Chapter 22
Computer Supported Collaborative Learning and Intelligent Tutoring Systems

Pierre Tchounikine[1], Nikol Rummel[2], and Bruce M. McLaren[3,4]

[1] Université Joseph Fourier - BP 53 - 38041 Grenoble Cedex 9, France
[2] Ruhr-Universität Bochum, 150 Universitätsstraße, 44801 Bochum, Germany, 0234 32201
[3] Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, US
[4] Germany Research Center for Artificial Intelligence (DFKI), Stuhlsatzenhausweg 3, D-66123 Saarbruecken, Germany
pierre.tchounikine@imag.fr,
rummel@psychologie.uni-freiburg.de,
bmclaren@cs.cmu.edu

Abstract. In this chapter we discuss how recent advances in the field of Computer Supported Collaborative Learning (CSCL) have created the opportunity for new synergies between CSCL and ITS research. Three "hot" CSCL research topics are used as examples: analyzing individual's and group's interactions, providing students with adaptive intelligent support, and providing students with adaptive technological means.

22.1 Introduction

The field of Computer Supported Collaborative Learning (CSCL) focuses on how students learn by collaborating and how this collaboration can be supported by technology. Research in collaborative learning has shown, in general, that collaboration can increase group performance and individual learning outcomes. However, an educational setting with collaboration is not, on its own, sufficient for learning to occur (see Slavin 1996, for a review). CSCL research has shown that it is difficult to clearly define the interaction between the initial conditions of collaboration (e.g., the composition of the group or the type of task the group is engaged in) and learning outcomes (Dillenbourg et al. 1996). Moreover, collaboration leads to positive outcomes only when students engage in knowledge-generative interactions such as giving explanations, and engaging in argumentation, negotiation, conflict resolution or mutual regulation (Dillenbourg and Jermann 2007). At the same time, several potentially problematic issues must be avoided, such as unequal engagement or social loafing. Generally, it has been discovered that the occurrence of knowledge-generative interactions is not a given: such interactions do not necessarily emerge spontaneously (Cohen 1994; Salomon and Globerson 1989). Researchers attempting to understand how to

R. Nkambou et al. (Eds.): Advances in Intelligent Tutoring Systems, SCI 308, pp. 447–463.
springerlink.com

foster collaborative learning have thus focused on how best to promote fruitful interactions among collaborative learners.

A well studied and documented approach to supporting collaboration is scripting with *macro-scripts*. The purpose of CSCL macro-scripts and associated technology is to introduce structure and constraints that guide collaborative interactions among distant students or co-present students whose action or interaction is (at least partially) mediated by a computer-based system. A CSCL script typically describes the task to be achieved by students and defines how the task is to be divided into subtasks, the sequencing of these subtasks, the role of each student, sets constraints or rules for the interaction, and prescribes the features or tools of the computer-based system to be used by the students (Fischer et al. 2007; Dillenbourg and Tchounikine 2007; Kollar et al. 2006; Rummel and Spada 2005a and 2005b; Tsovaltzi et al. 2010). Implemented in this manner scripts, thus, provide predefined, fixed assistance to learners as they collaborate. Other support methods can similarly be classified as providing fixed assistance; for instance, giving students declarative instruction on how to collaborate before they collaborate (e.g., Saab et al. 2007) or providing students with examples of good collaboration (e.g., Rummel and Spada 2005b). Scripts have proven to be effective in promoting fruitful interactions and student learning, but the design of scripts follows a "razor's edge" between useful guidance and control of student activities: if the scaffolding they provide is too weak, it will not produce the expected interactions; if it is too strong or irrelevant, it can lead to sterile interactions (Dillenbourg 2002).

Research in CSCL scripts has initially focused on how to design settings (i.e., define the task, the script structure, the content of hints provided to the individuals or the group, or the technology provided to support students' actions) whose properties are likely to prompt students to engage in particular activities and interactions. A recent development in CSCL is to also take into account students' *effective enactment* of the script. A rationale for this is that students' enactment of the script is influenced by many parameters (individual differences, group phenomena) that lay outside of the control of the designer, and, consequently, the enactment may differ from expectations (Tchounikine 2008). In particular, students mostly focus on solving the task (and not on collaborating or interacting as hoped) and it may thus be required to adapt the setting or provide adapted feedback in order to enhance collaboration. Moreover, students may have different and dynamically changing needs and thus require individualized support (Diziol et al. 2010).

An important objective of current research is thus to dynamically adapt to the conditions of students' interactions as they unfold. Addressing this objective requires on-the-fly assessment of students' activities (how they interact, which steps they take at solving the task, how they use the technology) as a basis for adaptation decisions (McLaren et al., in press). It also requires modelling how the system should react, that is, in which way it should adapt its support to individual and collective needs. Intelligent Tutoring Systems (ITS) are traditionally designed to provide user adapted support. Leveraging ITS approaches could thus be a promising direction for achieving adaptive support for CSCL.

In the remainder of this chapter we will present research efforts that address issues related to adaptation and illustrate potential synergies between CSCL and ITS research:

Analyzing individual's and group's interactions (interaction analysis). In section 2 we explore how one can analyze student interactions as they communicate and collaborate with one another, as a basis for providing adaptive guidance and feedback.

Providing students with adaptive intelligent hints. In section 3 we explore how ITS technology and CSCL scripts can be combined to adaptively tailor support and feedback to students' needs.

Providing students with adaptive technological means. In section 4 we discuss why technological frameworks should be flexible and adapt to students' effective activity in order to continuously provide technological support that maintains the targeted pedagogical conditions. We outline the requirements for adaptive technological platforms.

22.2 Interaction Analysis

In CSCL, *interaction analysis* refers to analyzing what students are doing as they communicate and collaborate with one another, at a relatively fine-grain level. This is a basis for, in the context of CSCL scripts, adapting the script, adapting the technological framework or providing individual or collective hints. More generally, interaction analysis can be a basis to guide collaborative behavior in general, including within non-scripted settings.

