



D23.5.1 (Final)

Framework for integrated learning

Main author : Pierre Dillenbourg (EPFL)

Nature of the deliverable : Report

Dissemination level : Restricted

Planned delivery date : December 2004

**No part of this document may be distributed outside the consortium / EC without
written permission from the project co-ordinator**

*Prepared for the European Commission, DG INFSO, under contract N°. IST 507838
as a deliverable from WP23
Submitted on 21-12-2004*

Summary

In this paper we outline the instructional approach of scripting CSCL. Scripts are didactic scenarios that structure collaborative learning activities in a number of phases. The scripts may define for each phase what task the students have to perform, the composition of the group, the way the task is distributed, the mode of interaction, and the timing of the phase. CSCL scripts can be further differentiated with regard to the design dimensions granularity, degree of coercion, locus of control, and degree of generality. Finally, scripts may structure different planes of a collaborative learning environment, such as the individual plane, the plane of small groups, or the plane of classes, schools etc. Against the background of these multiple dimensions, all kinds of CSCL scripts can be classified. Different major classes of scripts could be identified. These are, for instance, the Jigsaw class, providing learners with complementary information, the conflict class that aims to trigger socio-cognitive conflict in CSCL groups, and the reciprocal class that provides learners with roles to regulate the learning partners, which they are supposed to switch after a specified time. Future work focuses on the translation of different script components in a modelling language in order to systematize research on different script components, making scripts transferable from one CSCL environment to another, and to better understand what kind of collaborative learning activities work for what kind of tasks.

History

Filename	Status	Release	Changes	Uploaded
D23-05-01-F.pdf	Final	1	--	21/12/2004

1	Introduction	2
2	What is a script?	2
2.1	Cultural versus didactic scripts.	2
2.2	Ideal, mental and actual scripts.	3
2.3	Why scripting collaborative learning?	4
2.4	How do computers support scripts?	5
2.5	Why mobile?.....	6
2.6	"Integrated learning" scripts.....	7
3	Design Space	8
3.1	Design dimensions	8
3.2	Script schemata	9
3.3	Scripts commonalities	10
3.4	Core script versus didactic envelope.....	10
4	Inside core scripts	11
4.1	The Task Structure	11
4.2	The Time Structure	12
4.3	The Social Structure.....	13
4.4	The space structure	15
4.5	Integration operators	16
5	The Swish model	16
6	Conclusion	18
7	References	19

1 Introduction

The MOSIL project concerns scripts for computer-supported collaborative learning activities. Scripts are didactic scenarios that structure collaborative learning activities, specifying roles, subtasks, deadlines, ... Our research community has developed a certain number of scripts. The goal of this deliverable is propose a framework for the design and understanding of CSCL scripts. The intuitive isomorphism between the numerous existing scripts constitutes a great appeal for scientists to produce a design model. Beyond the sake of modeling, this language could be used to foster exchanges among teachers or designers and even to build tools for authoring CSCL scripts.

As a first step in that direction, this deliverable describes scripts under a specific angle that was discussed within our Network of Excellence. It is not presented as a cognitive model of collaborative learning but as a conceptual framework, i.e. as a way to envision scripts.

2 What is a script?

Before to read this general description of scripts, it is recommended to read the deliverable 23.3.1 that presents a variety of concrete scripts examples.

2.1 *Cultural versus didactic scripts.*

Many of our daily activities do implicitly or explicitly follow scripts. As Schank and Abelson pointed out (1977), our actions and interactions in a restaurant followed a generic pattern, more or less sequential such as: greetings, being seated, ordering food, ... Such an implicit script has been culturally acquired through many years of social life. Similar scripts do also drive students' behaviors in classrooms or in teamwork. When teachers engage students in collaborative activities, they usually provide them with pretty undefined instructions such as "do this task by group of 3". The students import acquired scripts that convey implicit expectations with respect to collaboration, for instance the fact an even group participation is usually desirable. These expectations are mainly inferred by students when the teacher announces how (s)he'll grade the group productions.

The MOSIL research group was instead concerned by a different type of scripts which are usually not part of students' habits but are made explicit by the teacher. Scripting collaboration is describing the way students have to collaborate: task distribution or roles,

turn taking rules, work phases, deliverables, etc. A collaboration scenario or script (O'Donnell & Dansereau, 1992) is an explicit didactic contract (Brousseau, 1998) between the teacher and the students regarding to their mode of collaboration.

