A-B link requires each person to read what the peer has written and, if needed, to interact with that peer.

- The ergonomics of the environment prototype were very poor, students having to edit too many html files. Several teams choose to meet physically and to build the grid with paper notes before drawing the table in html.

2.2. The ArgueGraph script

The goal of the 'ArgueGraph' script is that students relate courseware design choices with the underlying learning technologies. The script is based on a

simple multiple-choice questionnaire produced by the teacher. For each answer of each question, the teacher determines X and Y values that will be summed to compute the students' opinion in a two-dimensional space. This script includes five steps

1. Each student takes the quiz on-line. For each choice, the student enters an argument in a free-text entry zone.

2. The system produces a graph in which all students are positioned according to their answers. Students look at the graph and discuss it informally. The system or the tutor forms pairs of students by selecting peers with the largest distance on the graph (i.e., that are most different).

3. Pairs answer the same questionnaire as in step 1 together and again provide an argument. They can read their individual previous answer.

4. For each question, the system computes the answers given individually (phase 1) and collaboratively (phase 3). The tutor uses these data during a face-to-face debriefing session.

5. Each student writes a synthesis of all arguments collected for a specific question. The synthesis has to be structured according to the theoretical framework introduced by the teacher during the debriefing (phase 4)

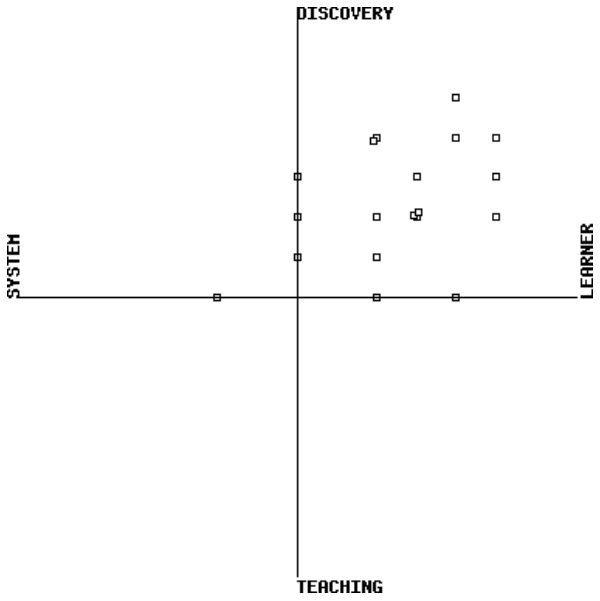

Figure 2: Graph representing individual answers. (Names have been erased)

We successfully used this script to teach the relationship between learning theories and the design of educational software (Jermann & Dillenbourg, 1999). It can be generalized to conceptual domains in which multiple theories co-exist. It leads us to a few remarks:

- The script integrates face-to-face and online activities.
- The script is not 100% collaborative: it includes a peer interaction phase (3), but also individual phases (1 and 5) and a collective phase (4). A collective phase involves all students in the class.
- The design rationale for this script is to create conflicts among students and engage them into interactions to resolve the conflict.
- We tested two versions of this script, one where all students were in the computer room and another one where they used the system at distance for phases 1 to 3. The two versions used different CSCL environments. The latter did not work very well for two reasons. First, the interface for phase 3 enabled students to avoid conflict resolution by weighting their degree of agreement with each proposal instead of being force to choose one and only one proposal. Second, the pairs who argued (phase 3) long before the debriefing (phase 4) were much less involved in the debriefing discussion than those who argued just before. In other words, the efficiency of this script is not only influenced by the choice of activities but also by factors such as the ergonomics of the environment (Jermann & Dillenbourg, to appear) and the timing of phases, not to mention the quality of the questionnaire

2.3. The UniverSanté Script

This script (Berger et al, 2001) was used in medical education, more precisely in teaching community health. It has been applied to a course jointly given at the Universities of Geneva (Switzerland), Beirut (Lebanon), Monastir (Tunisia) and Yaounde (Cameroon). The students are divided in five thematic groups: AIDS, cancer, infectious diseases, cardiovascular diseases and trauma related to accidents. Each thematic group includes four students of each country (16 on the whole) and a tutor. The script includes seven phases: starting from a clinical case (phases 1 and 2), students address the main issues of public health (phases 3 to 5), they tackle some methodological issues in epidemiology

(phases 5 and 6) and finally (phase 7) address strategies to cope with the main public health problems.

1. Each thematic group (16 students) is divided into two sub-groups (eight students, 2 per country). Each sub-group receives a clinical case. For example, the first 'cancer' sub-group works on the case of a woman with breast cancer whereas the second 'cancer' sub-group receives a case of a man with lung cancer. Each sub-group discusses the case in a specific forum. The tutor stimulates and guides the discussion in order to stimulate the students to identify and discuss the public health elements of the case. For example for cancer, the tutor asks questions like: What elements could have contributed to develop that cancer? How the patient could have been informed about the risks he took?

2. A synthesis of the elements identified by each thematic group is presented during a face-to-face debriefing meeting in each country.

3. Within a thematic group, the students of each country create a fact sheet describing the status of this public health problem in their country. For example, the four Swiss students in the cancer group create a fact sheet "Cancer-Switzerland ", which they enter in the database through an online form.

4. The students of each thematic group discuss the differences and the similarities between the fact sheets of the four countries in the forum.

5. All fact sheets are commented on during a face-to-face debriefing meeting in each country. The tutor prompts the students to identify any needs for clarification or refinement concerning the way in which statistical data were collected, treated or presented for the fact sheets.

6. Students modify their fact sheet according to the methodological comments received in phase 5.

7. Each thematic group is divided into two sub-groups working on the cases they studied during phase 1. Each sub-group proposes a health strategy to cope with the problems. The students enter their strategy (objectives, actions, resources, evaluation) in the knowledge base through an online form.

