Chapter 16

DESIGNING INTEGRATIVE SCRIPTS

Pierre Dillenbourg and Patrick Jermann
Ecole Polytechnique Fédérale de Lausanne (EPFL)

Abstract: Scripts structure the collaborative learning process by constraining interactions, defining a sequence of activities and specifying individual roles. Scripts aim at increasing the probability that collaboration triggers knowledge generative interactions such as conflict resolution, explanation or mutual regulation. Integrative scripts are not bound to collaboration in small groups but include individual activities and class-wide activities. These pre- and post-structuring activities form the didactic envelope of the script. In many cases, the core part of the script is based on one among a few schemata: Jigsaw, conflict, reciprocal. We propose a model for designing this core component. This model postulates that learning results from the interactions that students engage in to build a shared understanding of a task *despite* the fact that it is distributed. Hence, the way the task is distributed among group members determines the interactions they will engage in. Interactions are viewed as the mechanisms for overcoming task splits. A large variety of scripts can be built from a small number of schemata, embedded within activities that occur across multiple social planes, activities which are integrated with each other by few generic operators.

1. INTRODUCTION

When teachers ask students to carry out collaborative activities, they usually provide them with global instructions such as "do this task in groups of three". These instructions are completed with implicit expectations with respect to the way students should work together, for instance an even group participation is often believed as desirable. A script describes the way students have to collaborate: task distribution or roles, turn taking rules, work phases, deliverables, etc. This contract may be conveyed through initial instructions or encompassed in the learning environment.

Scripts illustrate the convergence between instructional engineering and socio-constructivism. The need for engineering collaborative learning results from 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, etc.), the task features and the communication media. 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). What predicts learning outcomes is the richness of social interactions (conflict resolution, elaborated explanations, mutual regulation, ...). Scripts aim at enhancing the probability that these knowledge productive interactions occur during collaboration. Hence, the key design issue: which interactions need to be scaffolded in order to reach the educational objectives?

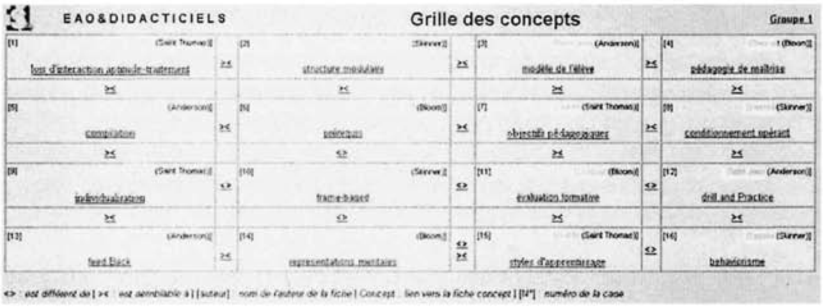
Most chapters in this volume address the notion of scripts in computer-supported collaborative learning (CSCL). Within the classification proposed by King (this volume), our approach clearly belongs to the pedagogical stream: our scripts are pedagogical artifacts designed by educators and explicitly imposed on learners. The striking similarity between the many existing scripts resembles an invitation to produce a design model. Beyond the sake of modeling, this model could be used to foster exchanges among teachers or designers and to build tools for authoring CSCL scripts. It is not presented as a cognitive model of collaborative learning processes but as a design metaphor, i.e., a way to envision scripts. Its basic principle is to introduce a perturbation in a distributed system, so that the system will trigger repair mechanisms. These repair mechanisms require the knowledge-intensive interactions that the script aims to trigger.

2. EXAMPLES OF CSCL SCRIPTS

We present four scripts that we have developed and used with our own students. These examples will enable us to better describe the variety of scripts (section 3) and then to explain our design model (sections 5 and 6).

2.1 The "Concept Grid" script

The best-known collaborative script is the Jigsaw: each group member has only access to a subset of the information needed to solve the problem (Aronson et al, 1978) and therefore no individual can solve the problem alone. Group members should not simply forward information to each other: the member who owns a body of information has to process it, to become an "expert" of that sub-domain, in order to share it and to contribute to problem solving. The information given to group members defines their role. There exist multiple variations of the Jigsaw model. Some scripts alternate two types of meetings: students work in mixed groups (role-x role-y role-z), but from time to time, they form perpendicular groups, also called *expert groups* (role-x role-x role-x…) to share their expertise. In our example, knowledge distribution is induced by the script, but another script may also exploit 'natural' differences in prior knowledge: students with qualitative versus quantitative knowledge in physics (Hoppe & Ploetzner, 1999), students in medicine versus students in psychology (Hermann, Rummel, & Spada, 2001), students from different countries (Berger et al, 2001, see 2.4),…


EAO&DIDACTICIELS — Grille des concepts — Groupe 1

[1] (Saint Thomas) lois d'interaction aptitude-traitement	><	[2] (Skinner) structure modulaire	><	[3] (Anderson) modèle de l'élève	><	[4] (Bloom) pédagogie de maîtrise
><		><		><		><
[5] (Anderson) compilation	><	[6] (Bloom) prérequis	><	[7] (Saint Thomas) objectifs pédagogiques	><	[8] (Skinner) conditionnement opérant
><		<>		><		><
[9] (Saint Thomas) individualisation	<>	[10] (Skinner) frame-based	<>	[11] (Bloom) évaluation formative	<>	[12] (Anderson) drill and Practice
><		<>		><		><
[13] (Anderson) feed Back	><	[14] (Bloom) représentations mentales	<> ><	[15] (Saint Thomas) styles d'apprentissage	<>	[16] (Skinner) behaviorisme

<> : est différent de | >< : est semblable à | [auteur] : nom de l'auteur de la fiche | Concept : lien vers la fiche concept | [N°] : numéro de la case

Figure 16-1. ConceptGrid Script in phase 4: Students build a grid of concepts. Each concept links to a definition they have written in phase 3. For each symbol between cells, they write a text explaining the similarity/difference between neighbor concepts. The 2 names in the cells are their own name (blurred) and the name of the role they are playing.

We implemented an instance of Jigsaw, the Concept Grid, in a master course on learning theories for educational software. Students have to learn the key concepts of the domain and the underlying theoretical framework. Figure 16-1 shows a grid produced for the first chapter, concerning learning theories in traditional computer-based teaching. The script runs as follows:

- Phase 1. Groups of four students are freely formed. They distribute roles among themselves. Roles correspond to theoretical approaches to be learned. In Figure 16-1, the roles are Skinner, Bloom, Anderson and Saint-Thomas. New roles are proposed for each chapter except for 'Saint-Thomas': his role is to be skeptical with regards to the effectiveness of the educational software under study and hence to review experimental studies. To enter into their role, students have to read three papers describing the related theory or studies.
- Phase 2. Groups receive a list of concepts to be defined. Examples of concepts appear in the cells of Figure 16-1. They cover the key notions that the teacher expects the learners to acquire. The group distributes

concepts to be defined among its members. The teacher does not specify which role is knowledgeable for which concepts.