2.2 *Ideal, mental and actual scripts.*

The way student do actually collaborate is of course be different from the way the teacher asked them to collaborate. The learning activities results from the interplay between three types of script:

- The *ideal* script is the set of behaviors that the teacher or the environment prescribes;
- The *mental* script is the mental representation that the group builds from the teacher prescription. The mental script includes both how each group member understands his or her role in collaboration but also, and more importantly, how the group builds a shared representation of the collaborative process.
- The *actual* script refers to the task and group interactions that students do actually engage.

The distance between the ideal script and the actual script depends on several script features:

- *Intelligibility*. We encountered problems in projects (Berger et al, 2001) where the prescribed script was too complex. Despite the fact that we provided teams with graphical representation of the script and offered a close follow-up by teaching assistants, the students – and even some tutors- did not manage to construct a clear mental script.
- Degree of *coercion* (Dillenbourg, 2002) of the didactic script. The script may be simply conveyed through initial instructions provided either by the teacher or by the learning environment. In other CSCL environments, the script is regularly enforced by prompts or other design features (e.g. each student is represented by an iconic description of his role). The degree of coercion is even higher is CSCL environments where the next authorized student's action is decided by the system, according to the rules of the script. High coercion scripts reduce the gap between ideal and actual scripts, but raise the risk of loosing the natural strength of collaborative learning (Dillenbourg (2002)).
- *Fitness*. Many scripts specify a distribution of roles among group members. For instance, one group member is asked to be leader or coordinator while another

one is in charge of taking notes. The actual script depends on the good match between the role requirements and the group member profiles. As reported in various jokes, if the French member is in charge of cooking and the German one in charge of organization, the match is fine, but, if it is the other way around, the actual script might be catastrophic. Low fitness leads either to low roles adoption or short roles adherence (the team does not stick to the prescribed roles very long).

Fisher's notion of locus of control (see section 3) is related to the difference between actual and mental scripts. When the script is just a didactic method, its mental representation is only instrumental to the actual script: play it and forget it! Conversely, when the goal is that students internalized the script, the actual script is instrumental to the mental script: play it in order to learn it! This is for instance the case of the reciprocal tutoring script (Palincsar & Brown, 1984).

2.3 *Why scripting collaborative learning?*

Scripts are the convergence point between the instructional engineering approach, which dominated learning technologies for two decades, and the socio-constructivist stream. They result from the effort to engineer collaborative learning. Scripting is some compromise between the constraints usually induced by instructional design and the freedom of collaborative learning. Constraining collaborative was suggested by empirical studies on the effectiveness of collaborative learning. These studies show that this effectiveness 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 (Dillenbourg, Baker, Blaye & O'Malley, 1995). Therefore, 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, ... In short, scripts are tools for enhancing the probability that productive interactions occur in the group. The key issue in the design of a CSCL script is: *which type of peer interactions does the teacher want to foster to reach educational objectives?*

Building a script is a specific instance of instructional design. It could be argued that the design space of script is the same as the instructional design space and that scripts are just a trendy word to refer to lesson plans. We defend the use of 'scripts' as different from any instructional sequences: *CSCL scripts are instructional sequences in which peer interactions are expected to constitute the core learning mechanism. Therefore, the*

current framework proposes a model for the design of the core mechanisms of group interactions. Of course, scripts trigger various other learning mechanisms besides collaborative interactions, as explained in the next section.

Witnessing the growing influence of instructional design over socio-cognitive aspects of collaborative learning, Dillenbourg (2002) emphasized the risk of over-scripting collaboration that is producing scripts that constraint natural collaboration in a way that make it sterile, inhibiting the targeted peer interactions or missing the above mentioned core collaborative mechanisms.

2.4 *How do computers support scripts?*

Didactic scripts are used for enhancing collaborative learning with or without computers. The MOSIL project investigated scripts in computerized environments and raised the issue of the added-value that technology brings to the use of scripts. This added-value takes several forms:

- *Connecting.* When scripts include remote activities, technology is simply the communication tool.
- *Managing:* Computerized scripts off-load groups and teachers from some logistics duties such as time management (reminding deadlines, new phases,...) and information flows (for instance, distributing data sets to different group members or aggregating individual data within groups).
- *Reifying:* Computerized scripts may provide students with a concrete script representation (phases, roles, ...) that is dynamically updated with time information but also activity information (for instance, representing group interactions).
- *Constraining:* Computerized scripts offer opportunities for shaping communication within groups, using semi-structured communication interfaces and/or dialogue grammars.
- *Enabling:* Computerized scripts enable scripts events that would be harder or impossible to create without computers, such as finding peers with most opposite opinions or pairs who obtain identical problem solutions.
- *Traceability.* To the extent to which computerized scripts record interactions and outputs, they offer both functionalities for the teacher to analyze and regulate teamwork and for the student themselves to access to previous steps.