As an example of an interaction analysis approach, past work of McLaren and colleagues (McLaren, et al. in press) involved developing techniques to analyze the interactions and communication of students as they debate, in an online environment, thorny ethical and social issues such as "Should experiments be performed on animals?" These collaborative e-discussions occur in the context of a shared graphical workspace, such as shown in Figure 22.1. Students make contributions by dragging and dropping shapes with different semantics (e.g. a "claim", "argument", "counter-argument"), filling the shapes with text containing their contributions to the discussion (e.g., "I don't agree with John's claim, because …"), and linking the shapes to the contribution of other students with labeled links, such as "opposes" or "supports."

The reason for interaction analysis in the system that McLaren et al. worked with – called ARGUNAUT – was to provide feedback to a teacher so that he or she can help students stay on topic, elicit contributions from all members of the collaboration groups, suggest the use of supported claims and arguments, and generally guide the students toward fruitful discussion and collaboration (Hever et al. 2007; De Groot et al. 2007). Since the burden of moderating multiple, simultaneous e-discussions – that is, supporting many groups of students at the same time – is too difficult for individual teachers, ARGUNAUT attempts to summarize

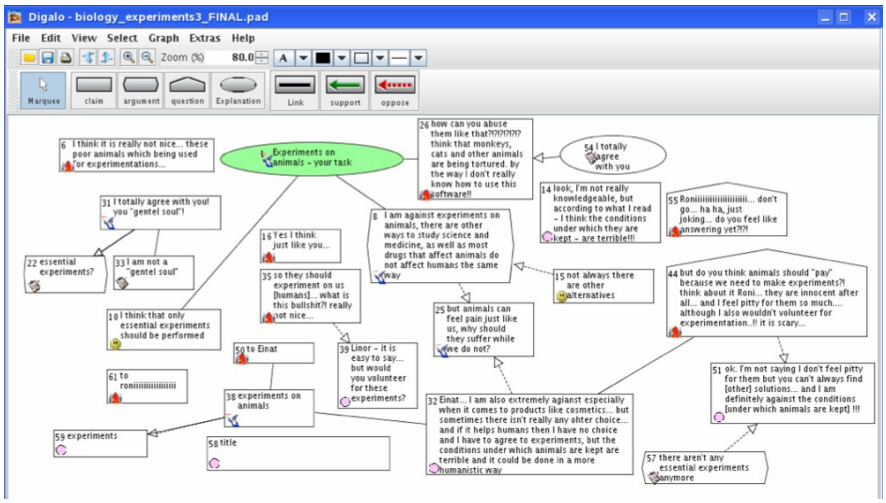

Fig. 22.1 An e-discussion in ARGUNAUT

the students' discussions and alert teachers to critical aspects and events in the e-discussions. In other educational settings, this type of analysis could be the basis for automated feedback to students or script adaptation.

The ARGUNAUT system provides the teachers with online, automated feedback regarding important aspects and characteristics of each discussion, explicitly focusing attention on events or situations that may require the teacher's intervention or support. The key idea is to analyze student contributions and e-discussions using machine learning (Witten and Frank 2005), shallow text processing (Rosé et al. 2008), and case-based graph matching (McLaren 2003). ARGUNAUT leverages (a) the structure of the argument graphs, (b) the textual contributions of the students, and (c) the temporal sequence of those contributions. It also analyzes and classifies, using machine-learned classifiers, student interactions at three levels: (1) the shape-level (i.e., individual contributions), (2) the paired-shape level (i.e., contributions by two students, linked together in the graphical representation), and (3) the cluster level (i.e., linked groups of contributions, consisting of 2 or more shapes). The classifiers that are created at each of these levels help to draw attention to activities of students that are positive, such as responding to one another's arguments, or negative, such as going off topic. Through extensive experimentation, six of the shape and paired-shape classifiers achieved at least satisfactory results (i.e., Kappa values above 0.6), while five of the cluster-level classifiers have achieved such results (for details, see McLaren, et al. in press and Wegerif et al. 2009). An example of the teacher's view of the results of one of the cluster classifiers – Argument + Evaluation – is shown in Figure 22.2.

In the following we review other projects that use machine-learning, language processing or data mining techniques.

Rosé and colleagues have done similar interaction analysis research to the work described above; in fact, they developed the text analysis tool, TagHelper (Rosé et al. 2008), also used in the ARGUNAUT. They analyzed a corpus of 1,250 coded

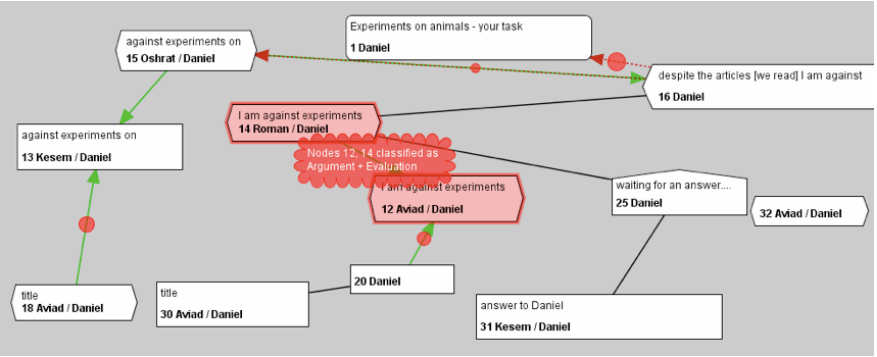

Fig. 22.2 Argument graph display of Argument + Evaluation clusters, with one matching cluster highlighted

text segments along multiple dimensions of argumentation to derive machine-learned classifiers, some of which achieved acceptable Kappa values of 0.7. In their work, Rosé and colleagues originally intended to assist coders in analyzing protocols after the fact; but, more recently, they have focused on providing real-time support to dyads collaborating on a problem-solving task (Kumar et al. 2007). Similar to the ARGUNAUT system, they perform online analysis of textual communication data, in their case, chat data. In contrast to ARGUNAUT, but more like traditional ITS, their analysis results are displayed directly to students, rather than to teachers. An empirical study (Kumar et al. 2007) showed significant learning benefits in terms of analytical knowledge and conceptual understanding when the adaptive support was provided.