- Phase 3. Each student writes a 10-20 lines definition of the concepts that were allocated to him/her.
- Phase 4. Groups assemble the concepts into a grid (see Fig. 16-1) and define the relationship between grid neighbors: The "<>" and "><" symbols are links toward a short text that describes relationship between two concepts: the symbol "<>" links to explanations that discriminate similar concepts (and could be confused by students) and the symbol "><" links to explanations that articulate concepts that are apparently unrelated. Groups have to try many organisations of the concepts on the grid before being able to define all relationships.
- Phase 5. The teacher analyses all grids before the debriefing session. During this session, he points out the inconsistencies between grids produced by different groups, the cases where close concepts have not been recognized as being similar and, vice-versa, concepts that have been associated while they have a very different meaning.

This script is not fully collaborative. Phase 3 is cooperative (each student individually writes a text). The core part is Phase 4: the only way to build the grid and to define the relationship between two concepts C1 and C2 is that the student who read about C1 explains it to the student who read about C2 and vice-versa. It cannot be a shallow explanation; they have to reach a reasonable level of shared understanding to write these "relationship" texts.

2.2 The "ArgueGraph" script

The "ArgueGraph" script was used in an educational technology course. The goal of the session was that students relate courseware design with learning theories. We tested several versions of this script, within two CSCL environments, with different combinations of co-presence and distance (Jermann & Dillenbourg, 2003). It includes five phases:

- Phase 1. Each student takes a multiple-choice questionnaire produced by the teacher. The questions have no correct or wrong answer; their answers reflect theories about learning. For each choice, the students enter an argument in a free-text entry zone.
- Phase 2. The system produces a graph in which students are positioned according to their answers (Figure 16-2). A horizontal and vertical score is associated to each answer of the quiz and the students' position is simply the sum of these values. 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 have most different opinions).

- Phase 3. Pairs answer the same questionnaire together and again provide an argument. They can read their individual previous answer.
- Phase 4. For each question, the system aggregates the answers and the arguments given individually (Phase 1) and collaboratively (Phase 3). During a face-to-face debriefing session, the teacher asks students to comment on their arguments. The set of arguments covers more or less the content of the course but is completely unstructured. The role of the teacher is to organize the students' arguments into theories, to relate them, to clarify definitions, in other words, to structure emergent knowledge
- Phase 5. Each student writes a synthesis of arguments collected for a specific question. The synthesis has to be structured according to the theoretical framework introduced during the debriefing (Phase 4).

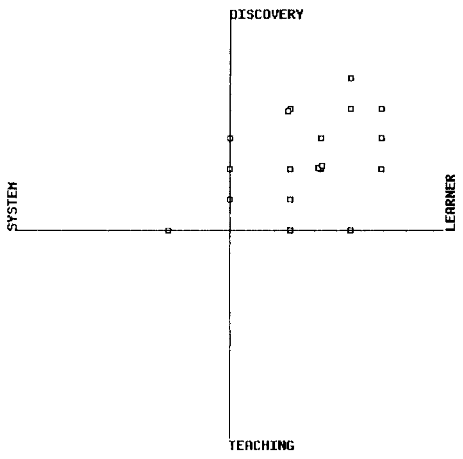
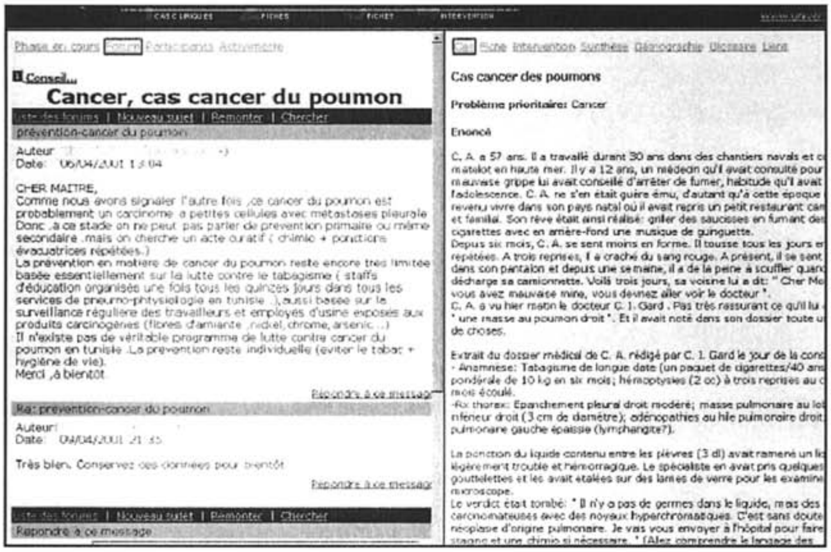


Figure 16-2. ArgueGraph, phase 3: Graph representing individual answers (names have been erased).

2.3 The "UniverSanté" script

This "UniverSanté" script was designed for teaching public health (Berger et al, 2001) in a course jointly given at the Universities of Geneva (Switzerland), Beirut (Lebanon), Monastir (Tunisia) and Yaounde (Cameroon). The students were divided into five thematic groups: AIDS, cancer, infectious diseases, cardiovascular diseases and accidents. Each thematic group includes four students of each country and a tutor. The script includes seven phases: starting from a clinical case (Phases 1 & 2), students address public health issues (3 to 5), explore methods of epidemiology (5 & 6) and build strategies to cope with public health problems (Phase 7).



Cancer, cas cancer du poumon

prévention-cancer du poumon

Auteur
Date: 06/04/2001 13:04

CHER MAITRE,
Comme nous avons signaler l'autre fois ,ce cancer du poumon est probablement un carcinome a petites cellules avec métastoses pleurale Donc ,à ce stade on ne peut pas parler de prevention primaire ou même secondaire ,mais on cherche un acte curatif (chimio + ponctions évacuatrices répétées.)
La prévention en matiere de cancer du poumon reste encore tres limitée basée essentiellement sur la lutte contre le tabagisme (staffs d'éducation organisées une fois tous les quinzes jours dans tous les services de pneumo-phtysiologie en tunisie .),aussi basee sur le surveillance reguliere des travailleurs et employes d'usine exposés aux produits carcinogènes (fibres d'amiante ,nickel, chrome, arsenic ...)
Il n'existe pas de véritable programme de lutte contre cancer du poumon en tunisie .La prevention reste individuelle (eviter le tabac + hygiène de vie).
Merci ,à bientôt

Re: prevention-cancer du poumon

Auteur:
Date: 09/04/2001 21:35

Très bien. Conservez ces données pour bientôt

Cas cancer des poumons

Problème prioritaire: Cancer

Enoncé

C. A. a 57 ans. Il a travaillé durant 30 ans dans des chantiers navals et c
matelot en haute mer. Il y a 12 ans, un médecin qu'il avait consulté pour
mauvaise grippe lui avait conseillé d'arrêter de fumer, habitude qu'il avait
l'adolescence. C. A. ne s'en était guère ému, d'autant qu'à cette époque
revenu vivre dans son pays natal où il avait repris un petit restaurant cam
et familial. Son rêve était ainsi réalisé: griller des saucisses en fumant de
cigarettes avec en arrière-fond une musique de guinguette.
Depuis six mois, C. A. se sent moins en forme. Il tousse tous les jours e
repétées. A trois reprises, il a craché du sang rouge. A présent, il se sent
dans son pantalon et depuis une semaine, il a de la peine à souffler quan
décharge sa camionnette. Voilà trois jours, sa voisine lui a dit: " Cher Mo
vous avez mauvaise mine, vous devriez aller voir le docteur ".
C. A. a vu hier matin le docteur C. J. Gard . Pas très rassurant ce qu'il lui
" une masse au poumon droit ". Et il avait noté dans son dossier toute u
de choses.