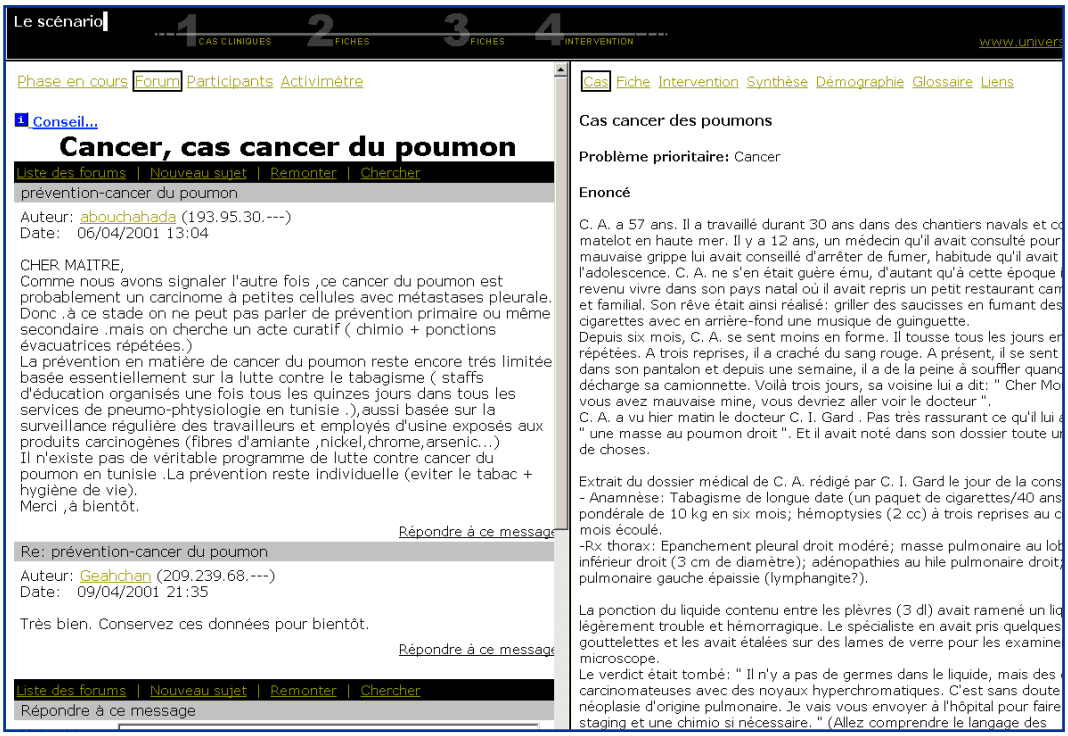

Figure 3. A screen from the Universanté environment

The application of this script leads to a few remarks:

- The script integrates face-to-face and online activities.
- The script defines multiple social circles: thematic groups, clinical case groups and national groups. At different phases, the learner discusses the course topics within one or another of these circles.
- The design rationale of this script is to play with differences between learners, both natural differences (e.g., public health problems and policies differ in each country) and differences created by the designers (e.g., the difference between the two cases of AIDS, one case of contamination by sexual contact and one by birth). These differences constitute an attenuated version of the conflict-solving paradigm.
- The script was too complex, both for the learners and for the tutors. I will address this issue in section 4.3. A new course has been run recently with a simplified version of the script.
- The role of the tutor was prominent both in all discussion forums and face-to-face debriefing faces.

2.4. Other examples

One could list numerous CSCL scripts. I simply mention a few other examples that will be useful for further discussion:

- The MagicBook: We used this script with Laurent Dubois in a project with primary schools following an expedition in Antarctica. In this script: (1) The teacher writes the beginning of a story; (2) All participants read this first chapter; (3) All participants write a second chapter and propose it as a continuation of the story; (4) Proposals for the next chapter proposals are read by the participants who vote for their favourite; (5) The elected chapter becomes the official chapter 2. The script iterates on phase 2. The 'participant' to this script can be an individual learner or, in our experience a whole class of kids.

- The Courseware Design Studio is an adaptation from the Phase-X script (Engeli, 2001) for supporting project-based learning. The project process is segmented into phases. In each phase, all teams put their intermediate product in a shared space. In the next phase, a team is allowed to borrow the work produced by another team and to continue its work from it. In our application, the project is to build courseware and the phases were goal definition, content analysis, activity design, and so forth. The rationale for this script is that the shared space would create a kind of permanent idea-seeding. While it seems to work very well in 3D-design projects, our students had difficulties exchanging intermediate results in their design process.

- Problem-based learning (Barrows & Tamblyn, 1980) is a rather standardized script that has been used in a large variety of training situations.

- Another well-known script is the 'reciprocal teaching' approach set up by Palincsar and Brown (1984): one peer reads a text paragraph and the other questions him/her about his/her understanding, for the next paragraph the roles are shifted. This is not strictly speaking a collaborative script as it was used with pairs made of an experienced tutor and a student with reading comprehension difficulties. The outcomes of their experiments were so positive, however, that the script might be extended to more symmetrical situations. Many variations of this script exist such as peer tutoring (O'Donnell &

Dansereau, 1992; Fantuzzo et al, 1989) or peer teaching (Reiserer, Ertl & Mandl, 2002).

3. *The syntax of CSCL scripts*

A large number of scripts can be built from the combination of a limited number of components, in the same way that a language is made of words and grammatical rules. This analysis provides the bases for such a formal grammar. I informally specify the elements and rules of this grammar with the goal that they can later on be turned into a more rigorous notational scheme like XML.

A script is a story or scenario that the students and tutors have to play as actors play a movie script. Most scripts are sequential: students go through a linear sequence of phases. Some of the presented examples (Phase-X, the MagicBook or Reciprocal Teaching) are defined in an iterative way, but from the student point of view, they are run as a linear sequence.

Script = [phase1 phase2 phase 3 …]

It is possible to design non-linear scripts, for instance to enable some groups to skip some phases, but as it will be explained in section 4.3, a main design concern is to keep scripts simple, and easy to adopt by the learners.

Each phase of the script specifies how students should collaborate and solve the problem. This requires five attributes: the task that students have to perform, the composition of the group, the way that the task is distributed within and among groups, the mode of interaction and the timing of the phase. These five attributes are now analysed one by one. The third attribute, the distribution of the task over the group, does not require a specific slot as it will be expressed by the syntax of the task and group descriptions.

Phase = [Task Group Mode Timing]

3.1. Task definition

A phase describes what students have to do. This *task assignment* can be described as a triplet [input activity output]. In the Grid script, the input of phase 3 includes texts and concepts, the activity is to write definitions and the output is the set of concept definitions. In Courseware Design Studio, the input of phase 2 is the learning objectives of the future courseware, the activity is content analysis and the output is a concept map of the contents to be taught. In ArgueGraph, the input of phase 3 is the question and the two

individual answers, the activity is argumentation and the output is the joint answer and the argument that supports it. A phase might include multiple activities, depending on the granularity of the description, but a script is simpler if each phase is clearly associated with one main activity, even if it is a complex one (e.g. writing a computer program solving a clinical case, …)

Task = [input activity output]

In the CSCL scripts presented, and despite the fact that this concerns computerized environments, only some activities are computer-mediated; there is no reason to exclude other activities. CSCL scripts are not restricted to 'pure' distance education, but they support blended presence + distance education approaches. Non-mediated activities are often integrated in the script by the fact that the output of the activity is introduced in the system, since the output of a phase generally becomes the input of a later phase (often the next phase).