Goodman and colleagues (2005) have also applied a machine-learning approach to the problem of interaction analysis. Within the EPSILON system peer groups work together on a problem in the domain of object modeling techniques (OMT). Their collaboration takes place within a shared whiteboard in which diagrams (e.g. class diagrams) are constructed. Peers communicate via a text chat with a sentence opener interface (an interface in which the beginning of sentences are provided as options to the student, e.g., "We have learned …"); an agenda tool supports task management. The system evaluates aspects concerning domain (e.g., domain knowledge of peers), task (e.g., progress in solving the task) and possible problems in the collaboration process (e.g. unanswered questions). The sentence opener interface plays a critical role in interaction analysis; it is used to automatically classify each chat contribution as a dialogue act. The dialogue acts, in turn, are used as features for machine-learning analyses, bypassing the complicated task of natural language (or text) processing tackled by McLaren and colleagues and Rosé and colleagues. The Goodman et al. (2005) analysis approach is similar to ITS: Relevant results are displayed immediately to the peers via meters, while feedback is provided by means of an artificial peer agent that verbally interacts with the participants.

Soller's work, also in the context of EPSILON, was concerned with the analysis of chat conversations that accompanied activities in a problem-solving

environment (Soller 2004). The chat tool was enhanced by a sentence-opener interface to structure users' communication (the same one used by Goodman et al.) and to make interaction analysis feasible. The analysis of EPSILON aimed at identifying episodes in which students communicated their knowledge to their peers ("knowledge sharing episodes") and episodes in which they failed to do so ("knowledge sharing breakdowns") using annotated data provided to Hidden Markov Models (HMMs). In a second step, she investigated reasons for knowledge sharing breakdowns using multidimensional scaling (MDS) and clustering techniques. In this way Soller identified different patterns (clusters) of successful knowledge sharing and knowledge sharing breakdowns. Soller's approach analyzes *sequences* of actions, as opposed to sub-graphs within an argument map that are sometimes sequential, sometimes parallel (such as in the ARGUNAUT system described above). The *textual* content of contributions is not analyzed in Soller's system; it relies exclusively on dialogue acts, which are trivially inferred from the selected sentence openers.

Another approach to interaction analysis is that of Ravi and Kim (2007) who analyzed threads in a technical discussion board in order to call an instructor's attention to discussions that contain unanswered questions. They developed two machine-learned classifiers (linear SVMs), one for detecting questions (QC), the second for detecting answers (AC), and achieved accuracies of 88% (QC) and 73% (AC). They also employed shallow language features, similar to the McLaren et al and Rosé et al. approaches, although somewhat more elaborated ones (e.g., tri- and quadro-grams). Furthermore they implemented a rule-based thread profiler that assigns one of four typical profiles to threads, with accuracies varying between 70% and 93%. In follow-up work, Kim and colleagues (2008) describe PedaBot, a threaded discussion system that scaffolds students' discussions by retrieving messages from past technical discussions (e.g., Operating Systems) that are possibly relevant to the current context. The work by Kim and colleagues does not specifically take into account structure and temporal features to support its classification approach, as, for instance, the ARGUNAUT system does.

Jeong (2005) has developed a software tool called the Discussion Analysis Tool (DAT) that uses sequential analysis to capture and model sequences of speech acts. DAT models an online threaded conversation as a network of transitional probabilities, called a transitional state diagram, building the diagram from prelabeled data. For example, in a particular diagram a "challenge" act might occur with a probability of 0.52 after an "argument" is made, while an "explanation" follows an "argument" with a (surprising) probability of 0.08. DAT has been used to, for instance, evaluate the interactions that are most likely to promote critical thinking and the effects of supportive language (e.g., "I agree," or "ask" questions) on subsequent group interactions. DAT can also be used to evaluate whether threaded conversations deviate from a norm; it creates a z-score matrix to show probabilities that were significantly higher or lower than expected in one state diagram compared to another. But Jeong's tool is intended more as a post-hoc analysis tool, rather than a tool for supporting online and "live" analysis, as in a traditional ITS system. Also, Jeong's system does no language analysis; it depends on human post-hoc coding (or real-time labeling) to identify the individual acts.

Other interaction analysis work has focused on identifying frequently occurring patterns. For instance, sequential pattern mining algorithms (Agrawal and Srikant 1995) has been used to try to find salient patterns of student collaboration. Such algorithms have, for instance, been used to account for both language and contextual attributes – to a reasonable degree of accuracy – in classifying email messages (Carvalho and Cohen 2005). Kay et al. (2006) used a variant of the Generalized Sequential Pattern Algorithm (Srikant and Agrawal 1996) to identify common interaction patterns in a source repository and Wiki log data from student software development projects.

The pattern that emerges from the interaction analysis research within CSCL is a movement away from computer tools used almost exclusively to facilitate collaborative communication to computer tools used to *analyze* that communication. A key difference to ITS work is that the nexus of analysis in CSCL systems is the interaction between the collaborating students, rather than between software tutor and human tutor. Furthermore, the communication between collaborators is typically much more varied and unpredictable than the interaction between a software tutor (which is typically consistent and somewhat predictable) and a human tutee. A similarity to ITS work, on the other hand, is that student actions in CSCL systems are now increasingly analyzed for the purpose of providing feedback, guidance, and scaffolding for learning.

22.3 Providing Adaptive Intelligent Hints

Providing adaptive intelligent hints consists of, first, monitoring and evaluating students' collaboration (on the basis of run-time data, similar to interaction analysis, as described in the previous section) and, second, automatically responding to the particular needs of the collaborating participants with context-specific hints and/or scaffolding.

A typical way that student interactions have been assessed is to compare the difference between those interactions and a model of optimal collaboration (a kind of model tracing approach) or by checking constraints (constraint-based approach).