Extrait du dossier médical de C. A. rédigé par C. J. Gard le jour de la cons
- Anamnese: Tabagisme de longue date (un paquet de cigarettes/40 ans
pondérale de 10 kg en six mois; hémoptysies (2 cc) à trois reprises au c
mois écoulé.
-Rx thorax: Epanchement pleural droit modéré; masse pulmonaire au lo
inférieur droit (3 cm de diamètre); adénopathies au hile pulmonaire droit
pulmonaire gauche épaissie (lymphangite?).

La ponction du liquide contenu entre les plèvres (3 dl) avait ramené un li
légèrement trouble et hémorragique. Le spécialiste en avait pris quelques
gouttelettes et les avait étalées sur des lames de verre pour les examine
microscope.
Le verdict était tombé: " Il n'y a pas de germes dans le liquide, mais des
carcinomateuses avec des noyaux hyperchromatiques. C'est sans doute
néoplasie d'origine pulmonaire. Je vais vous envoyer à l'hôpital pour faire
staging et une chimio si nécessaire. " (Allez comprendre le langage des

Figure 16-3. A snapshot from the UniverSanté environment.

- Phase 1. Each group receives a clinical case. For example, one "cancer" group works on the case of a woman with breast cancer whereas a second "cancer" group receives a case of a man with lung cancer. Each group discusses the case in a specific forum. The tutor guides the discussion in order to help the students identify and discuss the case with regard to public health.
- Phase 2. Two groups of the same country working on the same theme (e.g., the two "cancer" groups from Monastir University) interact through an on-line forum. A synthesis of the elements identified by each thematic group is presented during a face-to-face debriefing meeting in each country.
- Phase 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 Swiss students in the cancer group create a fact sheet "Cancer-Switzerland", which they enter into the database. The "Cancer" group of every country produces the same data.

- Phase 4. The students of each thematic group from different countries discuss the differences and the similarities between the fact sheets of the four countries in the forum.
- Phase 5. Fact sheets are discussed during a face-to-face debriefing meeting in each country. The tutor prompts the students to identify any issue concerning the way in which statistical data were collected, treated or presented.
- Phase 6. Students modify their fact sheet according to the methodological comments received in Phase 5.
- Phase 7. Each thematic group is divided into two subgroups working on the cases they studied during Phase 1. Each subgroup proposes a health strategy to cope with the problem. The students enter their strategy (objectives, actions, resources, evaluation) into the knowledge base through an on-line form.

This script generates interactions by playing with differences: differences between clinical cases of the same disease (phase 2) aim at generating abstraction; differences between the statistics collected in different countries generate discussion on the salience of the disease (phase 4) but also on the methods for collecting comparable data (phase 5). Comparison of the different societal answers to disease generates awareness of the public health policies.

2.4 The "Studio" script

As last example, our Courseware Design Studio is an adaptation from the PhaseX script (Engeli, 2001) for supporting project-based learning. The goal of the project was to design a courseware. The project is segmented into phases. At each phase, all teams deposit their intermediate product in a shared space. At the next phase, each team is allowed to borrow the work produced by another team and to continue its work from it. The phases were goal definition, content analysis, activity design, and so forth. The rationale for this script is that the shared space allows for a permanent idea-seeding. However, while it seems to work very well in Engeli's 3D-design projects, our students were reluctant to exchange intermediate results in their design process.

3. THE DIVERSITY OF SCRIPTS

This book presents a variety of scripts. Our scripts illustrate different script schemata but are still rather similar to each other compared to other examples in this book. This section reviews different understandings of a script while the next section specifies categories within our own scripts.

3.1 Role: Why playing a script?

For Kollar, Fischer, and Hesse (in press) and King (this volume), the term "external script" refers to the pedagogical scenario that students are asked to play, while the term "internal script" describes the mental representation that students construct of the external script. The external script is a didactic artifact to be used during a training session. The internal script is a cognitive structure that, in many cases, existed before the training session (e.g., "How to argue with a peer?") and will continue to exist after the training session. When the goal is that students internalize the script in order to reuse it in future situations, the script is a pedagogical *objective*. This is for instance the case of the reciprocal teaching script (Palincsar & Brown, 1984) which effectively fostered a high level of internalization. More modestly, the internalization of our Studio script was also an objective for our students, since the segmentation of courseware design into phases was something they had to learn.

When the script is "only" a method to be used during a training session and not internalized for the future, students still have to build some internal script in order to be able to participate in the learning activities. We did not expect our students to remember the ArgueGraph or the ConceptGrid scripts a few weeks later; we expected them to have learned the content being discussed in the script but not the script itself.

In summary, when the script is a method, the internal script is instrumental to play well the external script; when the internal script is the objective, it's the other way around. These are not exclusive: an argumentation script in which roles rotate may have as objectives both the content of argumentation (script as a method) and the ability to take the other's perspective (script as an objective). It is important to make explicit the status of a script before conducting an empirical study because they imply different forms of assessment, such as transfer task when the script is the objective and knowledge task when the content is the objective.

Finally, Harrer, Bollen, and Hoppe (2004) use *scripting collaboration* to refer to another pedagogical method: the post-hoc analysis of the interaction log files by the students themselves. This reflective activity is namely a useful phase when the script needs to be internalized. Our scripts are prescriptive while their approach is descriptive.

3.2 Congruence: Do they play the script?

When the teacher sets up an (external) script for the students, each of them constructs some internal script that will – to some extent – be different from the external script. Within a group, since students develop their own internal script, the interactions that actually take place will – to some extent – drift away from the interactions prescribed by the script. The congruence between the external script and emergent interaction patterns depends upon four script features: the degree of coercion, the intelligibility of the script, the degree of granularity and its fit to the team distribution. We now review these four congruence factors.

The first congruence factor is the *degree of coercion* of the script. A script may be simply conveyed through initial instructions or be regularly enforced by prompts or other design features. Although this is a continuous variable, we identified five levels of coercion (Dillenbourg, 2002) presented in increasing order

1. 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 shape the collaborative processes.
2. 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's expectations are made explicit, but they can of course be misunderstood, incorrectly applied, forgotten or completely ignored. Students have to build an internal script that corresponds to the external script presented by the teacher.
3. Trained scripts. Students are trained to collaborate in a certain way before using the script it in a real learning situation. The degree of coercion is higher than in the instructed scripts since the teacher may control the student's internal script.
4. Prompted scripts: The system displays cues that encourage the learners to take their respective role (Weinberger, Fischer, & Mandl, 2002). Their system delivers cues (text messages), that are supposed to lead students to take specific roles such as "analyzer" or "critic".
5. Follow-me scripts. Students interact with an environment that does not allow them to escape from the script.