$Task_{n+1}$ = [$output_n$ $activity_{n+1}$ $output_{n+1}$]

The reuse of the previous output integrates successive tasks within a coherent whole. The storage and management of these intermediate products contributes to the added value of a CSCL environment. However, it requires building elaborate web sites in which the pages are dynamically generated from database contents and in which students' behaviour or products are stored in the database. Since products have to be associated with their authors (individual or groups), these scripts also require user authentication and the possibility of defining multiple groups.

An operational script should include task *completion criteria*. These criteria can be defined with respect to the activity (e.g., answering all questions in the ArgueGraph script), as conditions on the output (e.g., produce a text of 300 words in the Grid script) or with time limits (see section 3.5). When the criterion can be computationally checked, the system may decide to move to the next phase. When the criterion requires ,validation by the tutor, the CSCL environment must be enriched with some *workflow* functionalities such as forwarding work for tutor validation or notifying the learner that the feedback is available.

3.2. Group definition

The *group size* varies between 1 and n, n being the total number of students in the population considered. In the samples above, there are basically four group sizes: individual work (n=1), small groups (n=2 as in ArgueGraph to n=4

in the Grid), medium groups (n=8 in UniverSanté), a whole class (n=20 in ArgueGraph, n=60 in Universanté and n=120 in Phase-X) or even a set of classes (in the MagicBook script used in the Antarctica experiment).

Some scripts, such as the MagicBook, can be applied with various group scales: each chapter can be written by an individual learner, by a group of learners or by a whole class as in our experience.

I want to stress the fact that, often the group size varies between phases. This discriminates these CSCL scripts from traditional group work that usually creates group for the whole activity. This variable group size acknowledges *the role of individual reflection within group activities*, a role that has been somewhat neglected over the last years. Moreover it enables us to integrate class-wide debriefing activities which are very important, namely in order to make sure that all groups have acquired the target knowledge.

When the group is made (n>1), the script should specify the *criterion for group formation*, i.e. how the group members are selected. By default, the group members mutually select each other based on affinity or other practical criteria. Actually, our adult students favour practical criteria such as where other group members live and when they are available. Some of the scripts described in the previous section specify the criterion for group formation. In the ArgueGraph script, the criterion is differences of opinions estimated as the distance between the individual positions on the graph. In the UniverSanté script, large groups are made of small groups based on their differences: different countries, different clinical cases and different public health problems. In the Hoppe & Ploetzner (1999)'s inverted Jigsaw script, the criterion is the complementarity of knowledge.

The group formation criterion can be internal or external to the script.

- *External criteria* distribute students into groups on the basis of some students' characteristics that pre-exist the CSCL activity: friendship is the most used criterion, but also level of expertise (e.g., pairing a good reader with poor reader in the ReciprocalTeaching script), domain of expertise (e.g., pairing a student in psychology and medical student – Hermann et al., 2001), or the geographical or cultural background (e.g., in the UniverSanté script, some phases involve students from different countries while other phases occur between students from the same country),. In this case, the *group is defined in extension*: the

script grammar must specify the profile of each group member such as [low_reader high_reader] or [Switzerland Lebanon Tunisia Cameroon].

- *Internal criteria* distribute students into groups based on the students' behaviour or products that have been collected in a previous phase of the script. These intra-script criteria contribute to added value of CSCL, as computers offer functionalities for collecting behavioural data and products, for analysing them (at least with formal criteria) and for applying group formation criterion even to large number of students.

These criteria not only determine the constitution of a group, but also determine the *differences between groups*. These differences particularly enrich the synthesis or debriefing phase of the script. While teachers are often concerned by the group composition (should I mix girls and boys, good and poor learners, ...), the heterogeneity between groups offers innovative ways to engineer collaborative learning. The teacher may select which class to collaborate with according to the very notion to be taught: different geographical concepts can be approached through interactions with classes living at a very different latitude, water quality with a class living 200 kilometres upstream or downstream the city river, and so forth.

3.3. Distribution

The distribution of a global process over different individuals or groups is a mechanism commonly exploited in CSCL scripts; it's almost the essence of these scripts. I review different ways to distribute a global process: distributing the input of the activity versus distributing the activity and distributing over the members of a group versus distributing over different groups.

Input distribution and/or activity distribution: The Jigsaw script defines an input distribution, providing each member of the group with different information. For instance, in the Grid script, each group member receives a different set of texts to read. Their activity is more or less the same, to read the texts and to write concept definitions. The Reciprocal Teaching script distributes the activity on the cognitive / metacognitive axis: both actors are concerned with the same text but one has to read and understand while the other has to monitor the other's understanding by asking questions. Of course, the input distribution may induce an activity distribution: if two learners have to estimate the volume of an oil reservoir, one receiving seismic data and the other well data, the processing of these different inputs implies

different methods for volume estimation. Conversely, the activity distribution[2] may lead students to pay attention to only a subset of the input, for instance if two medical students receive the same patient file but one has to play the role of the cardiologist and the other the role of the anaesthetist.

One may object that when the different activities are independent from each other, the learning phase should be described as cooperative instead of collaborative (at least in my definition of collaboration – Dillenbourg, 1999). I don't see any reason to exclude cooperative phases within a CSCL script. The division of labour varies across activities; there is no formal threshold that would discriminate cooperation from collaboration. Moreover, distinct cognitive activities as in the reciprocal teaching script still create a collaborative situation, as they require a close cognitive coupling between the peers.

Intra-group distribution versus *inter-group distribution.* The examples mentioned so far describe the distribution among the individuals of a group. This is not the case in the UniverSanté script where the set of public health problems has been distributed over different groups. Each thematic group considers a different public health problem (cancer, aids, …). Intra-group content distribution is more frequent than inter-group content distribution since the different groups pursue indeed the same learning objectives. Inter-group differences are nevertheless acceptable when they provide various instances of the target concepts: in the UniverSanté script, students are not learning about cancer or AIDS, but about public health issues that are illustrated with the cancer or AIDS. The diversity of instances supports the generalisation of concepts during the synthesis or debriefing session. A script may include both intra-group and inter-group content distribution, as would be the case if an architecture teacher asked each student group to choose a different shopping centre in the city and each student in the group to critique it from a specific viewpoint, the client, the company or the employee.