For instance, one way of assessing the quality of student interactions is by tracking student dialogue patterns. This is commonly accomplished by asking students to indicate the type of contribution that they would like to make before they make it. For example, students may select a sentence starter like "We need to work together on this..." to begin their utterance, an approach used in several projects described earlier (Goodman et al. 2005; Soller 2004). Based on the starters that students select, the system can make inferences about what students are saying and use these inferences to provide feedback (Tedesco 2003). However, as students often do not accurately label their utterances, the inferences that the system makes can be inaccurate. Thus, automated dialogue assessment solutions have been developed. So far, this technology has only been used successfully in limited ways, such as classifying the topic of conversation (Kumar et al. 2007), characterizing general argumentation patterns (McLaren et al, in press; discussed above), or assessing student accuracy when they use sentence starters (Israel and Aiken

2007). Some researchers try to circumvent the problems of assessing dialogue by relying on simple metrics like participation to trigger feedback. For instance, these systems evaluate the amount or length of contributions collaborators make to a shared workspace or to a dialogue and support the interaction by directly encouraging non-contributors to participate more (Constantino-Gonzalez et al. 2003). Unfortunately, the same assessment metrics cannot be used to give students feedback on how to participate, which may ultimately be more valuable. Some other works propose to facilitate tutor's perception of collaboration issues such as student's organization breakdowns by proposing students with specifically design organization tools, whose usage makes learners' organization more easily detectable and analyzable (Moguel et al. 2010).

Another approach to providing adaptive support for collaborative learning may be more promising – and leverages ideas from ITS. In individual learning settings, adaptive ITS feedback has been shown to be quite successful at improving student learning in a variety of domains such as physics (VanLehn et al. 2005), mathematics (Koedinger et al. 1997) and reading (Beck et al. 2004). Further, it has been argued that using existing ITS problem-solving models for individual learning to provide interaction support in collaborative learning settings may be fruitful (Walker et al. 2009b). Some adaptive collaboration systems already partially capitalize on the ITS approach. For example, when students submit a group solution in COLLECT-UML (Baghaei et al. 2007), the system evaluates the solution using a constraint-based model, and provides feedback on the quality of the solution. Occasionally, this problem-solving support even leverages student talk rather than only student action: When CycleTalk (Kumar et al. 2007) detects problem-relevant topics in student conversation, it engages the collaborating students in a tutorial dialogue, asking them to answer questions that concern these aspects. This tutorial dialogue often yields increased interaction between the collaboration partners. Both COLLECT-UML and CycleTalk implemented support in a collaborative setting by extending an existing *individual* learning system.

In two other projects, the problem-solving models of an existing intelligent tutoring system were used to provide *interaction support* (Diziol et al. 2010). Here collaborative extensions to the Cognitive Tutor Algebra (CTA), a widely-used tutoring system for mathematics instruction on the high-school level (Koedinger et al. 1997) were developed. The CTA covers different aspects of algebra learning such as linear equations and inequalities. To provide adaptive tutoring, the CTA evaluates the student's problem-solving actions by comparing them to a cognitive model of successful student performance, represented using a set of production rules. If an error is detected, the CTA immediately marks it as incorrect and provides context-sensitive feedback. In one project (Rummel et al. 2010), two students worked on the same computer, and the CTA had a joint model of the dyad's problem-solving. The output of this problem-solving model served as input for an interaction model that assessed students' learning behaviour. If ineffective learning strategies, such as trial and error and hint abuse behaviours, were detected, this triggered adaptive support prompting fruitful collaboration. In a second project (Walker et al. 2009b) a peer tutoring scenario was involved: one student tutored another student, while the students worked on separate computers. The system

integrated the CTA problem-solving model with a model of good tutoring in order to provide the peer tutor with support.

There are several other design options for providing interaction support along the lines of ITS. Instead of modelling either one student (the peer tutor) or two students together, another option is to model the interaction partners separately. For instance, the COMET system developed by Suebnukarn and Haddawy (2006) assesses the individual expertise of participants and encourages the collaborators to share their complementary knowledge with others. Similarly, a system for adaptive collaboration support could intervene if it detects undesirable asymmetries in students' skill acquisition: if a particularly difficult problem-solving step is mainly solved by one interaction partner, this might indicate that the other student has not yet acquired the relevant skills and could benefit from adaptive collaboration support.

Furthermore, as argued in (Walker et al. 2009a), leveraging existing problem-solving models can facilitate the comparison of different types of adaptive collaboration support, and thus help to gain information on the conditions of optimal assistance. While the evaluation of the two adaptive collaborative extensions to the CTA revealed an impact on student interaction, the improved interaction did not yield the differences in student learning outcome that have been found in other studies (e.g. Baghaei et al. 2007). This indicates that the need for further research on how to optimize collaboration support for particular interaction conditions. This optimization can be considered an instantiation of a more general assistance dilemma (Koedinger and Aleven 2007), where in order to discover how to best deliver assistance, one must manipulate the amount, type and timing of help provided to students. For example, we so far only have limited knowledge on how to time support most effectively. It is still an open question whether it is best to provide adaptive collaboration support immediately, or whether it might sometimes be beneficial to withhold it. Mathan and Koedinger (2005) investigated this research question in an individual learning setting, and discovered that the two timing options served different goals. Immediate feedback ensured that students did not get stuck in problem-solving and thus was more immediately effective and efficient. Delayed feedback enabled students to practice their monitoring skills and consequently yielded improved learning transfer. Similarly, immediate feedback to collaboration may improve the current interaction, while delayed feedback may increase students' collaboration skills and thus promote future interactions (Kapur 2008). Thus, the type of feedback may have to be adapted to the goal of the instruction. On a practical level, the accelerated development of adaptive collaboration support conditions based on existing intelligent tutoring technology may help us to increase our knowledge of optimal collaboration assistance, as it enables us to more rapidly implement and compare different design options concerning the amount, type and timing of support.

In summary, while there are preliminary promising results that hint toward the effectiveness of adaptive collaboration support, clear conditions and guidelines have not yet emerged on how to best deliver adaptive assistance. Leveraging existing problem-solving models can facilitate the implementation of adaptive collaboration support, but this work is still in its early stages. A clear next step is to investigate different types of adaptive collaboration support in more detail to increase our knowledge of when and why adaptive collaboration support is effective.