A high degree of coercion reduces the gap between the external script and emergent interaction patterns but increases the risk of *overscripting* (see 4). Our scripts have a low degree of coercion, obtained in various ways. In the ArgueGraph, the coercitive factor was the interface. In a first environment we used, pairs could only provide one answer per question and argumentation was more intensive than in a second environment, where the interface enabled them to enter more subtle answers. In the ConceptGrid, coercion was induced by the grid structure which forces the students to explain concepts to each other. The UniverSanté degree of coercion was very low and tutors had to permanently reinforce the script. In the Studio script, the most coercitive feature was the linear structure of the project segmentation. When coercion is naturally induced by the interface, as in ArgueGraph, we could talk about *affordances*, which sound more positive than coercion.

The second congruence factor is the *intelligibility* of the script. We face intelligibility problems with the UniverSanté script that occurred to be too complex (Berger et al, 2001) in this international public health course (students from Switzerland, Lebanon). Since we were aware of the script complexity, we provided teams with a graphical representation of the script and offered a close follow-up by teaching assistants, but nonetheless the students – and even some tutors – did not manage to construct a clear internal script. The interaction patterns drifted away from the external script.

The third congruence factor, *granularity,* refers to the time scale (duration of each phase) and the grain size of phases (subtasks) definition. For instance, the Studio script included a "programming" phase that lasted four weeks, the whole script running over the academic year, while the ArgueGraph script ran over four hours with phases ranging from 5 to 100 minutes. At the lower end, finest grain scripts reach the utterance level, i.e., specify the authorized dialogue moves at the next utterance. Fine grained scripts tend to be more coercitive. The gap between the external script and emergent interaction patterns may increase if there is a mismatch between the natural granularity of the task and the granularity enforced by the script. A mismatch could occur if the questions in ArgueGraph or the concepts in the ConceptGrid were too specific to capture the key differences between the theories under scrutiny. Another mismatch would occur if the Studio script structured a design phase as a sequence of questions while designers would address these questions in parallel.

The fourth congruence factor, *fitness*, is important for scripts that 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 interaction patterns depend on the good match between the role requirements and the group members' skills or profiles. Fitness inspired various jokes such as "If the French member is in charge of cooking and the German one in charge of organization... (high fitness), but, if it is the other way around..." Low fitness is detrimental to role adoption and role adherence (students do not stick to the roles very long). Fitness inherits from transactive memory (Moreland, 1999), that is the representation that each group member has of the skills of the others: what matters is not only that team members are able to play their role but also that their team mates believe they are able to play that role. We did not encounter fitness problems in the ConceptGrid, for instance cases where one student would not manage to play "Skinner" for personal reasons. The fact they choose the roles themselves probably increases fitness. The fitness question is a greater concern in project-oriented scripts that select one student as team leader.

3.3 Granularity: Macro versus micro-scripting

We introduced the notion of script granularity as a continuous variable. There is however a qualitative difference between *macro* and *micro scripts*. Let us illustrate these differences with scripts that aim at raising argumentation. A micro-script scaffolds the interaction process per se: when learners state a hypothesis, the script will for instance prompt their peer to produce counter-evidence. A macro-script sets up pairs in which argumentation should occur, as in the ArgueGraph, by pairing students with opposite opinions. The micro-script reflects a psychological perspective, acting on the internal script (scripting as a goal), while the macro-script reflects an educational perspective, influencing the process more indirectly (scripting as a method). Micro and macro-scripts do not constitute clear-cut categories but rather define a continuum. Most examples described in this volume are on the "micro" side: in the work reported by King (this volume), by Lauer and Trahash (this volume), by Weinberger et al. (this volume), by Carmien et al. (this volume), the script includes prompts that directly scaffold interactions and is expected to be internalized as higher-order thinking skills (argumentation, problem solving or metacognition). The grain size is somewhat coarser in the scripts of Rummel and Spada (this volume), and Ertl et al. (this volume), where the script prompts episodes of interactions. The example presented by Kolodner (this volume) is, like our examples, on the macro side. Ayala (this volume), and Haake and Pfister (this volume) describe environments that articulate micro-scripts within phases of a macro-script.

3.4 Integrated learning

We use CSCL scripts for promoting a vision of e-learning that is broader than what the *CSCL* label may indicate. Our script examples are neither strictly collaborative, nor strictly computerized; they illustrate our *integrated learning* approach that we define with 3 features:

- Despite the first *C* in *CSCL*, there is no reason to restrict CSCL scripts to distance interactions. ArgueGraph and ConceptGrid scripts have mostly been used in a situation where students were co-present. UniverSanté used distant interactions, since geographical diversity was the key principle, but still included key face-to-face discussion (one per country). Computers are justified by other reasons than simply connecting distant learners (see section 4). 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 articulate activities which are on-line or not, in front of a computer or not, occurring across 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. Integration is pedagogical but also functional: scripts support data flow between multiple activities (see 7.4). For instance, in the ArgueGraph, the individual answers (phase 1) are used to form pairs (phase 2) and the pairs' answers and arguments are collected for the debriefing (phase 4).

- Despite the second *C* in *CSCL*, there is no reason why collaborative learning should be treated as an exclusive pedagogical approach. Instead, group activities gain from being integrated with other classroom activities. Scripts may include individual work (e.g., writing a synthesis, reading a paper,...) and/or class-wide activities (introductory lectures, debriefing, ...). In ArgueGraph, phases 1 (answering the quiz) and 5 (writing a summary) are individual while phases 2 (observing the graph) and 4 (debriefing) are done with the whole class. In the ConceptGrid, phase 3 is individual (reading papers and writing concept definitions) while phase 5 (debriefing) is at the class level. The designers' challenge is to integrate these diverse activities within one consistent script.
- Last but not least, the illustrated scripts maintain the teacher in his leading role. He or she is not properly teaching but is active and salient as the *chef d'orchestre* of the whole script: he or she may shorten a phase, regulate groups, give feedback, etc. We therefore should be concerned by the script *flexibility*, i.e., the possibility for the teacher to modify the script on the fly (see section 4).

These features define what we refer to *as integrated learning*, a pedagogical approach that is broader than the approach indicated by the terms *collaborative* and *computer* in CSCL. However, the breadth of this concept may weaken the identity of a *script*. Are scripts just a trendy word to refer to lesson plans? No! CSCL scripts are instructional sequences in which peer interactions are targeted to be the core learning mechanism. Therefore our design model distinguishes the core script, which governs collaborative interactions, from the didactic envelope, that encloses the core activities into other activities, forming the integrated learning approach.

4. BENEFITS AND RISKS IN COMPUTERIZED SCRIPTS

This volume concerns scripts in computerized environments. What is the added value that technology brings to the use of scripts? What are the drawbacks? We start with the advantages:

- *Connecting*: When scripts include remote activities, technology is simply the communication tool.
- *Sharing*: Computers provide a space for sharing products, allowing teams to get inspired by what other teams produce, as in the Studio script. This *simple* feature is important, as long as plagiarism can be controlled.
- *Management*: Computerized scripts off-load teachers from some logistics duties such as time management (reminding deadlines, ...) and information flows (e.g., distributing data to different group members).
- *Reification*: Computerized scripts provide students with a concrete representation of the external script, which is dynamically updated.
- *Scaffolding*: Computerized scripts offer opportunities for shaping communication with semi-structured communication interfaces and dialogue grammars or both (as illustrated by Runde et al, this volume).
- *Traceability*: Computerized scripts enable recording interactions and outputs, which, despite privacy concerns, enable teachers to analyze and regulate teamwork and enable students to reflect upon previous steps.
- *Adaptivity*: Computerized scripts enable *dynamically* generated events that would be harder to create without computers, such as, in the ArgueGraph, finding peers with most opposite opinions. Real-time adaptations can be improved by real time analysis of interactions among peers (e.g., Soller, Martinez, Jermann, and Muehlenbrock, 2005).