Among the negative aspects of computerized scripts, we find of course the many drawbacks of computer-mediated communication versus face-to-face communication. However, these drawbacks, that appear in any CSCL environments, are compensated by the integration of computerized activities and computer-less activities such as face-to-face collaboration or class discussion (see next section).

Another weakness of CSCL scripts is the *loss of flexibility*: good teachers adapt their plan on the flight, based on their observation of group processes, and this adaptation is often more difficult when the script is embedded into a computerized environment. This rigidity is however not intrinsic to any software. They rather result from the fact that CSCL designers so far under-estimated the teachers' needs to modify scripts on the fly.

2.5 Why mobile?

Scripts software can be run on a variety of hardware, including mobile devices. Some of the scripts described in our deliverable 23.3.1 rely on mobile technologies. The added value of mobility takes several forms:

- *Integration*: Mobile devices enable to integrate within a single script the activities that occur across multiple spaces, for instance collecting data into field trip or experimental lab and analyzing them in the classroom.
- *Context enrichment*. Mobile devices enrich collaboration software with context-awareness features such as location information: selecting peers who are close to each other or who have conflicting viewpoints on some building, ...
- *Identification*. A mobile device lies in a pocket. In many scripts, it is useful to identify information subsets to persons and vice-versa. Showing my PDA display to my partner becomes an explicit act of sharing information.

These functions concern the relevance of mobile technologies for CSCL scripts. They do not discard other interesting features of mobile devices that apply as well to individual learning:

- *Nomadic students*. Obviously, mobile devices offer the advantage of running script with learning groups that are intrinsically mobile (e.g. salesmen)
- *Multimodality*. We envision the role of mobile devices as Swiss army knives for educators. At the opposite of desktop computers, they will progressively be enriched with a variety of sensors. The build-in camera for instance enable to capture gestures in the same way an optical mouse do it. Thermometers,

barometers, accelerometers and other information capturing tools will enrich phones or PDAs as they enriched watches.

- *Recorder.* Always in one's pocket, mobile devices enable individual or groups to record information all day long: positions, pictures, audio records, notes, ...

Our framework takes some distance with the 'learn anytime anywhere' approach which favors access to information while we see mobile devices as tools for enhancing group interactions. Mobile technologies are mainly conceptualized as ways to deepen the integration of multi-location activities and to augment group processes with features that desktop computing does not offer. It must be however acknowledged that the role of spaces and mobility is not fully developed in the current framework.

2.6 "Integrated learning" scripts

There is no reason why collaborative learning should be treated as an exclusive pedagogical approach. Instead, group activities gain from being integrated with other classrooms activities. Many scripts illustrated in deliverable 3 include individual work (e.g. reading a paper, writing a synthesis,...) and/or class-wide activities (introductory lectures, debriefing, ...). The power of scripts is to integrate these diverse activities within one consistent whole. We refer to this as a *pedagogical integration*: An integrated learning script hence goes beyond collaborative learning *stricto sensu*. Very importantly, many scripts maintain the teacher in his leading role. He is not properly teaching but is very active and salient as the "chef d'orchestre" of the whole script: he does intervene much on interactions inside group but manages the whole sequence of activities, As long as this role is not too much restricted by the lack of software flexibility, preserving the teachers importance constitutes a positive factor of acceptance among teachers.

There is no reason either to restrict CSCL scripts to distance education. Integrated learning differs from the so-called 'blended learning', which is often the mere juxtaposition of face-to-face and computer-mediated activities. Integrated learning scripts integrate into a consistent whole some activities which are on-line or not, in front of a computer or not. They may occur in a variety of places (classroom, lab, field trip, home, work ...). The rapid transition between activities with or without computers is facilitated by lighter/mobile hardware. We refer to this as *physical integration*.

Finally, high end scripts reach what we refer to as *functional integration*: they support data flow between multiple activities, e.g. collecting individual solutions for supporting group argumentation, distributing partial information sets to the different roles within groups.

Our focus "integrated learning" stresses the fact that CSCL scripts convey a conceptual move away from e-learning discourse: the technology does not define the course. The course is constructed around a script and it occurs that some of the script activities rely on technologies.

3 Design Space

The proposed framework conceptualizes scripts by defining the scripts design space.