22.4 Providing Students with Adaptive Technological Means

Providing adaptive technological means consists of, on the basis of monitoring and evaluating students' activity (cf. preceding sections), adapting accordingly the technological means they are presented with.

With respect to CSCL support, the role of the computer-based system is two-fold: (1) provide the necessary technological means and (2) participate to the scripting objective, i.e., supporting and constraining.

Let us consider a setting where students must collaboratively build a graphical model of some physics phenomenon, and anchor its design in explicit argumentation. First, the role of the computer is to provide the *technological means* for students' activities. For instance, students can be proposed with a chat to communicate synchronously and a shared whiteboard to collaboratively edit their model. Second, the computer might be used as a way to provide *support and constraints*. Proposing students with a basic chat allows argumentation to occur, but does not foster it. Differently, proposing students with a dedicated e-discussion tool such as ARGUNAUT (cf. section 2) introduces some support (and constraints). Similarly, a whiteboard allows students to build a model. A dedicated modeling tool introducing specific physics notions and relations (as ARGUNAUT introduces specific communication constructs) will here again introduce some support (and constraints).

More generally, the computer-based system can participate in structuring and constraining the sequences of activities or the way students engage in individual and collective activities by specifying roles, introducing specific dataflow or workflow, specific tools (e.g., modeling tools proposing carefully defined epistemic primitives) or communication functionality that impact students' interaction (e.g., imposing sentence-openers or turn-taking structures). As a matter of fact, technology is not neutral: although not necessarily explicit, a given design always denotes some usage expectations, and influences users' activity.

Seen from the perspective of usage, macro-scripts create socio-technical settings. Technology impacts the script enactment, but this impact is not necessarily the one that is expected, in particular because of the uncertainties of how students will perceive and use the technology. Thus, it is important to take into consideration not only the script and the technology as considered by designers, but also the phenomena related to the effective use of technology.

A general difficulty with designing technology to support human activity is that designers have limited control over how their designs will be enacted. Goodyear (2001) emphasizes the fact that teachers set tasks and students interpret the specifications of the task. Their subsequent activity is related to their interpretation of the task (which may change from student to student and from the teacher's wills), and also to other dimensions (e.g., students' motivations or perception of the setting) that evolve over time, and are interrelated within systemic relations. Students' perception and enactment of CSCL scripts and their use of the provided technological means are intrinsically situated. The students' activity that will emerge from the confrontation of the students with the task and the technological setting is thus subject to different contingences. As a consequence, its details are unpredictable (Tchounikine 2008).

Macro-scripts, as a particular kind of pedagogical method, are also intrinsically related to open issues that cannot be fully defined or predicted. It is not possible to exhaustively list and consider all of the pedagogical parameters of a macro-script situation. As a consequence, when monitoring the script as it is enacted by the students, the teacher often has to manage unexpected events (originating from inside or outside the script), manage requests from the students that will lead him/her to consider the script's or technology's modifications, or use a pedagogical opportunity that appears (Dillenbourg and Tchounikine 2007). For example, teachers may want or need to modify, at run-time, decisions taken when tuning the script: change the groups because a student drops out of the course or because two conflicting leaders emerge from a group and this becomes problematic; postpone some deadlines in order to deal with external or internal reasons (network failure, bad appreciation of the task difficulty, etc.); change the script structure (change the order of phases, add or remove a phase, merge some tasks, change the argumentation tool because students face problems with it); etc. (Dillenbourg and Tchounikine 2007). Here again, this creates uncertainties related to macro-scripts' enactment, to be taken into account when studying the technological dimensions.

By definition, macro-scripts provide learners with some flexibility, i.e., let them decide on their own part of how they will follow the script. Many experiments reported in the literature show that learners use this flexibility, e.g., in context, decompose tasks into subtasks and refine their division of labor, adopt sub-strategies or alternative ways of using the technological means they have been offered: they involve self-organization activities: part of the organization is externally set by the script, and part is related to emergent features of learners' enactment of the script at run-time (Tchounikine 2007).

Phenomena related to the perception by students of the task and the technological setting, and their use of technology, is a general issue of educational technology. It is however of particular importance in open or semi-open settings in CSCL within which learning is supposed to originate from student's process, interactions and initiatives: the focus is not on the task output, but on the process and its characteristics.

From the point of view of computer-based system design, the fact that scripts enactment is difficult to anticipate can be an argument in favor of keeping the technological support very generic. This is the principle underlying platforms such as generic learning management systems. Students are presented with generic platforms that offer students a list of autonomous technological means (chat, forum, whiteboard, etc.) they can choose to use as they want. These means are not thought to as a *milieu* proposing resources and means that have been designed and articulated according to the details of the students' hypothesized activity. Although largely adopted (if anything else, because of its simplicity), this approach does not address the issue of using the technology as a way to provide support and/or constraints in line with the objectives of the script and the expected students' knowledge-construction processes. It also makes it difficult to provide precise scaffolding.

Recent CSCL works has attempted to understand how to design and implement platforms addressing the following system of constraints: (1) provide students

with the functionality necessary to achieve the tasks proposed by the script, (2) participate in the objective of structuring students' collaboration by reifying constraints/support related to the script's pedagogical objective, and (3) be sufficiently adaptive to be coherent with students activity if the actual interaction pattern differs from expectations, or if some unpredictable events arise. The system must be sufficiently flexible not to over-constrain students' activity whilst keeping the script's *raison d'être* and remaining coherent with the pedagogical objectives.

In order to both (1) orchestrate activities and manage the workflow and (2) be able to adapt at run-time to requests from students or teachers whilst adhering to the learning objectives constraints, a powerful evolution for CSCL is to address the technological setting as a *script engine*: the technological settings (tools, interfaces, etc.) must be the result of careful *a priori* decisions (when the script is launched) and then run-time decisions, during its enactment. Figure 22.3 (Tchounikine 2008) presents a general theoretical architecture of such an (intelligent) flexible adaptive activity framework.