Among the drawbacks of computerized scripts, we find the general disadvantages of computer-mediated communication versus face-to-face communication. It is not the place here to review them (see Bromme, Hesse, & Spada, 2005). With integrated scripts, these drawbacks are compensated by face-to-face situations (see 3.4).

A key problem is the loss of *flexibility*. Good teachers adapt their plans on the fly, while a computerized script can hardly be modified in real time. Of course, the very idea of a script implies a decrease of flexibility: a script aims at structuring group processes, which requires some rigidity. However, implementing the script often generates constraints that are not part of the pedagogical intentions. Designers have to disentangle the flexibility loss inherent to the pedagogical intentions from the flexibility loss that is an undesired effect of translating the script idea into a computer program (Dillenbourg & Tchounikine, accepted).

Another risk is what we called *over-scripting* (Dillenbourg, 2002), i.e., situations that constrain natural collaboration in a way that makes it sterile, inhibiting the natural peer interaction mechanisms. Factors of over-scripting are:

- Disturbing natural interactions. If a learner wants to express A while the CSCL system only offers means for interactions B or C, either the learner will fail to say what he wanted to say or he will pervert the system (e.g., re-purpose B to say A). If similar breakdowns occur frequently, they may spoil the collaboration process. This risk concerns scripts that cumulate a high degree of granularity and a high degree of coercion.
- Disturbing natural problem solving processes. A script usually segments a global task into a sequence of activities. In our Studio script, this segmentation was a problem for students who had a holistic approach of courseware design. The script proposed an approach that was very linear. Some students rejected this artificial linearization. Our Grid script also introduces coercion with respect to the task: it is easier to draw a free concept map than to arrange concepts on a two dimensional grid. To some degree, coercion may become incompatible with the students' cognitive processes. Overscripting may then make the task so hard that it spoils the students' motivation.
- Increasing cognitive load. Complex scripts may interfere with the main learning process by augmenting the learners' cognitive load. The extraneous load comes from the necessity to understand, memorize and execute the script. However, an alternative hypothesis is that scripts reduce cognitive load by partly offloading interaction management (Dillenbourg & Bétrancourt, 2006).
- "Didactising" collaborative interactions. Collaborative problem solving triggers natural interactions. A peer asks a question because he wants to know the answer, while a teacher usually asks questions which he already knows the answer to. Peers negotiate a concept when they disagree on interpreting the phenomenon they jointly observed while teachers discuss concepts for which they own the right definition. A danger of "didacticised" interactions is to miss the engagement that is expected from genuine collaboration.
- Goalless interactions. Collaboration is driven by a shared goal. Scripts being quite didactic, they may prevent students from adopting the script goals as their own goals. The more the scripts segment collaboration into subprocesses, the more it seems difficult for team members to forget the didactic nature of the script.

Pitfalls are numerous; scripts need to be thoughtfully designed. The rest of this chapter investigates the design of CSCL scripts.

5. THE STRUCTURE OF SCRIPTS

Integrated learning scripts include a kernel, the core script, and a set of pre- and post-structuring activities, the didactic envelope. The core script is the collaborative activity in which the interactions that the script is intended to trigger should appear. In the ArgueGraph script, the core activity is the formation of conflicting pairs and argumentation triggered for answering the questionnaire together (Phase 4). In the ConceptGrid script, the core activity is the distribution of knowledge and the mutual explanation process necessary to build the grid (Phase 4). In the UniverSanté script, the core activity is when students must identify similarities and dissimilarities between the ways different national health systems cope with the same medical issue. The core script defines how the knowledge or task is *distributed* over the group members. We therefore borrow the distributed cognition model as explained in section 6.

The didactic envelope encloses the core script with other activities that contribute to the script consistency. Pre-structuring activities provide the conditions necessary to make the core script activities work well: introductory lectures, readings, exercises to activate pre-requisite skills, metaphors, etc. They namely enable students to play their role in the script. Post-structuring activities include debriefing activities such as the comparison of multiple solutions, synthesis lectures or readings, summary writing, etc. These are mostly reflective activities, aimed at turning group experience into knowledge. The activities in the envelope make the difference between collaborative learning in a restricted meaning and integrated learning, as explained in section 3.4.

The envelope has two salient features, its temporal structure and its social structure. A clear time structure differentiates scripts from free collaboration: scripts define a sequence of phases and in many cases these phases are limited in time. The rationale for setting up a semi-rigid time frame is threefold:

- Time management is a critical factor in everyday educational practice, for both teachers and learners. It is even more important for web-based activities taking place outside the time habits that exist in schools.
- The time structure facilitates teacher regulation by providing him or her with an easy way to follow the teams' progress.
- The time structure makes the task distribution more salient, especially since deadlines define clear boundaries between consecutive subtasks.

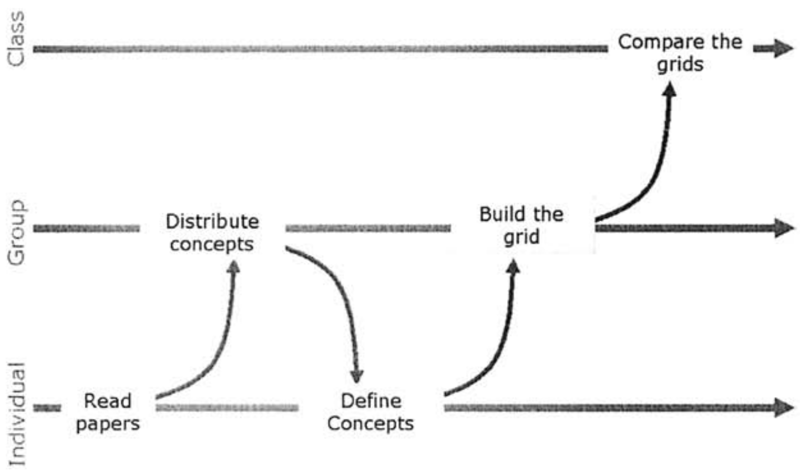


Figure 16-4. Structure of the 'ConceptGrid' script, time is represented horizontally and the social structure vertically.

The second dimension of integrated learning scripts is their social structure: activities occur at different social planes. Vygostky (1978) 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, group size is a continuous variable, but there is a cognitive threshold: group activities occur at the inter-psychological plane as long as team members maintain some representation of their teammates' cognition; the social plane is the level where individual representations disappear behind the culture that the community members jointly constructed. If we relate these psychological levels to CSCL environments, we usually observe five levels of activity:

- Individual Plane: Solo activities.
- Group Plane: Activities in small groups ranging from two to, let's say, eight people. This is where proper collaboration occurs.
- Class Plane: Activities involving all students enrolled in the same course. We also refer to them as collective activities.
- Community Plane: Activities that involve external but identified actors such as other classes, expert groups, families. For instance, when a class

from school X designs a mathematical challenge for all other classes in the community, this activity is at the community level.