The grammar should reflect these different modes of distribution through simple syntactical rules, for instance [task [group]] could mean distributing the task on the group members while [group [task]] would describe distributing the task on different groups. In some cases, the input or activity

[2] I deliberately avoid the term 'roles' as it is may refer both to input and activity distribution. If I ask two students to read about Piaget and Vygotsky respectively and then to play these roles in an argument on cognitive growth, we are in an input distribution approach. If they play the role of a salesman and a customer in a sales training course, we are in an activity distribution mode.

distribution is induced 'naturally' by the group composition (e.g., pairing a nurse and a doctor, a or a student in Switzerland with a peer from Lebanon, …) as explained in the previous section. This distribution is then implicit to the group composition criterion.

I argued (Dillenbourg, 1999) that what discriminates collaboration from cooperation is less the degree of division of labour than the rigidity of this division. Other authors (Burton, 1998, Soller et al, 1998) argued that rotation of roles within a group is beneficial to collaborative learning. These *rotating scripts* can be simply described by including permutation rules in the grammar.

3.4. Mode of interaction

Phases differ with respect to the mode of social interaction. The mode varies according to the size of the group, one cannot expect joint problem solving in large groups. There is an infinite number of modes of collaboration; I just point out a few features that are especially important or relevant.

All CSCL scripts mentioned above integrate *distant* and *co-present activities*. There is no reason to design scripts that exclude face-to-face activities except when the students cannot meet physically at all. Face-to-face phases increase the robustness of the script; the rich interactions compensate what could not be exchanged through remote communication. Our examples of scripts concern adult students who have tight time constraints. As we have few opportunities for face-to-face meetings, the art of CSCL script-design is to limit face-to-face to the critical phases. Let me stress that in the three scripts presented in section 2, the co-present activities are computer-based: the students are side by side and do not communicate via the computer, but they act together on a computerized task. Of course, the design of the computerized tool (the graphical representation, the language used, …) has an impact on the social interactions. The CSCL software must be taken in the sense of an "environment" that supports the whole script activities, even if none of them involves computer-mediated distance communication.

In remote interactions, the mode of interaction is determined by all of the classical features of communication media: media richness (textual, graphical, audio, video, …), *synchronous versus asynchronous* (and everything in between), one-to-one versus one-to-many or many-to-many,…. Synchronicity is common place, but worth stressing as it constitutes an important design trade-off: synchronicity enables rich interactions, but introduce high

organisational constraints for the target audience. Therefore, I recommend scripts that use synchronous activities for the key phase(s), but relax constraints on the other phases by using asynchronous interactions.

The scripts presented focus on intra-group interactions. Some scripts add *inter-group interactions*, for instance when one class produces a mathematics problem to challenge other classes.

This section needs the construction of a taxonomy of modes of interaction that goes beyond the scope of this contribution. Among the multiple dimensions of this classification, an important ergonomic feature is the *degree of integration of task interactions and social interactions*. Integrated task/communication software enables for instance two medical students to attach notes to specific items of the patient file under scrutiny. Communication does not occur in the vacuum, but "on" the object under scrutiny. One may expect t hat integrated task / communication interfaces foster task-focused social interactions, but this remains to be proved. However, this integration has a drawback: students are used to a particular chat or email software across multiple activities and are hence usually reluctant to use different communication tools for different activities.

Finally, the interactions do not only occur between students, but also with the tutor. The *tutor* has two important roles is the script phases.

- *Regulating students' interactions*. None of the scripts mentioned here is tutor-free. The degree of tutor intervention varies along the script from none (e.g., in the pair argumentation phase of the ArgueGraph script), to moderate (e.g., in the discussion forums of the UniverSanté script) and even to tutor-centric phases (e.g., in the debriefing phase of each of these scripts). When the interactions among students are computer-mediated, regulation becomes difficult (too many messages and messages spread over multiple places) and faces obvious privacy obstacles. Inventing tools that facilitate the regulation of on-line group interactions is an important item on our research agenda.

- *Feedback on the phase output*. In many adult education scripts, this feedback is the main source of interaction and mostly via asynchronous interactions: groups upload the output of their activity on the server and the tutor associates the feedback to the object or sends the feedback directly to the team. As a matter of fact, the management of feedback can be very complex as groups have different

rhythms. Therefore, a CSCL environments requires follow-up functionalities enabling the tutor to view in one snapshot which teams have been posted their work, which teams have received a feedback and which ones have updated their work according to this feedback.

3.5. Timing

A simple but fundamental role of a CSCL script is the timing of collaboration. A script is a sequence in which phase n ends before phase n+1 begins. In remote education, the students often loose the time referential that is provided by traditional weekly courses. In distance education or blended education, the lack of routine raises time management problems for many students. Although distance education builds upon individual organisation, my colleagues and I had paradoxically to introduce more and more time constraints in our on-line distance courses. A global time constraint (when the script has to be completed) is often not enough. It was necessary to specify the timing phase per phase, turning the script into a *time-management prosthesis*. The timing of a phase can be described by its duration (e.g., 2 hours) or an output delivery deadline (e.g., "post your report by 12.12.2002"), usually combined with activity completion criteria (e.g., to give all answers, to post 3 messages, …). The automatic management of deadlines (to sending reminders and warnings, to notify delivery date) is one feature that one might expect from CSCL environments.

4. The semantics of CSCL scripts

"The red sister of the train drinks the democracy of my horse". A grammatically correct sentence can be meaningless. The grammar of scripts, presented in the previous section, may produce meaningless combinations of activities. It describes any pedagogical method in which the designer includes social interactions since the grammar was deliberately left open to a variety of non-collaborative activities. As pointed out earlier, an interesting feature of the illustrated scripts is precisely that they do not stick to a narrow definition of collaboration: they include individual activities and collective (class-wide) activities as well. Now, the more open the grammar is, the more important it is to reflect on the pedagogical meaning of the script. What is the idea behind the script? What are the pedagogical values? Is the script 'playable' by the students? Is the script specific to the content to which it is applied?

4.1. Design rationale

What distinguishes a CSCL script from any sequence of phases is that its design reflects a hypothesis. This hypothesis relates the social interactions supported by the script with respect to the learning objectives. If the hypothesis was not about learning from social interactions, one could hardly speak of a CSCL script. The script is built around a core mechanism and the hypothesis is about how this mechanism fosters some interactions or inhibits other and about how the expected interactions are supposed to produce learning effects. The ArgueGraph script is expected to create conditions in which conflicting views are confronted. The Reciprocal Teaching script is designed to foster mutual regulation. In the UniverSanté script, the design rationale was to promote abstraction, as Schwartz (1995) showed that bridging individual representations leads to more abstract representation. The Jigsaw approach is expected to create intensive interactions and to prevent the so-called free-rider effect (Salomon & Globerson, 1995) since one group member is not able to do the task without taking his/her partner contributions into account.