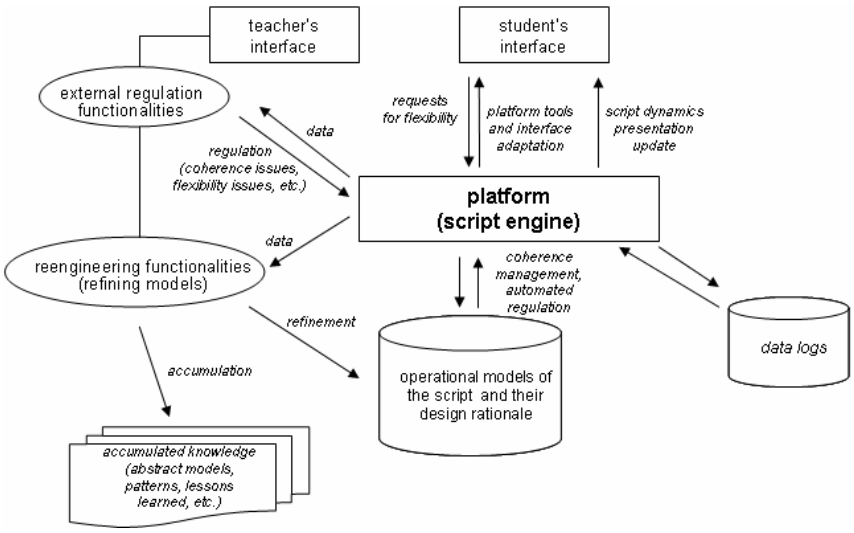

Fig. 22.3 A general theoretical architecture

Putting into practice this approach requires computer-tractable models of the script. Different efforts have been recently dedicated to this such as defining scripts' general components and mechanisms (Kobbe et al. 2007) and exploring different modeling languages that can be used to represent such issues (finite automata, statecharts, activity diagrams, Petri nets, etc.; see Harrer and Malzahn (2006) for a review). As an example of how flexibility can be introduced, Haake and Pfister (2007) propose to describe scripts (roles, possible sequences of actions, etc.) as a finite state automaton, a formalism that allows complex control structures. The script can then be deployed on a platform that is compliant with this formalism. Within such an approach, the platform runs the script and provides

access to functionalities or data according to the automata. The script can be modified at any time via its specification, without requiring any hand-modification of the platform, which provides a certain form of flexibility.

Adapting the script or the platform at run-time requires modeling not only the script structure (phases, division of work, etc.) but also the underlying design rationale. For this purpose, Dillenbourg and Tchounikine (2007) propose to dissociate intrinsic constraints (script's core mechanisms, which set up the limits of flexibility, i.e., what cannot be accepted in order for the script to keep its raison d'être) and extrinsic constraints (constraints bound to contextual factors, which define the space for flexibility, i.e., the space within which a script should be modifiable by teachers and/or students because the related decisions result from arbitrary or practical choices).

Another important dimension of such architecture is to support accumulation of knowledge by, for instance, supporting data analysis and recurrent-patterns identification, which will help in iteratively refining scripts, and in progressing in the understanding of script enactment. Here again, Machine Learning techniques appear promising.

Implementing such an innovative approach to CSCL technological settings thus pushes one to consider issues that are similar to those tackled in ITS: creating operational languages that denote the different models and systems of constraints; understanding students' activity by interpreting data and logs (cf. section 2), and dynamically (intelligently) reacting to adapt the script and/or the technological framework or to scaffold individual and groups (cf. section 3).

22.5 Discussion

Historically, in CSCL systems, the computer has been used as a mediating mechanism between learners and teachers or other learners. Whereas Intelligent Tutoring Systems address issues such as the analysis and understanding of learners' activity and production, problem solving or interaction control, classical CSCL systems have not addressed these issues at all. Whereas ITS research has, since its inception, leveraged Artificial Intelligence techniques, CSCL research has instead focused on HCI issues related to providing students with a good experience of communicating with their fellow students (Tchounikine et al. 2009). However, times have changed. In this chapter, we have shown that at least some current CSCL research is exploring issues of adaptivity, automated analysis, and feedback. These changes bring the field of CSCL closer to techniques of ITS research.

In this chapter we have disentangled interaction analysis, providing students with adaptive hints and adapting the technological framework. This is indeed an analytical presentation. Interaction analysis is a basis for adaptivity in general. Adapting the support provided by individual or collective hints or provided by the technology serves the same objective (maintaining positive conditions that enhance interactions) and may be powerfully connected. Moreover, the examples we have raised are but approaches. For instance, some other works address adaptivity by identifying useful *adaptation patterns* to be embedded in systems for adaptive collaboration scripting, i.e., adaptation processes that can be initiated by

the system when specific conditions are identified during script enactment (Karakostas and Demetriadis 2009).

One of the reasons of CSCL rapid development in basic settings (i.e., out of research experiences) is the fact many CSCL projects are based on simple, stable, well disseminated, and almost freely available technologies. CS difficulties raised by CSCL (e.g., HCI issues) are less binary and non-contingent problems than ITS issues such as AI issues related to building learners' models or solving problems. In fact, historically, CSCL research focused on education-psychology issues rather than on CS support and, in particular, adaptive support. The CS dimension was often limited to communication devices and simple interfaces. Advanced technologies can however powerfully support and enhance communication and collaboration. Addressing adaptivity will indeed conduct to face difficult issues as ITS does since its early ages. This is a promising perspective for research.

Acknoledgements. The content of this article is based on the authors' works with colleagues and doctoral students. In particular, without the support and contributions of Dejana Diziol, Erin Walker, Oliver Scheuer and Ken Koedinger this chapter would not have been possible.