- World Plane: Activities that are accessible 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 user may vote.

What matters here is not to agree on the exact definition of the levels but to stress the fact that script activities define moves across multiple planes. Figure 16-4 illustrates the time by social structure of the script "Concept Grid". One could argue that activities always occur at multiple planes: individual cognition does not freeze during class interactions and culture does not stop shaping our thinking during individual work. Activities do occur in parallel on multiple planes, but their focus varies with time.

The curved arrows on Figure 16-4 represent what we called *functional integration* in section 3.4. Functional integration refers to dataflow between activities at different planes. The output of an activity A_i at social level N is later on reused by an activity A_{i+1} at social level M, in many cases, N being different from M. This dataflow may appear as a technical feature, but in fact it affords the design of innovative scripts by combining storing, processing, distributing and representing data during collaborative learning. These data are student productions (answers in the ArgueGraph, concept definitions in the ConceptGrid and deliverables in the Studio) and student interactions (e.g., their arguments in the UniverSanté). We describe dataflow operators in section 7.4.

6. THE SWISH MODEL

How to design the core script activities? We propose a model, called SWISH, which borrows the distributed cognition vision, according to which a group of actors and the tools they use can be understood as a single cognitive system. The components of the system are the students who participate in the scripted teamwork as well as the tools and resources available. The script itself can be considered as a tool that shapes the functioning of the distributed system.

The core script defines the organization of a distributed cognitive system i.e., which team member will perform which subtasks. We refer to subtasks 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. Scripts often define roles that induce a somewhat natural distribution of work into subtasks.

Why would we formalize task distribution while we aim to support collaboration? A formal task division appears to be in contradiction with the close interactions expected in collaboration (Dillenbourg, 1999). Since collaborative learning is often defined as the process of constructing and maintaining a shared understanding of the task (Roschelle & Teasley, 1995), it may sound counter-intuitive to split the task among different learners: this opens the door to misalignment of views, understandings and goals. To the same extent, scripts that foster conflict among peers would be detrimental to the construction of a joint solution. To bypass this counter-intuition, we 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, etc.) engaged to develop this shared understanding. Scripts that trouble a smooth collaboration increase the cognitive effort and hence are expected to augment the learning outcomes. In other words, learning results from over-compensating the drawbacks of task distribution.

This principle is the base of our design model: "Split Where Interaction Should Happen". SWISH can be formulated in three points:

1. Learning results from the interactions students engage in while constructing a shared understanding of the task *despite* the fact that the task is distributed.
2. Hence, the task distribution determines the nature of interactions. Interactions are mechanisms for *overcoming* task splits.
3. Hence task splits can be, following some kind of *reverse engineering*, designed for triggering the interactions that the designer wants to foster: Split Where Interaction Should Happen.

This model can be applied for describing the main script schemata, i.e., classes of scripts. We distinguish three basic schemata:

- The *jigsaw schema* distributes the knowledge or information necessary to solve the task, either by forming pairs that have complementary knowledge (e.g., in UniverSanté, students from different countries import knowledge of their national health system) or by providing them with complementary information (e.g., different readings in the ConceptGrid). Since none of the group members has enough information or knowledge to solve the task alone, they need to explain or justify their knowledge or contribution to others. For describing the ConceptGrid in SWISH terms, the split is performed by distributing information and it is compensated by explaining concepts to each other.
- The *conflict schema* triggers argumentation among group members by forming pairs of students with conflicting opinions (e.g., ArgueGraph), by providing them with conflicting evidence or by asking them to play conflicting roles. For describing the ArgueGraph in SWISH terms, the

split is performed by finding peers with conflicting opinion and it is compensated by argumentation.
- The *reciprocal schema* defines two roles in teams, one of the peers regulating the other and then switching roles. A well known example is the reciprocal teaching approach (Palincsar & Brown, 1984). For describing the reciprocal tutoring script in SWISH terms, the split is performed horizontally, between cognitive and metacognitive layers of the task and is compensated by mutual regulation. Since the cognitive and metacognitive subprocesses need to remain tightly coupled, the only way to build a shared solution is that peers continuously engage in mutual regulation interactions.

7. GENERALIZING SCRIPTS

As any pedagogical method, scripts raise hopes of generalization: can we reuse these scripts to teach a large variety of contents? The ArgueGraph script can be used for different subject matters but is only relevant in domains where key notions can be argued about. The ConceptGrid script can be generalized to many conceptual sets, but not all conceptual domains can be segmented as in the grid. The UniverSanté was very specific to the content to be taught, public health: using national differences is a natural way to let students discover the variety of societal answers to a similar medical problem. The Studio script can be generalized to a variety of design processes but with the constraint that this design process should be rather linear.

Generalisability is not bound by classical scientific boundaries (e.g., a script would be good for mathematics but not for social sciences) but by the specific learning objectives (ArgueGraph could be used in mathematics if students argue to choose among three ways to compute a value). In other words, there is definitely a potential of generalisability; a script is not universally relevant but can be reused in various domains.

7.1 Descriptive model

Scripts can be defined as variations of a generic template with a limited set of attributes. Most scripts can be defined with a limited number of components (groups, participants, roles, activities and resources) and mechanisms that capture the dynamics of scripts, i.e., how individual learners are distributed over groups (group formation), how roles, activities or resources are distributed over participants (component distribution) and how both components are distributed over time (sequencing) (Kobbe, Weinberger, Dillenbourg, Harrer, Hämälainen, & Fischer, submitted).

This simple description scheme could be translated within an educational modeling language (EML). These languages propose a well-structured terminology for describing instructional sequences. They do constitute a step forward compared to the content-centric approach of the educational metadata initiatives. However, they do not constitute a design model; they provide a description of the scripts but fail to capture the core idea of a script. A design model should describe the mechanisms by which the script is expected to generate learning. IMS Learning Design[1] could be expanded to model the core script, but 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 teaching script where roles rotate among group members at each script phase. The social structure of a script should be explicitly represented in the model.

Instead of producing yet another pedagogically neutral authoring language, we deliberately aim for a non-neutral model, i.e., a modeling scheme that conveys specific pedagogical ideas. This is the condition to produce scripts that differ from genuine lesson plans. Therefore, instead of looking for a highly abstract modeling scheme, we identify classes of similar scripts and infer their core idea, their identity.

7.2 Script schemata

Despite the diversity of scripts, there are recurrent patterns. We called them *schemata* instead of *patterns* to avoid confusions with the term *design pattern*, which has a more technical meaning in software engineering. A schema simply indicates commonalities among scripts, independently from the algorithms used by the CSCL environments to support these scripts. For instance, scripts that belong to the jigsaw schema have in common to distribute the necessary information among team members. Schemata are more abstract than programming structures, but if we translate them into software components, we could design tools that reduce the computational burden of CSCL script construction.

In section 6, we described three types of script schemata, the jigsaw schema, the conflict schema and the reciprocal schema. Other methods for group-based learning can also be described as script schemata:

- The *project* schema defines phases of a project, roles among teams (moderator, leader, writer, ...) and a calendar of intermediate deliverables. These scripts vary in coercion: does each team work on the same project; are they free to define the phases of their work and the calendar. The focus is often put on the regulation of project work.