The design rationale is the spirit of the script. It inspires the construction of the key learning activities, the key phases of the script. These key phases are then complemented with phases aiming to enhance or consolidate the script: conceptual introduction, debriefing, synthesis, intensive practice, transfer, socialisation and evaluation. These complementary phases illustrate the blending of the collaborative learning tradition and traditional educational engineering.

4.2. Coercion degree

The scripts vary according to the degree of freedom that the learners have in following the script. The degree of coercion is a continuum, but several levels can be emphasized:

- *Induced scripts*. The communication interface induces interaction patterns, it implicitly conveys the designer's expectations with respect to the way students should tackle the problem and interact with each other. This low degree of coercion is elegant but often not sufficient to significantly influence the collaborative processes.

- *Instructed scripts*. Students receive oral or written instructions that they have to follow. The coercion is higher than in the induced script since the teacher expectations are made explicit, but they can of

course be misunderstood, incorrectly applied, forgotten or completely ignored.

- *Trained scripts*. Students are trained to collaborate in a certain way before using them it in a real learning situation. A teacher who plans to use a brainstorming script several times might devote an initial session to train students in brainstorming methods (no premature criticism, …). The degree of coercion is higher than instructed scripts since the teacher may control the student's understanding and application of the collaboration rules. Experimental studies on script effectiveness require subjects to be trained.

- *Prompted scripts*: The system display cues that encourage the learners to take their respective role (Weinberger, Fisher & Mandl, 2002). Their system delivers cues as text messages in the discussion board used by the students for discussing cases. The cues were supposed to lead students to take specific roles such a 'analyser' or 'critique'.

- *Follow-me scripts*. Students interact with an environment that does not allow them to escape from the script. In the ArgueGraph script, at phase 3, the students have to agree on one and only one answer, the interface simply did not allow the students to answer in another way. Moreover, the system does not allow them to move to phase 2 as long as they have not completed phase 1.

Coercion concerns several aspects of the script: the choice of teammates can be open or constrained by the system as in the ArgueGraph; the timing of an activity can be fixed or left open; the interactions between learners can be free or constrained; the tutor can be intrusive or keep a minimal intervention strategy, etcetera. Choosing the appropriate level of coercion is the oldest educational design trade-off. A certain degree of coercion is required for efficiency reasons, but too much might be in contradiction with the very idea of collaborative learning and might decrease student motivation. This design dilemma is salient in the work on semi-structured communication interfaces. Their purpose[3] is to bias social interactions towards a specific interaction

[3] Another purpose of a semi-structured communication interface was to ask learners to classify their own dialogue moves, because of the metacognitive benefits that were expected from this reflective process and for the methodological advantages of collecting user-coded interaction transcripts. However, the overload of this self-coding activity is such that users get bored very

model, basically a dialogue model. A bias may also be conveyed by a graphical representation as in Belvédère (Suthers et al, 1995).

A dialogue model includes a set of primitives or communication acts and a set of dialogues rules that specify which acts can be 'legally' performed after another one. For instance, the Dialab environment (Pilkington et al., 1992) offer a communication tool based on Mackenzie's (1979) dialogue game. The primitives concern the task (e.g., "I suggest to increase power in engine 3"), the interactions (e.g., "I don't understand, please explain") or the collaboration (e.g., "Please do the next step"). The users select these primitives in menu lists or buttons sets. The dialogue rules enable deactivation in user-B's menus those dialogue acts that cannot – within the model - follow user-A's last dialogue act. For instance, "I suggest to increase power in engine 3" could 'legally' be followed by 'Ok', 'I disagree" or '"Why?", but not by "What do you suggest?" or "Let's consider engine 2".

The degree of coercion of these interfaces also vary as to whether dialogue acts are partly or fully defined. Partly defined dialogue acts are, for example, sentence openers (e.g., "I propose to ..."). The user has to complete the sentence. Partly defined dialogue models include a text entry area where the user can interact with free text. The dialogue rules may be imposed with varying degrees of coercion. For instance, a high coercion interface deactivates the buttons including a speech act that cannot legally be performed at the next turn.

The design trade-off is obvious. Except for a few tasks, it is difficult to define a highly controlled communication interface. How does one anticipate everything (useful) that learners would need to say to each other? Would a fully controlled interface support a meaningful dialogue or lead to a very artificial situation that has not much to do with collaboration?Experiments have shown that lower coercion interfaces have a weak impact on interactions, beyond the mere reduction of off-task discussion (Baker & Lund, 1997; Jermann & Schneider, 1997). Users use interface components in a way that is not necessarily consistent with the meaning given by the designer (Baker et al, 1999). Students may even complete sentences in a way not consistent with the sentence opener. The work on semi-structured interfaces is promising, as it turns pedagogy into subtle ergonomic choices, but empirical studies have failed to prove that the interfaces have an important structuring effect on interactions (Veerman &

---

quickly and tend to type anything, reducing both the metacognitive and the methodological advantages of this approach.

Treasure-Jones, 1999). This lack of empirical proof is probably due to the fact that these semi-structured communication tools only address one facet of collaboration processes while collaboration actually involves three concurrent processes, which are neither independent of each other, nor identical:

- The communication or interaction process, i.e. the way group members communicate with each other (e.g., verbal dialogues).
- The organisation or coordination processes, i.e. the way group members establish shared goals, distribute task, coordinate each other, regulate mutually, and so forth.
- The task level or problem solving process, i.e. how the group performs the task, which strategy is elaborated, which understanding is required, et cetera.

The limitation of semi-structured communication tools is that they address only the first process, communication. They also introduce an ergonomic bias, i.e. guide learners to choose the most effective way to perform an action, but the bias will not decide which action has to be performed. The way learners communicate is still less influenced by the interface than by what they have to say to each other.

The interest of the CSCL scripts is to *distribute the coercion load over these different processes*, for instance to augment argumentation through group formation (ArgueGraph script), or to impart on the verbal interaction through the objects of the task interface (Belvédère: Suthers et al, 1995). A level of coercion that would be unacceptable when concentrated on a single component can be obtained by using multiple smaller constraints distributed on the task, the division of labour, the timing, the tutoring, and so forth. In other words, scripts have the advantage of covering globally the collaboration process.