3.1 Design dimensions

As deliverable 23.3.1 illustrates, there exist a large diversity of scripts, conducted with or without computers. The design space for scripts is structured around at least four axes:

- *Granularity*. Scripts vary along the time scale (typically from 20 minutes to one semester) and on the grain size of sub-tasks definition. A coarse grain script such as project-based teaching, each phase includes a significant task that may last a few weeks, such as "customer needs analysis". At the other end, finest grain scripts may reach the utterance level, i.e. specify the authorized dialogue moves at the next utterance.
- *Degree of coercion*: Scripts vary according to the freedom loss they generate, i.e. to which extent the actions of the students are constrained by the script. Some scripts force the students to achieve specific subtasks while other simply inducing it. High granularity scripts then to be more coercitive. Some scripts focus on setting up initial conditions while other, more coercitive, constraint the collaborative process per se during the whole session.
- *Locus of control*. Some scripts are used as a didactic contract or game to be played by the learner while others aim to be internalized by the students. For instance, the Arguegraph (Jermann & Dillenbourg, 2003) is just an activity that students may forget at the end, while the reciprocal teaching script used by Palincsar and Brown (+984) gains its efficiency by being then turned into self-monitoring skills for text understanding. Kollar, Fischer & Hesse (to appear) refers to former as having an external locus of control and to the latter as an internal one.
- *Degree of generalisability*. As any pedagogical method, scripts raise questions of generalisability. It is clear that scripts such as the ArgueGraph are only relevant in domains where key notions can be argued about. This generalisability is not bound by classical scientific boundaries (e.g. this script would be good for maths but not

for social sciences) but instead by the relevance of the interactions the script intends to trigger. A deeper analysis of generalisability leads us to the notion of design patterns

3.2 Script schemata

All scripts are different from each other but there are recurrent patterns. We do not use the terms 'design pattern' here since it is used in software engineering with a rather operational meaning. Instead, we refer to more abstract commonalities among scripts. Many of them are variations around the JIGSAW model, i.e. they share the general schema of distributing the necessary information among team members. A script schema is an abstract description of the script structure; it expressed what is common to various scripts that belong to the same class. Deliverable 23.3.1 lists several scripts schemata illustrated by different families of scripts. Examples of schemata are:

- The Jigsaw schema build upon the partition of the knowledge or information necessary to solve the task, either by forming pairs who have complementary knowledge (Hoppe & Ploetzner, 1999) or by providing them with complementary information or by asking them to play complementary roles. The design principle is that no learner has the necessary information/knowledge to solve the problem aloe. It can only be achieve through intense interaction with the other team members.
- The conflict schema triggers argumentation among group members by forming pairs with student with conflicting opinions (e.g. ArgueGraph), by providing them with conflicting evidence or by asking them to play conflicting roles.
- The reciprocal schema defines two roles in teams, one of the peers regulating the other and then switching roles. Examples are the reciprocal teaching approach (Palincsar & Brown, 1984),

This list and the longer list in deliverable 23.3.1 are of course not exhaustive. There exist many schemata and even more of them are still to be invented. Other methods for group-based learning could also be described as script schema: project-based learning, problem-based learning, inquiry based-learning,... They specify team work process and integrate it with other activities, individual learning and debriefing with a tutor.

We stress the fact that these schemata are *not recipes* for collaborative learning. They provide a general structure but the art of design is to apply this structure to the specificity of the domain to be learned and the peculiarities of the target audience.

3.3 *Scripts commonalities*

If we move one level more in the abstraction level, scripts can be define as variations of a generic template with a limited set of attributes. Basically, a script is a sequence of phases; each phase was defined by five attributes (Dillenbourg, 2002):

- the task that students have to perform at this phase,
- the composition of the group: number of subjects, group formation rules, ...
- the way that the task is distributed within and among groups (subtasks, roles, ...),
- the mode of interaction (face-to-face, a/synchronous, text-based or voice-based, ...)
- the timing of the phase.

This simple description scheme could be translated within the new educational modeling languages (EML) that proposed well-structured terminology for describing instructional sequences. These languages do constitute a step forward compared to the content-centric approach of the educational metadata initiatives (SCORM and others). However, this simple scheme and the EML languages do not constitute a design model: they provide an external description of the scripts but fail to capture the core idea of a script. A script model should encompass the core mechanisms by which this activity sequence has been constructed, or, in other words, why this script is expected to generate learning. It may the case that IMS Learning Design could be expanded, but currently it is more at the descriptive level than at the modeling level. For instance, groups are not defined explicitly but indirectly by assigning roles. This prevents for instance building a Jigsaw script where team members have different roles within each team, or a Reciprocal Tutoring script where roles rotate among group members at each script phases. The social structure of a script should be explicitly modeled.

The long term goal of our research is to model script not at the descriptive level (as a sequence of activities) but at the level of core or generative mechanisms. The quest is to translate these core mechanisms into a design language and later on into software components. This framework constitutes a fist step in that direction. A second step has been carried out during this year and is reported in our deliverable 23.2.1, where several schemes for describing CSCL scripts are compared and analyzed.