References

Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the 11th International Conference on Data Engineering (ICDE 1995), Taipei, Taiwan, pp. 3–14 (1995)

Baghaei, N., Mitrovic, T., Irwin, W.: Supporting collaborative learning and problem solving in a constraint-based CSCL environment for UML class diagrams. International Journal of Computer- Supported Collaborative Learning 2(2-3), 159–190 (2007)

Beck, J.E., Mostow, J., Bey, J.: Can automated questions scaffold children's reading comprehension? In: Lester, J.C., Vicari, R.M., Paraguacu, F. (eds.) ITS 2004. LNCS, vol. 3220, pp. 478–490. Springer, Heidelberg (2004)

Carvalho, V.R., Cohen, W.W.: On the collective classification of email "speech acts". In: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 345–352. ACM Press, New York (2005)

Cohen, E.G.: Restructuring the Classroom: Conditions for Productive Small Groups. Review of Educational Research 64(1), 1–35 (1994)

Constantino-Gonzalez, M.A., Suthers, D., Escamilla de los Santos, J.: Coaching web-based collaborative learning based on problem solution differences and participation. Artificial Intelligence in Education 13(2-4), 263–299 (2003)

De Groot, R., Drachman, R., Hever, R., Schwarz, B.B., Hoppe, U., Harrer, A., De Laat, M., Wegerif, R., McLaren, B.M., Baurens, B.: Computer Supported Moderation of E-Discussions: the ARGUNAUT Approach. In: Proceedings of the Conference on Computer Supported Collaborative Learning (CSCL 2007), vol. 8, pp. 165–167 (2007)

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

Dillenbourg, P., Jermann, J.: Designing integrative scripts. In: Scripting Computer-Supported Collaborative Learning - Cognitive, Computational, and Educational Perspectives. Computer-Supported Collaborative Learning Series, pp. 275–301. Springer, Heidelberg (2007)

Dillenbourg, P., Tchounikine, P.: Flexibility in macro-scripts for CSCL. Journal of Computer Assisted Learning 23(1), 1–13 (2007)

Dillenbourg, P., Baker, H.P.M., Blaye, A., O'Malley, C.: The Evolution of Research on Collaborative Learning. In: Learning in Humans and Machines: Towards an Interdisciplinary Learning Science. Elsevier/Pergamon, Oxford (1996)

Diziol, D., Walker, E., Rummel, N., Koedinger, K.: Using intelligent tutor technology to implement adaptive support for student collaboration. Educational Psychology Review (2010), doi:10.1007/s10648-009-9116-9

Fischer, F., Kollar, I., Mandl, H., Haake, J.M. (eds.): Scripting computer-supported communication of knowledge – cognitive, computational, and educational perspectives. Springer, Heidelberg

Goodman, B., Linton, F., Gaimari, R., Hitzeman, J., Ross, H., Zarrella, G.: Using Dialogue Features to Predict Trouble During Collaborative Learning. User Modeling and User-Adapted Interaction 15, 85–134 (2005)

Goodyear, P.: Effective networked learning in higher education: notes and guidelines. Final Report to JCALT: Networked Learning in Higher Education Project (3) (2001), http://csalt.lancs.ac.uk/jisc/ (Retrieved from December 2, 2006)

Haake, J., Pfister, H.-R.: Flexible scripting in net-based learning groups. In: Fischer, F., Kollar, I., Mandl, H., Haake, J.M. (eds.) Scripting computer-supported cooperative learning. Cognitive, computational, and educational perspectives. Springer, Heidelberg (2007)

Harrer, A., Malzahn, N.: Bridging the gap –towards a graphical modelling language for learning designs and collaboration scripts of various granularities. In: International Conference on Advanced Learning Technologies, Kerkrade, The Netherlands, pp. 296–300 (2006)

Hever, R., De Groot, R., De Laat, M., Harrer, A., Hoppe, U., McLaren, B.M., Scheuer, O.: Combining Structural, Process-oriented and Textual Elements to Generate Alerts for Graphical E-Discussions. In: Proceedings of the Conference on Computer Supported Collaborative Learning (CSCL 2007), vol. 8, pp. 286–288 (2007)

Israel, J., Aiken, R.: Supporting collaborative learning with an intelligent web-based system. International Journal of Artificial Intelligence and Education 17(1), 3–40 (2007)

Jeong, A.: A Guide to Analyzing Message-Response Sequences and Group Interaction Patterns in Computer-Mediated Communication. Distance Education 26(3), 367–383 (2005)

Kapur, M.: Productive failure. Cognition and Instruction 26(3), 379–424 (2008)

Karakostas, A., Demetriadis, S.: Adaptation patterns in systems for scripted collaboration. In: O'Malley, C., Suthers, D., Reimann, P., Dimitracopoulou, A. (eds.) Proceedings of the 9th international Conference on Computer Supported Collaborative Learning. Computer Support for Collaborative Learning. International Society of the Learning Sciences, vol. 1, pp. 477–481 (2009)

Kay, J., Maisonneuve, N., Yacef, K., Zaïane, O.: Mining Patterns of Events in Students' Teamwork Data. In: Proceedings of the Workshop on Educational Data Mining at the 8th International Conference on Intelligent Tutoring Systems, pp. 45–52 (2006)

Kim, J., Shaw, E., Ravi, S., Tavano, E., Arromratana, A., Sarda, P.: Scaffolding On-Line Discussions with Past Discussions: An Analysis and Pilot Study of PedaBot. In: Woolf, B.P., Aïmeur, E., Nkambou, R., Lajoie, S. (eds.) ITS 2008. LNCS, vol. 5091, pp. 343–352. Springer, Heidelberg (2008)

Kobbe, L., Weinberger, A., Dillenbourg, P., Harrer, A., Hämäläinen, R., Häkkinen, P., Fischer, F.: Specifying Computer-Supported Collaboration Scripts. International Journal of Computer-Supported Collaborative Learning 2(2-3), 211–224 (2007)

Koedinger, K.R., Aleven, V.: Exploring the assistance dilemma in experiments with Cognitive Tutors. Educational Psychology Review 19(3), 239–264 (2007)

Koedinger, K.R., Anderson, J.R., Hadley, W.H., Mark, M.A.: Intelligent tutoring goes to school in the big city. International Journal of Artificial Intelligence in Education 8, 30–43 (1997)