[1] http://www.imsglobal.org/learningdesign

- *Problem-based learning* (Koschmann, Kelson, Feltovich, & Barrows, 1996) covers a variety of scripts that, despite differences, include similar phases: analysing the problem, defining learning objectives, acquiring the necessary knowledge and solving the problem collectively.
- The *science making* schema includes scripts in which team work is structured into a sequence of phases that drive learners through the scientific process of knowledge construction, as researchers are supposed to do. One example for these schemata is inquiry based learning (Hakkarainen & Sintonen, 2002).

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 specific learning objectives, the peculiarities of the target audience and the specific content.

7.3 Generalization hierarchy

The ConceptGrid is a subclass of jigsaw schema. We could reuse the same script but replacing the Cartesian grid used in Phase 4 by a graphical concept map. The diversity of links between concepts that is offered in a concept map might be more appropriate to complex semantic fields. This new script, let's call it *ConceptGraph*, and the ConceptGrid are two subclasses of a higher script class, let's call it ConceptStructure.

In the ArgueGraph, a subclass of the argumentation schema, pairs are formed on the basis of their distance on the graph. The distance is computed by associating an X- and Y- value to each answer. These values are not computed in a scientific way, they are arbitrarily fixed by the designer. Their interest is to provide positions on the map with a semantic value. To avoid this arbitrary value allocation, one could use an algorithm that forms pairs of students with the lowest number of common answers. Let's call this new script *ArgueList* and their super-class *ArgueFromQuizz*. In another version, rather than using a chat or face-to-face discussion, we could have students argue with a semi-structured communication interface such as Belvedere (Suthers et al, 2001).

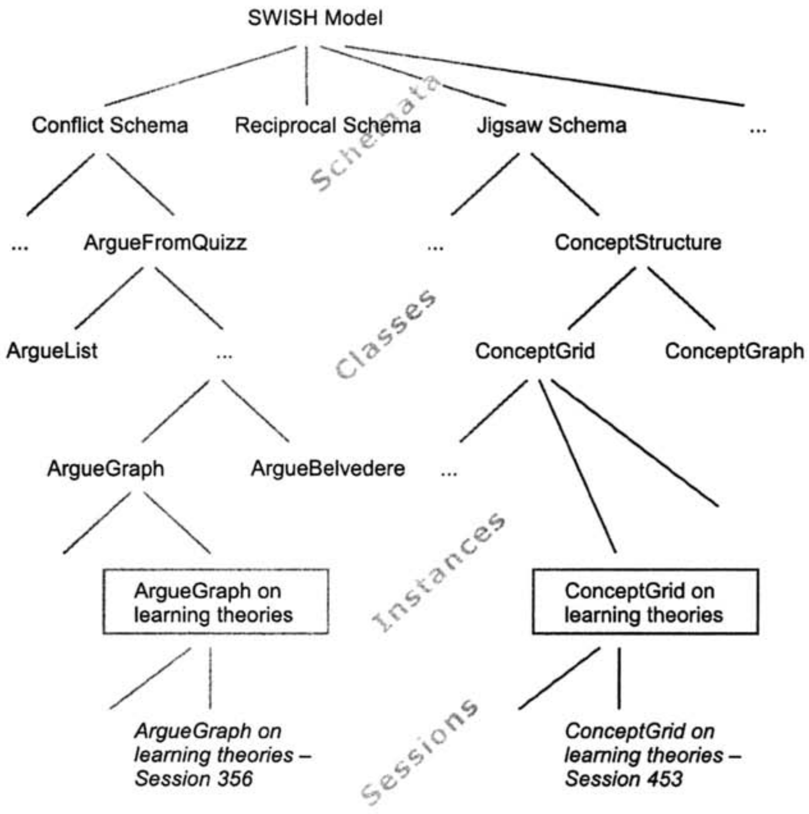


Figure 16-5. Generalization hierarchy for CSCL scripts.

Figure 16-5 represents the hierarchy of generalization. We arbitrarily discriminate four levels:

- Schemata describe the core mechanism of a large set of scripts.
- Script classes and subclasses define scripts, including their didactic envelope, independently from a specific content.
- Instances are scripts that have been instantiated with a specific content.
- Sessions are scripts instances with the student-specific data (users per groups, deliverables, …), dates, etc.

This hierarchy is not a proper tree: A script may borrow ideas from several schemata. The UniverSanté script for instance plays with both the complementarity (Jigsaw schema) and the conflicts among students knowledge.

One could argue that this tree-like representation is not appropriate. For instance, in the ArgueFromQuizz family (a subgraph), the mode of pair formation (Phase 2: graph distance versus common answers) is independent from the mode of argumentation (Phase 4: free versus semi-structured dialogues). Each of the pair formation modes could be combined with each of the argumentation modes. A script grammar, combined with these different modes as vocabulary, would be more powerful for describing all possible combinations. However, syntax may not carry semantics. A combinatorial approach may lead to assemble script elements into something that does not constitute a script, i.e., a sequence of events that will not trigger specific interactions. Script classes make the design space discrete, which is a simplification, but enables to convey the design rationale.

7.4 Executable model

As pointed out by Kobbe et al (submitted), a script can be defined as a number of mechanisms that manipulate a set of script components (roles, activities, ...). Some of these components are intrinsic to the script class (e.g., the grid structure of ConceptGrid; some prompts in ArgueGraph), some objects are specific to the script instance (e.g., the questions included in an ArgueGraph on biology; the list of documents to read in a ConceptGrid on history) and some objects are specific to a script session (e.g., the definition produced by ConceptGrid students; the answers produced by ArgueGraph students). An executable model of script has to manipulate these objects, e.g., to allocate individuals to groups or roles to individuals, to gather answers within a group or conversely to distribute data among group members, etc. We expect these mechanisms to be formalized as the combination of a limited number of basic operators.

Dataflow operators. Dataflow enables the design of dynamic CSCL scripts. The dataflow used in our scripts can be described by a small number of operators for moving up and down the planes of the social structure. The research in producing these operators should benefit from the advances on workflow technology, namely workflow standards such as WFXML[2].

- *Upward* operators are aggregate, list, differentiate, etc. They collect data at a social plane and turn them into a data structure at higher social plane. The type of processing depends on the nature of data. If data are structured in a table with social planes (individuals, groups, ...) in rows and task outputs in columns, we can define simple operators. The *aggregation* operator collapses columns, e.g., computes a value (sum, mean, ...) for all individuals. The *differentiation* operator collapses columns (data)

[2] http://www.wfmc.org/standards/wfxml_demo.htm

for each user. It is used in the ArgueGraph, where the answers to the 10 questions are summarized into a [X Y] pair for each individual in order to plot them on the graph. When data are too complex to be turned into a single value, they can simply be listed as it is the case in the Studio script (*list operator*).

- *Downward* operators distribute an object among members of the lower social plane. For instance, in the ConceptGrid, each group of four students (social plane 2) is associated with four roles and 12 readings. A downward operator would distribute the roles and readings to each team members (social plane 1). As for aggregation, the simplest operator is non-transformational: it distributes the same object O from plane N to all members at plane N-1.

Social operators. Other operators transform the structure of the groups either by reallocating roles within a group (role rotation) or by moving individuals between groups (group rotation).