4.3. Appropriation

The description of the UniverSanté script clearly shows that we went too far in the complexity of the script, namely the number of different forms of grouping (by public health issue, by clinical case and by country). Both the students and the tutors complained about it. A simpler version of the script is currently in use. We learned (Berger et al, 2001) that scripts should kept as simple as possible such that all actors - students and tutors - are able to appropriate these scripts. Two levels of appropriation can be discriminated.

The first level of appropriation is *adoption*. Students and tutors have to understand the script, i.e. to know what they have to do, without additional difficulty or overload. Moreover, to claim that actors have adopted the script, they should play the script more or less as the designer intended it to be played. The cognitive load of memorizing and applying the script can be reduced in two ways. First, it is possible to help students to give meaning and memorize the script by embedding it within a story. A second way to reduce memory load is to offer a representation of the script or any navigation tool that help learners to know where they are and what is left to do.

The second level of appropriation is *internalisation*. The goal of the Reciprocal Teaching script is that the reader, in the end, is able to play the script individually; that is he/she is able to simultaneously play both the role of reader and the role of regulator. Similarly, the goals of the ArgueGraph script was that students could later on, individually, consider multiple arguments, for or against a design decision, whatever the learning theory that is behind this argument. This Vygotskian postulate is that the learners would individually and internally replay the distributed cognitive processes in which they participated during the script. Only a few scripts are build upon this internalisation hypothesis. To-be-internalised scripts require that the way the cognitive task is distributed among individuals is compatible with individual cognition. Theoretically, one might expect that rotating scripts[4] should facilitate the internalisation of multiple roles. As far as I know, this remains to be proved.

4.4. Generalisability

If a script works well in a domain, it is a sensible idea to try to apply it to another domain. How easily a CSCL script can be reused on a different teaching / learning domain is a concern both for teachers and developers. The definition of a formal grammar enables one to dissociate the script content from the script structure and thereby to define domain-independent scripts that are, at least technically, reusable to a variety of domains. The interest of course is to produce new CSCL environments by editing only the content-

[4] Scripts in which the group members shift roles between phases

specific parts of it. The goal behind the grammar definition is namely to facilitate this authoring[5].

One has to discriminate the technical difficulty and the pedagogical relevance of generalizing scripts. Technically, software reuse and domain-independent content ontologies are complex issues. From the pedagogical viewpoint, *script generalisability is bound by its design rationale,* which restricts reusability to objectives where the hypothetical core learning mechanism remains plausible. The abstract definition of the Jigsaw script and the Phase-X script makes them applicable to a broad variety of learning objectives. The Grid script is a more specific instance of the Jigsaw script that fits with conceptual knowledge but not with procedural knowledge. The ArgueGraph script can be reused in a variety of conceptual domains, but fits mostly with domains where there is no definite right or wrong answer. Finally, the UniverSanté script concerns also arguable conceptual knowledge, but has been specifically designed for domains where the difference between social contexts (countries) is a central point. Public health is an instance of this domain.

Generalisability is a continuum, and the three script examples have been presented in decreasing order of generality. Generalisability is multidimensional. The first dimension is the target knowledge. It is important to stress that in the three quoted examples, the relevance script is not defined by traditional school disciplines (mathematics, chemistry, language, …), but with other characteristics of the knowledge to be acquired (procedural versus conceptual, clear-cut versus discussable, …). The other dimensions of script generalisability are its adaptation to the target audience (age, mobility, acceptability of the coercion degree, …) and its compatibility with the course organisational constraints.

5. Conclusions

Scripts are supposed to enhance the effectiveness of collaborative learning: Do they? This contribution did not review literature on the effectiveness of scripts because the concept of script encompasses such a broad range of methods that it would be non-sense to speak about the effectiveness of CSCL scripts in general. The present effort to specify the features of a script is a pre-condition to establish what makes scripts effective or not.

---

[5] This work also aims to augment software reuse by enabling advanced authors to built new scripts by assembling the components of existing scripts.

To enhance the effectiveness of collaborative learning activities, scripts integrate these activities within more traditional instructional sessions. As the script examples have showed:

- Scripts enable to integrate activities that were often separated: individual, cooperative, collaborative and collective activities.
- Scripts enable to integrate co-present activities and computer-mediated activities.
- Scripts often include an important role for the tutor.
- Scripts introduce a time frame in distance education where students often lack landmarks for their time management.

However, scripting collaboration has not only advantages; it also raises several risks:

- *Disturbing 'natural' interactions*. When a learner needs to make a dialogue move A, if the system only offers interactions B or C, either the learner will fail to say what he wanted to say or pervert the system (e.g. misuse B to say A). If similar interaction breakdowns occur frequently, they may spoil the whole collaboration process. Of course, the purpose of semi-structured interface is precisely to influence 'natural' interactions. For instance the ArgueGraph works better with the interface that forces students to choose one and only one answer, compared to the interface that allow them to find a natural consensus. In other words, this script makes the collaboration more difficult (not for the sake of difficulty but for supporting shaping interactions). This difficulty cumulates with the task intrinsic difficulty up to a level where the group may not be able to or not be willing to interact anymore with the system. This risk especially concerns scripts that support very specific interactions (versus scripts that globally specify phases) and have a high degree of coercion. Of course, scripts intend to shape social interactions, but at the same time, they should be *malleable* enough to permit students to adapt the script to their mode of collaboration.
- *Disturbing 'natural' problem solving processes*. A script usually segments a global task into a sequence of activities. In the Courseware Design Studio, this segmentation was a problem for students who had a more holistic approach. Of course, the whole purpose of this

segmentation is precisely to turn an unstructured design task into a clear sequence of activities, but some students had difficulties to adapt themselves to such an analytical approach. Moreover, the script proposed a linear sequence of phases while courseware design is – except in some textbooks – not a linear process. Some students rejected the artificiality of this 'linearisation'. The Grid script also introduces a high degree of coercion with respect to the task: students could find easier to draw a free concept map than to arrange concepts on a two dimensional grid. Again, this constraint was an explicit design decision but, at some point, this coercion might be incompatible with the students' cognitive processes. Over-scripting may make the task impossible and spoil the student motivation.

- *Increasing cognitive load*. Scripts may interfere with the main learning process by augmenting the learners' cognitive load in two ways. On the one hand, the load is increased by the necessity to understand, memorize and execute the script (see section 4.3). This issue appeared clearly in the UniverSanté script. On the second hand, the script may force the groups to interact and solve the problem in a non-natural way, to invent strategies to be able to *collaborate despite the script*. These strategies increase the cognitive load as well.