3.4 *Core script versus didactic envelope*

In order to capture the core idea, we discriminate the *core script* from its *didactic envelope*, i.e. a set of pre- and post-structuring activities:

- Pre-structuring activities provide the conditions necessary to make the core script activities "working well": introductory lectures, readings, exercises to wake-up pre-requisite skills, advanced organizers (e.g. metaphors), ... In addition to the cognitive aspects of pre-structuring activities, they are sometimes necessary to enable students to play their role in the script.
- Post-structuring activities includes debriefing activities such as the comparison of multiple solutions, synthesis lectures or readings, summary writing, ... These are mostly reflective activities, aimed at turning group experience into knowledge.

This didactic envelope is every important for the effectiveness for the script. It corresponds to broadening we explained in section 3.6 from collaborative learning *stricto sensu* towards integrated learning. However, this envelope dissolves the readability of scripts or their identity. Since these extra activities are standard classroom activities, some collaborative scripts may look like a genuine lesson plan. As we explained in section 3.3, the difference between a collaborative script and a genuine lesson plan is that these additional activities are plugged around a core distributed task. The SWISH model below concerns the structure of this distributed task.

4 Inside core scripts

Our framework borrows the distributed cognition models, according to which a group of actors and their tool can be understood as a single cognitive system. The components of the system are the students who participate into the scripted teamwork as well as the tools and resources available to them. Actually, the script itself can be considered as a tool that contributes to the functioning of the distributed system. In our framework, *the core script is what defines the organization of a distributed cognitive system*. This organization is defined by 3 dimensions: the task structure, the time structure and the social structure.

4.1 The Task Structure

What is a distributed system? Our complex world is an infinite regress of indented distributed systems: the individual, the group, the class, the school, the family, the friends, the local community, the society... Some tenants of the distributed cognition approach emphasize the larger scale (e.g. Lave, 1991) , namely how the culture of our social surrounding shapes cognition. We are closer to authors like Huchthins (1995) or Salomon (1993) for which a distributed system is the set of agents and tools involved in the accomplishment of the task. Our framework is based on three axioms:

- 1) *The task defines the system.* The foundation of a script is the definition of the task that the group of student has to achieve collaboratively. This task hence defines the scale of the distributed system. We refer to task-defined distributed system as the reference level.
- 2) *Task distribution.* The core script defines the way the task can be distributed, i.e. which team members will perform which sub-tasks. We refer to sub-tasks in a generic way: they can be independent from each other, like in cooperative work, or tightly coupled, like when one peer has to regulate the other. Many scripts produce task distribution by defining roles that induce a somewhat 'natural' distribution of work into sub-tasks. Task distribution can be formalized a matrix with subtasks on one dimensions and actors on the other.
- 3) *Plasticity & rotation.* A distributed cognitive system is more than a collection for interrelated components. In our previous research (Dillenbourg & Traum; to appear), the strength of this concept was revealed by our observation of the **plasticity**, i.e. the ability to re-allocate dynamically sub-tasks to different sub-components during the task completion process. When a pair uses several communication tools (a chat, a whiteboard, a shared notebook,...), any technical problems with one tool lead peers to immediately re-distribute communication functions (store findings, negotiate decisions, ..) over the other tools. Many scripts aim to create this plasticity by modifying the task distribution at different phases. We refer to them as *rotating scripts*. Obviously, rotation ensures that each team members practices each sub-competence of the main target competence. It can be argued that rotation is a condition for the internalization of scripts, but we do not know any empirical finding supporting this intuition.

It is interesting to notice that the design of script shares with other instructional design approaches the need to carry out a deep task analysis and/or task decomposition and that can hence borrow the task analysis techniques that have been developed in instructional engineering. An alternative approach used in project-based learning is to ask the team itself to define the task, the task distribution and sometimes the rotation mechanisms.

4.2 *The Time Structure*

As we said earlier, most scripts defined a sequence of phases and in many cases these phases are limited in time. At the opposite of the task structure, a time frame is not necessary conditions for a distributed system. Distributed system may function effectively without any formalized time frame, in an opportunistic way, while others may benefit

from a time structure as a tool for coordination. The rationale for setting a more rigid time frame up is threefold:

- The time structure makes the task distribution more *salient*, especially since deadlines define clear boundaries between consecutive subtasks.
- The time structure makes life easier: *time management* is a critical factor in everyday educational practices, for both the teachers and the learners. It is even a more important issue when part activities take place over the web, i.e. outside the genuine time structure that exists in schools.
- The time structure facilitates *tutoring* by providing the coach with an easy way to follow the team progress.

