

## InstanceCollage: A Tool for the Particularization of Collaborative IMS-LD Scripts

Eloy D. Villasclaras-Fernández<sup>1</sup>, Julio A. Hernández-Gonzalo<sup>1</sup>, Davinia Hernández-Leo<sup>2</sup>,  
Juan I. Asensio-Pérez<sup>1</sup>, Yannis Dimitriadis<sup>1</sup>, Alejandra Martínez-Monés<sup>1</sup>

<sup>1</sup>GSIC-EMIC Group, University of Valladolid, Spain // evilfer@ulises.tel.uva.es // jhergon@ulises.tel.uva.es //  
juaase@tel.uva.es // yannis@tel.uva.es // amartine@infor.uva.es

<sup>2</sup>University Pompeu Fabra, Spain // davinia.hernandez@upf.edu

### ABSTRACT

Current research work in e-learning and more specifically in the field of CSCL (Computer Supported Collaborative Learning) deals with design of collaborative activities, according to computer-interpretable specifications, such as IMS Learning Design, and their posterior enactment using LMSs (Learning Management Systems). A script that describes such collaborative activities is typically designed beforehand in order to structure collaboration, and defines the features that determine the behavior of the LMS, for instance, the sequence of activities or the groups/role distribution. In CSCL settings, group management and composition are especially relevant and affect the chances of achieving the expected learning outcomes. This paper presents a software tool, named *InstanceCollage*, which aims at facilitating the configuration and population of groups for IMS-LD scripts created with the authoring tool *Collage*, and discusses the implications of the IMS-LD specification with respect to this task. InstanceCollage is designed to process collaboration scripts based on CLFPs (Collaborative Learning Flow Patterns). Using this type of patterns, *InstanceCollage* focuses on the importance of understanding the function of groups within the learning strategy of the script. This paper describes the approach taken in *InstanceCollage* to facilitate this understanding for non-expert users. Additionally, two case studies are presented, which represent complex authentic collaborative learning scenarios, as a proof of concept of the functionality of this tool. The case studies are also used to illustrate the requirements of group configuration tools and to show that InstanceCollage complies to such requirements.

### Keywords

CSCL script, IMS Learning Design, group formation, Collaborative Learning Flow Patterns

### Introduction

Computer Supported Collaborative Learning (CSCL) research aims at enabling new forms of interactions among students to foster learning. Computer support is central in distance collaborative learning environments, but it may be used also to scaffold face to face collaborative activities (Stahl, Koschmann & Suthers, 2006). Technology can play different roles in collaborative learning, including the enabling of interactions in distance settings, scaffolding or guiding collaborative interactions among participants, or providing resources to carry out learning activities (Suthers, 2005). This paper will look into computer support for the guidance of collaborative activities through scripts. In spite of the potential benefits, collaborative learning scenarios do not guarantee learning, as the chances of effective interactions among participants depend on a wide range of factors, including group composition, the environment and the characteristics of collaborative activities (Dillenbourg, 2002). With the objective of structuring and scaffolding the collaborative process, collaboration scripts have been proposed as an effective solution to create conditions in order to promote the chances of effective interactions.

Different types of scripts tackle the specification of different features of collaborative scenarios. Typically, collaboration scripts are classified as macro-scripts or micro-scripts (Dillenbourg and Hong, 2008). Micro-scripts describe the communication model between participants (Kobbe et al., 2007), such as argument construction processes. In CSCL settings, software tools that control the communication process and induce students to follow a micro-script have shown to enhance learning (Weinberger, Fischer & Stegmann, 2005). On the other hand, macro-scripts model learning objectives, the sequence of activities that participants have to follow, role distribution (Kollar, Fischer & Hesse, 2006), resource distribution, or group formation (Kobbe et al., 2007). Through the configuration of these components, macro-scripts define the pedagogical method of the collaborative activities that students are expected to perform. In a similar fashion as micro-scripts, macro-scripts can also be deployed in software systems, or LMSs.

The development of LMSs capable of automatically managing collaborative activities described in a macro-script is an active research topic in the CSCL community. This kind of LMSs can, according to the collaborative learning strategy that drives a macro-script, manage groups of students, control activity schedules and task distribution, and provide students with resources and collaborative software tools.