- *'Didactising' collaborative interactions*. Collaborative problem solving triggers natural interactions: a peer student asks a question because he wants to know the answer, he negotiates the meaning of a concept because he wants to resolve the conflicting interpretation of some phenomenon, … At the opposite, a teacher usually asks questions whom he already knows the answer and negotiates concepts for which he owns the right definition. The learners knows that these weird interactions are part of a didactic contract in which each actors plays its role. A danger of scripts, especially scripts that specify questions to be asked, is that interactions are played like in the teacher-learner game and hence miss the engagement that can be obtained when the listener really needs our explanation.

- *'Goalless' interactions*. Collaboration is usually not a linear story, but a dynamic process that is regulated by a shared goal. Let's consider the case of the grounding process. The degree to which two interlocutors should understand each other is partly determined by interaction rules (such as Grice's maxims), which regulate interactions in the same

manner as a CSCL script does. But, more importantly, the grounding criterion is determined by the team goals: students will make the effort necessary to reach the level of shared understanding that is required to solve the problem (Clark & Brennan, 1991). The same statement can be made about the need for mutual modelling and for mutual regulation. These important cognitive processes are triggered when they are required by the goals. The risk of scripts is twofold. First, they may interfere with the dynamics of goal achievement; second, they may prevent the team to establish shared goals. Shared goals are often referred to as an important criterion to define collaboration (Dillenbourg, 1999), but it is a challenge for a teacher to specify pedagogical goals that students adopt as their own goals. The more the scripts segments collaboration into sub-processes, the more it seems difficult for the team to adopt/choose shared goals and organise themselves to each it.

CSCL scripts are very different, they vary along a continuum: some are really close to free collaboration augmented by some light support, while others are nothing else than a traditional well-controlled pedagogical method with a collaborative painting. By emphasizing these risks of the latter, my point is not to criticize scripts for the sake a being at counter stream, but raise awareness of both sides of the coin. On the positive side, the notion of script creates an interesting bridge between collaborative learning and traditional instructional design. It brings closer two theoretical lines, respectively socio-cultural approaches mastery learning. On the negative side, scripts may lead to introduce fake collaboration. Scripted interactions may appear like a negotiation but under the surface, lack of any reason for the learners to negotiate meanings. Learners may ask scripted questions as they repeat a song, without convincing the explainer that his explanation is needed. Scripted collaboration may appear superficially as genuine collaboration, but may fail to trigger the cognitive, social and emotional mechanisms that are expected to occur during collaboration.

The purpose of this text is neither to define the orthodoxy of what should deserve the 'collaborative' label, nor to claim that all scripts have to be collaborative. Non-collaborative scripts are of course relevant to some learning objectives. This framework aims to support meaningful engineering: if the designer chooses to build a script that can hardly be described as collaborative, it's fine as along as this is an explicit design choice. Making

designers aware of their choices is especially relevant for our work. We[6] are building environments in which teachers edit script components, such as those listed in section 3, and assemble them into a new script. The risk of loosing the meaning of collaborative learning would be high if script building was purely syntactical. This is why the semantics of scripts have been emphasized in section 4. From the designer's viewpoint, a script remains within the 'collaborative learning' philosophy if the script design rationale calls upon social interactions as core learning mechanism, not simply as an add-on to individual activities. The co-construction of a shared understanding should be part the design rationale. Now, the real issue is whether the script remains collaborative from the student's viewpoint; this obviously requires further empirical research!

The current excitation of CSCL scripts looks a bit like the quest for the gold method, the magic script that will prove to be super-effective across many domains. I believe more into the construction of very specific scripts which can, through experimentation, reveal why they are or not efficient. Once efficiency is understood, the script can then be progressively generalized to other domains where they will be experimented again. Our challenge is not the golden script but the understanding of why some scripts are effective.

6. Acknowlegments

This paper first appeared in P. A. Kirschner (Ed). Three worlds of CSCL. Can we support CSCL (pp. 61-91). Heerlen, Open Universiteit Nederland. Thanks for their autorisation to reprint it. The scripts mentioned in this contribution have been developed with Patrick Jermann, Elia De Iaco, Laurent Dubois, Anouk Berger, Bengt Kayser and Daniel Sherly. Thanks to Paraskevi Synteta for her comments on this text.

7. *References*

Aronson, E., Blaney, N., Sikes, J., Stephan, G., & Snapp, M. (1978). *The Jigsaw Classroom*. Beverly Hills, CA: Sage Publication.

Baker, M. & Lund, K. (1997) Promoting reflective interactions in a computer-supported collaborative learning environment. *Journal of Computer Assisted Learning*, 13, 175-193.

---

[6] The TECFA members of the European SEED project: Daniel Schneider, Paraskevi Synteta, Catherine Frete et Stefane Morand.

Baker, M., Hansen, T., Joiner, R. & Traum, D. (1999) The role of grounding in collaborative learning tasks. In P. Dillenbourg (Ed) *Collaborative learning: Cognitive and Computational Approaches* (pp. 31-63) Oxford: Pergamon.

Barros, B. & Verdejo, F. (2000) Analysing student interaction processes in order to improve collaboration: The DEGREE approach. *Journal of Artificial Intelligence in Education*, 11, 211-241.

Barrows, H. S. & Tamblyn, R. N. (1980) *Problem-based learning: An approach ro medical education*. New York: Springer.

Berger, A., Moretti, R., Chastonay, P., Dillenbourg, P., Bchir, A., Baddoura, R., Bengondo, C., Scherly, D., Ndumbe, P., Farah, P. & Kayser, B. (2001) Teaching community health by exploiting international socio-cultural and economical differences. In P.Dillenbourg, A. Eurelings & K. Hakkarainen. *Proceedings of the first European Conference on Computer Supported Collaborative Learning (pp. 97-105)*, Maastricht, March 2001.

Burton, J. M. (1998) *Computer modelling of dialogue roles in collaborative learning activities*. Unpublished PhD thesis. Computer Based Learning Unit. University of Leeds, UK.

Clark, H.H. & Brennan S.E. (1991) Grounding in Communication. In L. Resnick, J. Levine & S. Teasley (Eds.), *Perspectives on Socially Shared Cognition* (127-149). Hyattsville, MD: American Psychological Association.

Constantino-González, M. & Suthers D. (2992) Coaching collaboration in computer-mediated learning. In G. Stahl (Ed.), *Computer support for collaborative learning: foundations for a CSCL community* (pp. 583-584). Mahwah, NJ: Lawrence Erlbaum Associates

Dillenbourg P. (1999) What do you mean by collaborative leraning?. In P. Dillenbourg (Ed) *Collaborative-learning: Cognitive and Computational Approaches* (pp.1-19). Oxford: Elsevier.