4.3 The Social Structure

Integrated learning activities occur at different social planes. Vygostky discriminated three planes: the intra-psychological plane, the inter-psychological plane and the social plane. The intra-psychological plane is individual. The difference between the inter-psychological and the social plane is not clear-cut. It is of course a matter of group size, but we prefer to define it cognitively. Group activities can be located at inter-psychological plane as long as team members maintain some representation of their teammate's cognition. The social plane is the level where individual representations disappear behind the culture that the community members jointly constructed. More practically, we discriminated collaborative versus collective activities (Jermann et al, 1999), the former concerning small groups (2-7) while the latter occurred when the teacher gathers all the class students (whether they are 20 or 300).

The scale of social activities is of course a continuum from 1 to the entire planet. However, if we consider the range of activities in CSCL environments, we often encounter the following five levels of activity:

1. Individual plane: solo activities
2. Group plane: activities in small groups ranging from 2 to, let's say, 8 people. This is often the target or reference level of CSCL scripts
3. Class Plane: activities involving all students enrolled in the same course
4. Community Plane: activities that involve identified actors such as other classes, expert groups, families; for instance a class from school X designed a mathematical challenge that is then proposed to all classes in the community.

5. World Plane: activities that are available to unidentified actors, for instance when a class journal is produced on the web, the entire world may read it. If a survey is conducted via the web, any works user may vote.

These levels match many activities we are familiar with, but are of course arbitrarily defined. What matters is not the exact definition of the levels but the fact that integrated scripts define moves across multiple planes. Therefore, we present scripts as illustrated in figure 1, where social planes are represented vertically and time horizontally. Figure 1 illustrates the social structure of the script "Concept Grid" (Dillenbourg 2002), in which the two upper planes are not exploited. The concept grid script is described in deliverable 23.3.1 but here is a short description to understand the social structure;

1. Groups of four students have to distribute four roles among themselves. Roles correspond to theoretical approaches and are defined by a notorious defender of this approach. 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. They cover the key notions that teacher expects learners to acquire. The group distributes the concept definition work among its members.
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 and to define in 3-5 lines the relationship between each grid neighbor. They often have to try many organisations of the concepts on the grid.

The rationale of this script is that, during phase 4, each student has to explain the concepts to each other. If Paul has defined concept X and Lucil as defined concept Y, the only way for them to write together the difference between X and Y is that Paul explains Y to Lucil and Lucil explains Y to Paul. A simpler explanation would not be productive; the explanation has to be pushed up to the point where they can write down the difference or similarity between the concepts.

This script involves multiple movements between social planes. As explained above, the reference plane is the social plane where the task structure is defined (core script). Other planes are numbered relatively from the reference plane (-2, -1, +1, +2, ..). The didactic envelope often includes activities at these other planes. Actually, activities always occur at multiple planes: individual cognition does not freeze during class interactions and culture does not stop shaping our ideas during individual work.. However, designing a CSCL script requires to select a focus plane for each phase.

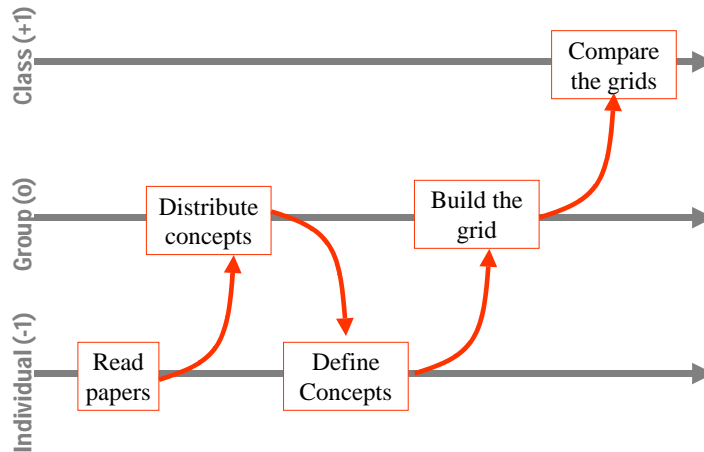


Figure 1: Social structure of the 'ConceptGrid' script.