In order to adapt the LMSs behavior to the content of a script, formal specifications have been proposed. A formal specification for learning scripts provides a model and a computational representation to describe the features of a learning setting, which can be interpreted by software systems. In the case of macro-scripts, IMS Learning Design (IMS-LD) (IMS Global Learning Consortium, 2003), the *de facto* standard to model learning scripts, enables the description of activity sequences, groups/roles, task distribution, etc. Currently different authoring tools and players have been developed, facilitating the task of creating IMS-LD scripts and their enactment in software systems in real learning scenarios. Designed as pedagogically neutral, IMS-LD has been also used to model collaborative activities (Tattersall, 2006). In spite of certain limitations (Miao, Hoeksema, Hoppe & Harrer, 2005), such as the lack of mechanisms to describe complex activity sequences (Harrer & Malzalm, 2006), it has been argued that MS-LD provides sufficient support for relevant features of CSCL macro-scripts (Hernández-Leo, Burgos, Tattersall & Kopper, 2007; Koper & Burgos, 2005), and has actually been used as the computational representation format of CSCL scripts by the Collage authoring tool (Hernández-Leo et al., 2006a), as discussed in the following section.

This paper is focused on the configuration of a particular aspect of the enactment of CSCL macro-scripts in IMS-LD players: *group particularization*. We use the term *particularization* throughout this paper to refer to the *configuration* of groups (i.e., the specification of the necessary number of groups for each type of group, to which students are assigned) and the *population* of groups (i.e., the assignment of students to groups) for a specific scenario. Group particularization may occur in different moments, including during the creation of the script (the design phase), during the preparation of the script for a particular scenario (in a concrete LMS) or even during the enactment of the activities. We will discuss the implications of collaborative learning and IMS-LD with respect to this issue. Group management is especially relevant in collaborative learning (Dillenbourg, 2002), as it affects the development of activities, and thus the learning outcomes. Several research works (Ikeda, Go & Mizoguchi, 1997; Ounnas, Davis & Millard, 2008) have focused on the importance of creating optimum groups, in the sense that the characteristics of the selected students create better chances for each group's collaboration to result in learning.

However, group formation policies are also part of the mechanisms on CSCL scripts (Kobbe, 2006). Along a single script, students might be required to carry out activities in various groups: in different phases, they may work together with the whole classroom, in small groups, and even carry out individual activities embedded in the collaboration script (Dillenbourg, 2004). For instance, the ArgueGraph and Universanté scripts (Dillenbourg, 2002) or those based on the well-known *jigsaw* technique (Aronson, Blaney, Stephan, Sikes & Snapp, 1978), propose activity flows in which students must change groups, in order to carry out different activities. These group changes are sometimes key mechanisms of the rationale of the script: In the case of the *jigsaw*, students move from *expert groups*, whose members work on a specific topic, into *jigsaw groups*, composed by students that have worked on different topics; this mechanism is expected to trigger certain interactions. Therefore, group formation needs to be understood as part of the learning strategy on which the script is based, and is interrelated with activity sequencing and task distribution.

Accordingly, group formation should be reflected in the way the LMS guides students along the activities included in the script. Group formation, being part of the learning strategy of the script, is a requirement justified also in technological terms. For instance, management of document flow and of software collaborative tools requires knowledge about group configuration (Palomino-Ramírez, Martínez-Monés, Bote-Lorenzo, Asensio-Pérez, & Dimitriadis, 2007). Computer support for the enactment of the script includes providing appropriate tools to enable the realization of activities such as communication tools (chats, forums, etc.) or others (e.g., collaborative editors) (Bote et al., 2008; Tchounikine, 2008). In this case, the system needs to take into account group configuration to allocate the needed resources and set appropriate access rights. For instance, the system should separate each group's private application space (for instance, chat sessions, own documents, etc.). On the other hand, document flow also depends on group composition. This is the case of, for instance, peer review activities, in which authors and reviewers need to exchange several documents in a precise way. If an LMS is expected to perform these tasks, the system needs information about groups: number, size and population.

These reasons have motivated the development of a software tool, *InstanceCollage*, with the objective of facilitating the particularization of groups that will intervene in a CSCL scenario. *InstanceCollage* is based on the general principles of a group formation tool, but it focuses on interoperability with IMS-LD compliant LMSs. While several tools currently allow users to particularize groups, they lack functionality to help non-expert users in the case of complex collaborative learning scripts, as we discuss in this paper.

The rest of the paper is structured as follows. The following section describes the type of script that *InstanceCollage* is focused on. After that, the paper discusses the task of group particularization (configuration and population, first in relation to the life-cycle of CSCL scripts in general, and later in the case of IMS-LD collaboration scripts. Having reviewed different approaches to this task, the paper describes *InstanceCollage*, indicating the rationale of its design as well as its main functionality. This functionality is further discussed in the context of three case studies, which illustrate the type of scripts that *InstanceCollage* is intended to process, as well as the technological issues related to group formation in authentic CSCL scenarios.