Dillenbourg, P., Baker, M., Blaye, A. & O'Malley, C. (1995) The evolution of research on collaborative learning. In E. Spada & P. Reiman (Eds) *Learning in Humans and Machine: Towards an interdisciplinary learning science*. (Pp. 189-211) Oxford: Elsevier.

Dillenbourg, P., Ott, D., Wehrle, T., Bourquin, Y., Jermann, P., Corti, D. & Salo, P. (2002). The socio-cognitive functions of community mirrors. In F. Flückiger, C. Jutz, P. Schulz and L. Cantoni (Eds). *Proceedings of the 4th International Conference on New Educational Environments*. Lugano, May 8-11, 2002.

Engeli M. (2001)(ed) Bits and Spaces, Architecture and Computing for Physical, Digital, Hybrid Realms, 33 Projects by Architecture & CAAD,

ETHZ: Basel: Birkhäuser Publishers

Fantuzzo, J. W., Riggio, R. E., Connelly, S., & Dimeff, L. A. (1989). Effects of reciprocal peer tutoring on academic achievement and psychological adjustment: A component analysis. *Journal of Educational Psychology*. 81(2), 173-177.

Glachan, M.D., & Light, P.H. (1981) Peer interaction and teaching: can two wrongs make a right? In G. Butterworth & P.H. Light (Eds.) Social cognition: studies of the development of understanding. Brighton: Harvester Press.

Hermann, F., Rummel, N. & Spada, H. (2991) Solving the case together: The challenge of net-based interdisciplinary collaboration. In P.Dillenbourg, A. Eurelings & K. Hakkarainen. *Proceedings of the first European Conference on Computer Supported Collaborative Learning (pp. 293-300)*, Maastricht, March 2001.

Hoppe, U. H. & Ploetzner, R. (1999) Can analytic models support learning in groups. In P. Dillenbourg (Ed.) *Collaborative-learning: Cognitive and Computational Approaches* (pp.147-168). Oxford: Elsevier.

Inaba, A. & Okamoto, T (1996) Development of the intelligent discussion support system for collaborative learning. *Proceedings of Ed-Telecom* '96. (pp 494-503), Bostoo.

Jermann, P. & Dillenbourg, P. (1999) An analysis of learner arguments in a collective learning environment. *Proceedings of the third CSCL Conference*, pp. 265-273, Stanford, Dec. 1999.

Jermann P, & Dillenbourg, P. (to appear). Elaborating new arguments through a cscl scenario. In. G. Andriessen, M. Baker & D. Suthers. (Eds). *Arguing to Learn: Confronting Cognitions in Computer-Supported Collaborative Learning environments*. CSCL Series. Amsterdam: Kluwer.

Jermann, P. & Schneider, D. (1997) "Semi-structured interface in collaborative problem-solving". First swiss worshop on distributed and parallel systems, Lausanne. http://tecfa.unige.ch/tecfa/publicat/jermann-papers/lsnne97/lsne-97-1.html.

Jermann, P. (2002) Task and Interaction Regulation in Controlling a Traffic Simulation. In G. Stahl (Ed.) Computer Support for Collaborative Learning. Proceedings of CSCL2002, Boulder, Colorado. (pp. 301-302), Hillsdale, NJ: Lawrence Erlbaum.

O'Donnell, A. M., & Dansereau, D. F. (1992). Scripted cooperation in student dyads: A method for analyzing and enhancing academic learning and performance. In R. Hertz-Lazarowitz and N. Miller (Eds.), *Interaction in cooperative groups: The theoretical anatomy of group learning* (pp. 120-141). London: Cambridge University Press.

Palincsar A.S. and Brown A.L. (1984) Reciprocal Teaching of Comprehension-Fostering and Comprehension-Monitoring Activities. *Cognition and Instruction,* vol.1, nº2, pp. 117-175.

Pilkington, R., Hartley, J.R, Hintze, D. & Moore, D.J., (1992) 'Learning to Argue and Arguing to Learn: An Interface for Computer-Based Dialogue Games', *Journal of Artificial Intelligence in Education*, 1992, 3(3), pp. 275-295.

Reiserer, M., Ertl, B., & Mandl, H. (2002) Fostering Collaborative Knowledge Construction in Desktop Videconferencing. Effects of Content Schemes and Cooperation Scripts in Peer-Teaching Settings. In G. Stahl (Ed.), *Computer support for collaborative learning: foundations for a CSCL community* (pp. 379-388). Mahwah, NJ: Lawrence Erlbaum Associates.

Salomon, G. & Globerson, T. (1989) When teams do not function the way they ought to. *International journal of Educational research*, 13 (1), 89-100.

Schwartz, D.L. (1995). The emergence of abstract dyad representations in dyad problem solving. *The Journal of the Learning Sciences*, 4 (3), pp. 321-354.

Soller, A., Goodman, B., Linton, F., and Gaimari, R. (1998) Promoting Effective Peer Interaction in an Intelligent Collaborative Learning Environment. *Proceedings of the Fourth International Conference on Intelligent Tutoring Systems* (ITS 98), San Antonio, TX, 186-195. Berlin:Springer-Verlag

Suthers, D., Weiner, A. Connelly J. & Paolucci, M. (1995) Belvedere: Engaging students in critical discussion of science and public policy issues. In. J. Greer (Ed). *Proceedings of the International Conference in Artificial Intelligence in Education*, Washington, August 16-19, pp. 266-273.

Veerman, A.L. & Treasure-Jones, T. (1999) Software for problem solving through collaborative argumentation. In P. Poirier and J. Andriessen (Eds) *Foundations of argumentative test processing* (pp. 203-230). Amsterdam: Amsterdam University Press.

Weinberger, A., Fischer, F. & Mandl, H. (2002) Fostering computer supported collaborative learning with cooperation scripts and scaffolds. In G. Stahl (Ed.), *Computer support for collaborative learning: foundations for a CSCL community* (pp. 573-574). Mahwah, NJ: Lawrence Erlbaum Associates.

Zumbach, J., Mühlenbrock, M., Jansen, M., Reimann, P. & Hoppe, H.U. (2002) Multi-dimensional tracking in virtual learning teams: An exploratory study. In G. Stahl (Ed.), *Computer support for collaborative learning: foundations for a CSCL community* (pp. 650-

651). Mahwah, NJ: Lawrence Erlbaum Associates.