4.4 The space structure

As explained in section 2.5, an integrated script may include activities that occur across multiple spaces. Mobile technologies enable designers to *erase space*, i.e. to run a script across various spaces as if it would occur within a unique space. In this case, the spatial structure does not play a role in the definition of the core script. In this case, the added value of technology relates to the functions listed in section 2.5

The more interesting avenue is to use mobile technologies to *emphasize space*, to take full advantage from it, to explore how spatial information enriches the script mechanisms. To explore this direction, we specify the relationship between space and other dimensions of collaborative process:

- *The relationship between individual position and information:* being in a room give access to certain resources; having different positions may induce different perceptions of a phenomena,...
- *The relationship between physical space and problem space:* if teammates perceive the mapping between the problem space (conceptual) and the physical space, the may infer the activities of their teammates from their position.
- *The relationship between trajectory and strategy.* If there is a mapping between problem space and the physical space, the spatial path followed by an individual is also a solution path. It informs partners about his problem solving strategy and serves as a basis for inferring intentions or strategy.

Script designers may give more or less importance to these relations by tuning different features of locative media:

- Degree of *context awareness*: how much information may the system provide by knowing its position? This refers to the precision of location but also to the underlying *geographical information system* that enables to match spatial position with any database from which useful information can be retrieved: rooms, objects around, people around, city name, vegetation, weather information...)
- Degree of *mutual awareness*: does the environment inform each member about where are his teammates? Spatial awareness support mutual modeling mechanisms, i.e. the mechanisms by which partners build a representation of each other.

4.5 Integration operators

When we talked about 'functional integration' (Section 2.6), we refer to the dataflow between activities at different planes. The output of an activity at level N is later on reused at level M, where, in many cases, M is different from N. Our research work is based on the hypothesis that the dataflow in many scripts can be described by a *small number of operators* for moving up and down the different planes of the social space, whatever the level is.

- Upward operators are 'aggregate', 'list', 'differentiate', ... They collect output from plane N and turn them into input for plane N+i. The type of processing depends on the nature of data. Data form a matrix with social items (people, groups, ...) in columns and the behavioral variables in rows. The aggregation operator collapse columns e.g. summarize all individual values for each variable. The differentiation operator collapse rows. i.e. summarize all variable for each individuals, emphasizing their differences. When data are too complex to be combined (e.g. all groups produced engine models), they can simply be listed.
- Downward operators distribute an object O attached to level-N , i.e. assign sub pieces of O to members of level-N. For instance, at the group-plane each group of 4 students is associated with 4 roles and 12 readings and the operators allocate roles and readings to each team members (plane N-1).

5 The Swish model

So far, we proposed a static description of the structure of scripted CSCL. In order to account for learning, our model must grasp the dynamics of scripts. Collaborative learning is often defined as the process of constructing and maintaining a shared understanding of the task (Roschelle & Teasley, 1995). As we talked several times about

distributed task, it may sound counter-intuitive to distribute the task among different learners, since this opens the door to misalignment of views, understandings and goals. To the same extent it is counter-intuitive to develop scripts that foster conflict among peers and are hence detrimental to the construction of a joint solution. To bypass this counter-intuition, we need to rely on Schwartz' (1995) definition of collaborative learning as the *effort* necessary to build a shared understanding. Learning is the side effect of the cognitive processes triggered by the interactions (explanation, argumentation, mutual regulation, ...) engaged to develop this shared understanding.

On this basis, our design model - "**Split Where Interaction Should Happen**" – can be summarized in 3 points:

- Learning results from the *interactions engaged by students to build a shared understanding of task **despite** the fact that it is distributed.*
- Hence, the task distribution determines the nature of interactions: *Interactions are mechanisms for overcoming task splits.*
- Hence task splits can be, in some kind of *reverse engineering*, designed for triggering the interactions that the teacher wants to foster: Split Where Interaction Should Happen

The SWISH model can be related to the design rationale of the scripts schema listed in section 2.2.

- In a Jigsaw-type script, task split occurs by distributing the knowledge or information necessary to solve the task as in the GIRD script presented in section 4.3: since none of the group members has enough information to solve the task alone, he or she needs to share his knowledge or contribution to others.
- The conflict-type scripts aim to split the task into controversial subsets in such a way that a joint solution cannot be reached without intensive argumentation
- The reciprocal-type scripts horizontally divide the task split into cognitive and metacognitive processes. Since these processes need to be tightly coupled, the only way to achieve a shared solution is that both peers continuously engage into mutual regulation interactions

The SWISH mechanism can be described in a very abstract way. The process starts by introducing a perturbation in a distributed system. The system engages into repair mechanisms for reducing the perturbation. These repair mechanisms necessitate intensive interactions and these interactions trigger the target learning mechanisms. Learning is hence the result of over-compensation the drawback of task split.

Obviously, this model only holds if the group as both the ability and the will to compensate task splits. Solving conflicts, explaining complex concepts or regulating bad problem solves may be beyond the skills of individual. A more frequent issue is the lack motivation to reach a shared understanding. The SWISH model is only valid for tasks that required a high level of shared understanding. If students manage to solve the task without constructing a shared understanding, the repair effort will not be worth.