### Pattern-based approach for designing CSCL scripts

*InstanceCollage* is proposed within the context of a particular approach to CSCL script design. Focusing on the difficulties that non-expert designers face when designing collaboration scripts, this approach is based on the application of a certain type of learning design patterns, Collaborative Learning Flow Patterns (CLFPs), to facilitate the design task (Hernández-Leo et al., 2006a). This section discusses this approach, which is a key element to understand the features of *InstanceCollage*.

Design patterns (Alexander et al., 1997), which have been applied to different fields, including architecture and computer science, have been also proposed in the domain of e-Learning, as a mechanism for sharing ideas about educational design problems (Goodyear et al., 2004) and validated general solutions. Among different types of design patterns for learning, CLFPs propose solutions to create CSCL macro-scripts in order to achieve certain learning outcomes, extracted from best practices in collaborative learning. Thus, CLFPs capture best practices in organizing collaborative learning sessions, which can be used by designers to identify possible solutions applicable to real scenarios. With respect to this, CLFPs can be applied to a wide range of scenarios, since they are neutral with respect to specific domain content. Examples of CLFPs are PYRAMID, THINK PAIR SHARE or the aforementioned JIGSAW (Hernández-Leo et al., 2006a).

In order to facilitate the design of CSCL macro-scripts for non-expert designers, an IMS-LD authoring tool has been proposed and used: *Collage* (Hernández-Leo et al., 2006a). *Collage* supports designers through the process of reusing and adapting CLFPs. This process involves the selection of one or more CLFPs, according to certain learning objectives, and the configuration of the embedded activities and resources to the characteristics of the scenario.

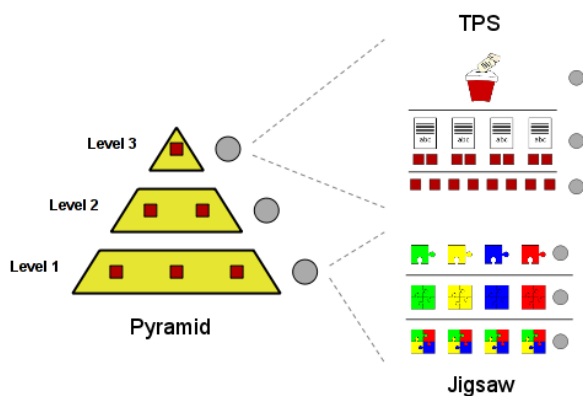


Figure 1. Representation of a CLFP-based script

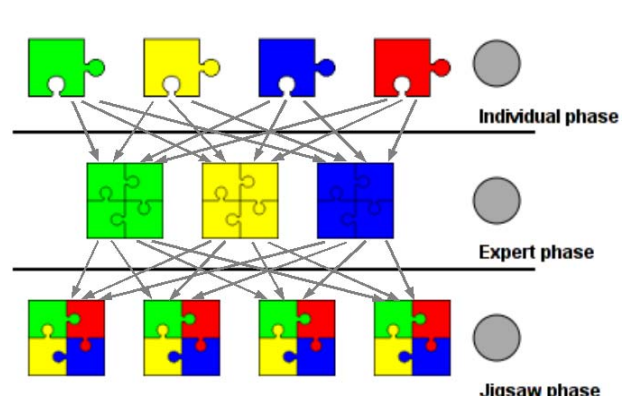


Figure 2. Groups and student change between groups in a JIGSAW-based script

Additionally, this tool creates a computer-interpretable representation of the script, using the IMS-LD specification (Hernández-Leo et al., 2007a), thanks to which scripts generated with Collage can be interpreted with any IMS-LD compliant system.

As mentioned before, each CLFP captures a concrete learning strategy: each pattern aims at creating certain conditions to promote the occurrence of interactions among participants, which in turn are expected to promote certain learning outcomes. This is related to one of the main features of *Collage*: the usage of a graphical representation to depict the sequence and distribution of collaborative activities (also called learning flow). Each CLFP has a specific graphical representation, which uses graphical metaphors to facilitate the recognition of CLFPs, and to provide visual hints about the type of activities contained in the script. Figure 1 shows the graphical representation for CLFPs JIGSAW, THINK PAIR SHARE and PYRAMID.

A multicase evaluation study of *Collage* has shown that the use of CLFPs facilitates the design task, and that the tool itself provides an intuitive and easy way for non-expert users to design complex collaborative learning scenarios (Hernández-Leo et al., 2008). However, as discussed in the following section, the IMS-LD specification, and therefore *Collage*, does not provide means to allow assigning students to groups. In addition, scripts created with *Collage* may have a complex group structure, rooted in the activity sequence and involving several group changes. Thus, configuring and populating groups in a real scenario can be a challenging task. Figure 2 shows how students change groups between the three phases that compose a JIGSAW-based script. The complexity of the script increases the odds of making mistakes in group formation, which would consequently hinder the development of the activities. For instance, a typical error may occur if a *jigsaw group* does not contain an *expert* from every different topic, as JIGSAW requires. This type of requirements needs to be taken into account during group particularization.