- The *role rotation* operator redistributes the roles (subtasks) among group members at different phases. Role rotation reinforces the distributed system model that underlies the SWISH model. A set of interrelated components can be depicted as "a system" if it is capable of plasticity, i.e., to reallocate dynamically subtasks to different subcomponents. The rotation operator enforces this plasticity.
- The *group rotation* operator redistributes individuals among teams. It is applied in scripts where individuals are member of two groups, namely in Jigsaw scripts, where an individual sometimes works with his or her team but sometimes works with the individuals that have the same role in other teams.
- The *group formation* operators determine how groups are formed from individuals: it relies on the difference of opinions in ArgueGraph and the complementary of knowledge in Hoppe and Ploetzner's (1999) scripts.

These few examples of operators stress both the usefulness and the complexity of developing abstract mechanisms that would apply to a variety of script domains.

8. SYNTHESIS

The SWISH model can be explained simply. First, one introduces a perturbation in a distributed system, by splitting it. Second, the system triggers repair mechanisms for reducing the perturbation. These repair mechanisms – hopefully – are knowledge-intensive interactions that produce learning.

Learning is therefore the result of over-compensating the drawback of task splits.

However, this model only holds if the group has both the ability and the will to compensate task splits. In some cases, solving conflicts, explaining complex concepts or regulating bad problem solvers may be beyond the skills of individuals. In other situations, the motivation to reach a shared understanding may be insufficient. The SWISH model is only valid for tasks that require a high level of shared understanding. If students manage to solve the task without constructing a shared understanding, repairing the system will not be worth the effort.

SWISH is not a cognitive model grounded in experimental results. We used these scripts with our own students, but only two of them have been formally assessed. However, no script could be proved to be generally effective. We cannot establish the effectiveness of a script class in general since it depends on its relevance for specific learning objectives and target groups. Nonetheless, by describing CSCL scripts in a structured way, this chapter may help researchers to clarify the variables they investigate when running experimental studies.

This framework contributes to design tools for authoring CSCL scripts. Most scripts are implemented in specific CSCL environments. Our script examples were implemented as dynamic web pages, generated with PHP programs from database contents (MySQL). Not all teachers can install a database and write PHP. Tools for authoring CSCL scripts aim at promoting practices of e-learning that are more innovative than those offered by existing learning management systems.

ACKNOWLEDGEMENTS

This chapter benefited from the discussions within CoSSICLE a research group of KALEIDOSCOPE, a European Network of Excellence on "*Concepts and methods for exploring the future of learning with digital technologies*", funded by the European Union and the Swiss Federal Office for Education and Science. This work is also partly funded by the Swiss Center for Innovation in Learning (SCIL; University of St Gallen, Switzerland). This chapter has been enriched by comments from Frank Fischer, Armin Weinberger, Karsten Stegmann, and the book reviewers.

REFERENCES

Berger, A., Moretti, R., Chastonay, P., Dillenbourg, P., Bchir, A., Baddoura, R., et al. (2001). Teaching community health by exploiting international socio-cultural and economical differences. In P. Dillenbourg, A. Eurelings, & K. Hakkarainen (Eds.). *Proceedings of the first European Conference on Computer Supported Collaborative Learning* (pp. 97-105). Maastricht.

Bromme, R., Hesse, F., & Spada, H. (Eds.) (2005). *Barriers and biases in computer-mediated knowledge communication. Computer-Supported Collaborative Learning Series.* New York: Springer.

Brousseau, G. (1998). Théorie des situations didactiques. Grenoble: La Pensée Sauvage.

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

Dillenbourg, P., & Bétrancourt, M. (2006). Collaboration Load. In J. Elen & R. E. Clark (Eds.), *Handling complexity in learning environments: research and theory* (pp. 142-163). Advances in Learning and Instruction Series, Pergamon

Dillenbourg, P. (2002). Over-scripting CSCL: The risks of blending collaborative learning with instructional design. In P. A. Kirschner (Ed.), *Three worlds of CSCL. Can we support CSCL* (pp. 61-91). Heerlen: Open Universiteit Nederland.

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

Dillenbourg, P., & Tchounikine, P. (in press). Flexibility in macro-scripts for CSCL. *Journal of computer assisted learning.*

Engeli, M. (Ed.). (2001). *Bits and spaces, architecture and computing for physical, digital, hybrid realms.* Basel: Birkhäuser Publishers.

Hakkarainen, K., & Sintonen, M. (2002). The interrogative model of inquiry and computer-supported collaborative learning. *Science & Education, 11*(1), 25-43.

Harrer, A., Bollen, L., & Hoppe, U. (2004). Processing and transforming collaborative learning protocols for learner's reflection and tutor's evaluation. In E. Gaudioso & L. Talavera (Eds.), *Proceedings of the European Conference on Artificial Intelligence.* Valencia.

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., & Dillenbourg, 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. Computer-Supported Collaborative Learning Series.* Amsterdam: Kluwer.

Kobbe, L., Weinberger, A., Dillenbourg, P., Harrer, A., Hämäläinen, R., & Fischer, F. (submitted). *Specifying Computer-Supported Collaboration Scripts.*

Kollar, I., Fischer, F., & Hesse, F. W. (in press). Computer-supported collaboration scripts - a conceptual analysis. *Educational Psychology Review.*

Koschmann, T., Kelson, A. C., Feltovich, P. J., & Barrows, H. S. (1996). Computer-supported problem-based learning: A principled approach to the use of computers in collaborative learning. In T. Koschmann (Ed.), *CSCL: Theory and practice of an emerging paradigm.* Mahwah, NJ: Lawrence Erlbaum Associates.

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

Moreland, R. L. (1999). Transactive memory: Learning who knows what in work groups and organizations. In L. Thompson, D. Messick, & J. Levine (Eds.), *Shared cognition in organizations: The management of knowledge* (pp. 3-31). Mahwah, NJ: Lawrence Erlbaum Associates.

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 & N. Miller (Eds.), *Interaction in cooperative groups: The theoretical anatomy of group learning* (pp. 120-141). London: Cambridge University Press.

Palincsar A. S., & Brown A. L. (1984). Reciprocal teaching of comprehension-fostering and comprehension-monitoring activities. *Cognition and Instruction, 1*(2), 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.

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

Soller, A., Martinez, A., Jermann, P., & Muehlenbrock. M. (2005). From Mirroring to Guiding: A Review of State of the Art Technology for Supporting Collaborative Learning. *International Journal of Artificial Intelligence in Education*, 15, 261-290.

Suthers, D., Connelly, J., Lesgold, A., Paolucci, M., Toth, E., Toth, J., et al. (2001). Representational and Advisory Guidance for Students Learning Scientific Inquiry. In K. D. & P. J. Feltovich (Eds.), *Smart machines in education: The coming revolution in educational technology* (pp. 7-35). Menlo Park, CA: AAAI/Mit Press.

Vygotsky, L. S. (1978). *Mind in society. The development of higher psychological processes.* Cambridge, MA: Harvard 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. Proceedings of the International Conference on Computer Support for Collaborative Learning (CSCL) 2002* (pp. 573-574). Hillsdale, NJ: Lawrence Erlbaum Associates.