6 Conclusion

Different kinds of scripts may be characterized along the lines of multiple design dimensions. There is yet little knowledge on how these different kinds of scripts may influence processes and outcomes of CSCL. So far, researchers of different labs have designed learning environments for single studies that were specific to the experimental scenario and the experimental cooperation script. Unfortunately, these experimental studies produce singular results on idiosyncratic and inflexible learning environments, difficult to integrate into a unified perspective on CSCL scripts and to transfer to different contexts. Future CSCL script research needs to be systematized conceptually and technically. First of all, script components and their specific effects on processes and outcomes of CSCL need to be conceptualized. Existing scripts need to be described and allocated on the design dimensions. Empirical studies need to abstract the effects of the script characteristics over multiple domains and tasks. Based on these results CSCL approaches such as the SWISH model can be developed further. Second, a technical abstraction of script components should be based on the script conceptualization. Script components can be translated into modeling languages. Technically described script components may be implemented into different learning platforms. Instead of re-inventing and designing scripts for one specific learning environment, a number of modular script components may be combined to form various scripted learning environments that, for instance, arrange learning resources spatially and temporally for collaborative learners. Modeling languages may reduce the design work, but also support isolation and generalization of effects found with CSCL scripts, e.g., how script components affect processes and outcomes of CSCL in different domains. Thus, modeling languages help to decouple the conceptualization of collaborative learning environments from their realization. Furthermore, standards represented in a modeling language for collaborative learning processes may aid interdisciplinary research in unifying terminology on CSCL scripts.

In order to advance research and practice of CSCL scripts, scripts need to be conceptualized based on an empirically grounded framework of script design dimensions

and scripts need to be technically described with a modeling language that can translate the framework. These two tasks require an interdisciplinary research team with a distributed expertise on modeling languages, script design, and analysis of processes and outcomes of CSCL. This research team should aim to provide a theoretical framework of CSCL scripts, translate scripts into a modeling language, and empirically test the theoretical framework. This work should enhance the instructional approach of CSCL scripts, but also move collaborative learning approaches forward in better understanding the relation between specific (scripted) activities of learners in groups and their learning outcomes. Furthermore, research is required to advance methodologies to analyze processes and outcomes of CSCL. Future efforts are to be invested in semi-automatically analyses of the interaction of learners. If the interaction could be analyzed with little time lag, deficits of the collaborative learning process could be identified immediately and the respective technically described script components may be selected and applied that support learners with regard to their specific deficits during collaboration. Conversely, CSCL scripts may be faded out after learners have internalized the script suggestions or do not require this support anymore in order to engage in the specified interaction patterns. It is in this vein of dynamic application of scripts that CSCL may have an advantage over face-to-face learning without compromising the idea of self-guided collaborative learning.

7 References

- Brousseau, G. (1998). *Théorie des situations didactiques*. Grenoble, La Pensée Sauvage.
- Dillenbourg P. & Traum, D. (submitted) The complementarity of a whiteboard and a chat in building a shared solution. *Journal of Learning Sciences*.
- Dillenbourg, P. & Jermann, P. (2003). 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.
- 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.
- 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.
- Hutchins, E. (1995). How a cockpit remembers its speeds. *Cognitive Science*, 19, 265-288.

Jermann, P. , P. Dillenbourg, P. & JC Brouze. (1999) Dialectics for collective activities: an approach to virtual campus design. Proceedings of the 9th *International Conference on AI in Education*, Le Mans (France), July 99.

Kollar, I., Fischer, F. & Hesse, F. W. (to appear). Computer-supported cooperation scripts - a conceptual analysis. *Educational Psychology Review*.

Lave J. (1991) Situating learning in communities of practice. In L. Resnick, J. Levine & S. Teasley (Eds.), *Perspectives on Socially Shared Cognition* (63 - 84). Hyattsville, MD: American Psychological Association.

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, n°2, pp. 117-175.

Roschelle, J. & Teasley S.D. (1995) The construction of shared knowledge in collaborative problem solving. In C.E. O'Malley (Ed), *Computer-Supported Collaborative Learning*. (pp. 69-197). Berlin: Springer-Verlag

Salomon, G. (1993) No distribution without individual's cognition: a dynamic interactional view. In G. Salomon. (Ed). *Distributed cognitions. Psychological and educational considerations* (pp. 111-138) Cambridge, USA: Cambridge University Press.

Schank, R.C. & Abelson, R. (1977). *Scripts, Plans, Goals, and Understanding*. Hillsdale, NJ: Earlbaum Assoc. 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.

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.