Kollar, I., Fischer, F., Hesse, F.W.: Computer-supported collaboration scripts—a conceptual analysis. Educational Review 18(2), 159–185 (2006)

Kumar, R., Rosé, C.P., Wang, Y.C., Joshi, M., Robinson, A.: Tutorial dialogue as adaptive collaborative learning support. In: Luckin, R., Koedinger, K.R., Greer, J. (eds.) Proceedings of Artificial Intelligence in Education, pp. 383–390. IOS Press, Amsterdam (2007)

Mathan, S.A., Koedinger, K.R.: Fostering the Intelligent Novice: Learning from errors with metacognitive tutoring. Educational Psychologist 40(4), 257–265 (2005)

McLaren, B.M.: Extensionally Defining Principles and Cases in Ethics: An AI Model. Artificial Intelligence 150, 145–181 (2003)

McLaren, B.M., Scheuer, O., Mikšátko, J.: Supporting Collaborative Learning and e-Discussions Using Artificial Intelligence Techniques. International Journal of Artificial Intelligence in Education, IJAIED (in press)

Moguel, P., Tchounikine, P., Tricot, A.: Supporting Learners' Self-Organization: an Exploratory Study. To appear in the Proceeding of the International Conference on Intelligent Tutoring Systems ITS 2010 (2010)

Ravi, S., Kim, J.: Profiling Student Interactions in Threaded Discussions with Speech Act Classifiers. In: Luckin, R., Koedinger, K.R., Greer, J. (eds.) Proceedings of the 13th International Conference on Artificial Intelligence in Education, AIED 2007, pp. 357–364. IOS Press, Amsterdam (2007)

Rosé, C., Wang, Y.-C., Cui, Y., Arguello, J., Stegmann, K., Weinberger, A., Fischer, F.: Analyzing Collaborative Learning Processes Automatically: Exploiting the Advances of Computational Linguistics in CSCL. International Journal of Computer-Supported Collaborative Learning (IJCSCL) 3(3) (2008)

Rummel, N., Spada, H.: Instructional support for collaboration in desktop videoconference settings: How it can be achieved and assessed. In: Bromme, R., Hesse, F.W., Spada, H. (eds.) Barriers and biases in computer-mediated knowledge communication and how they may be overcome, pp. 59–88. Springer, New York (2005a)

Rummel, N., Spada, H.: Learning to collaborate: An instructional approach to promoting collaborative problem-solving in computer-mediated settings. Journal of the Learning Sciences 14(2), 201–241 (2005b)

Rummel, N., Diziol, D., Spada, H.: Collaborative learning with the Cognitive Tutor Algebra. An experimental classroom study (2010) (Manuscript under revision)

Saab, N., van Joolingen, W.R., van Hout-Wolters, B.H.A.M.: Supporting communication in a collaborative discovery learning environment: The effect of instruction. Instructional Science 35(1), 73–98 (2007)

Salomon, G., Globerson, T.: When Teams do Not Function the Way They Ought To. International Journal of Educational Research 13, 89–100 (1989)

Slavin, R.E.: Research on cooperative learning and achievement: What we know, what we need to know. Contemporary Educational Psychology 21(1), 43–69 (1996)

Soller, A.: Computational Modeling and Analysis of Knowledge Sharing in Collaborative Distance Learning. User Modeling and User-Adapted Interaction: The Journal of Personalization Research 14, 351–381 (2004)

Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In: Apers, P.M.G., Bouzeghoub, M., Gardarin, G. (eds.) EDBT 1996. LNCS, vol. 1057, pp. 3–17. Springer, Heidelberg (1996)

Suebnukarn, S., Haddawy, P.: Modeling individual and collaborative problem-solving in medical problem-based learning. User Modeling and User-Adapted Interaction 16(3-4), 211–248 (2006)

Tchounikine, P.: Directions to acknowledge learners' self organization in CSCL macro-scripts. In: Haake, J.M., Ochoa, S.F., Cechich, A. (eds.) CRIWG 2007. LNCS, vol. 4715, pp. 247–254. Springer, Heidelberg (2007)

Tchounikine, P.: Operationalizing macro-scripts in CSCL technological settings. International Journal of Computer-Supported Collaborative Learning 3(2), 193–233 (2008)

Tchounikine, P., Mørch, A.I., Bannon, L.: A computer science perspective on TEL research. In: Balacheff, N., Ludvigsen, S., de Jong, T., Lazonder, A., Barnes, S. (eds.) Technology-Enhanced Learning – Principles and Product. Springer, Heidelberg (2009)

Tedesco, P.: MArCo: Building an artificial conflict mediator to support group planning interactions. International Journal of Artificial Intelligence in Education 13, 117–155 (2003)

Tsovaltzi, D., Rummel, N., McLaren, B.M., Pinkwart, N., Scheuer, O., Harrer, A., Braun, I.: Extending a virtual chemistry laboratory with a collaboration script to promote conceptual learning. International Journal of Technology Enhanced Learning 2(1-2), 91–110 (2010)

VanLehn, K., Lynch, C., Schulze, K., Shapiro, J.A., Shelby, R., Taylor, L., et al.: The Andes physics tutoring system: Lessons learned. International Journal of Artificial Intelligence and Education 15(3), 147–204 (2005)

Walker, E., Rummel, N., Koedinger, K.: CTRL: A research framework for providing adaptive collaborative learning support. User Modeling and User-Adapted Interaction: The Journal of Personalization Research (UMUAI) 19(5), 387–431 (2009a)

Walker, E., Rummel, N., Koedinger, K.: Integrating collaboration and intelligent tutoring data in evaluation of a reciprocal peer tutoring environment. Research and Practice in Technology Enhanced Learning 4(3), 221–251 (2009b)

Wegerif, R., McLaren, B.M., Chamrada, M., Scheuer, O., Mansour, N., Mikšátko, J., Williams, M.: Exploring Creative Thinking in Graphically Mediated Synchronous Dialogues. Computers & Education (2009) doi:10.1016/j.compedu.2009.10.015

Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)