## Approaches to the configuration of groups in CSCL settings

In the previous section we indicated that InstanceCollage is focused on scripts generated with Collage, that is, IMS-LD scripts based on CLFPs. This section discusses the technical implications of IMS-LD with respect to group particularization. Before focusing on the characteristics of IMS-LD, we will review the life-cycle of CSCL scripts. Finally, this section reviews the particular characteristics of CLFP-based IMS-LD scripts.

### Life-cycle of CSCL scripts

From the conception of a CSCL macro-script to its application in a real scenario, there exist different phases of a CSCL script life-cycle that involve several actors. Even though there is no consensus about the exact composition of the life-cycle of CSCL scripts, it is possible to find several proposals in the literature. For instance, Weinberger, Collar, Dimitriadis, Mäkitalo-Siegl, & Fischer (2008) proposes a life-cycle, emphasizing the different steps involved in the production of an effective CSCL script, before its application in a computer-supported environment. This life-cycle includes the *specification* of the script, its *formalization* with a computer-interpretable specification and, possibly, a *simulation* of the activity flow, before *deploying* the script, i.e., enacting the activities with students. On the other hand, other research works have focused on the technical issues around the enactment of CSCL script in LMSs. Vignollet, Ferraris, Martel, & Burgos (2008) describe a life-cycle with the following four phases: *design*, *operationalization*, *enactment* and *evaluation*. Hernández-Leo et al. (2006b) describe a similar approach, shown in Figure 3, using the term *instantiation* (IMS Global Learning Consortium, 2003) to refer to the phase in which the CSCL script is prepared to be enacted. In both cases, the operationalization and instantiation phases include the tasks necessary to prepare the script for enactment in a concrete scenario using an LMS. The rest of this paper will employ the term instantiation in the sense of creating an instance of a CSCL script for a specific situation.

We are interested in the relationship between design, instantiation and enactment, with respect to group particularization. Configuring and populating groups can be accomplished during the design phase. In this case, the designer may use information about the participants to create the script. For instance, Isotani & Mizoguchi (2007) propose a tool, *Chocolato*, which considers not only the number of participants but also their current learning state, in order to provide suggestions concerning the most appropriate configuration of collaborative learning activities. Information about participants can be, therefore, valuable for designers (and tools with intelligent support for design) to select the pedagogical method on which the script is based.

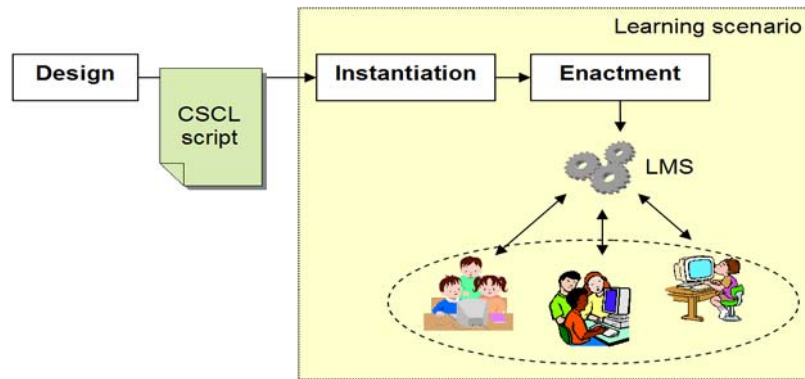


Figure 3. Life-cycle of CSCL scripts (Hernández-Leo et al., 2006b)

However, scripts do not contain necessarily all the information needed for the LMS to manage the activities. In particular, group composition may not be defined in the script. This approach is taken by the IMS-LD specification, which does not allow the definition of group population, as discussed below. Similarly, the learning design authoring tool developed by the LAMS project (Dalziel, 2003), does not include the task of populating groups during the design phase. In the case of LAMS, group population is done during enactment. The designer can however specify the group population policy (random, student choice or staff decision), which will determine the behavior of the player. Excluding group population from the script definition makes the script independent of a particular learning situation, and thus facilitates its reutilization. However, it also demands further configuration in later stages after design. Next section discusses this issue for the case of IMS-LD collaboration scripts.

### Current approaches in IMS-LD systems

If we focus on the case of IMS-LD scripts, it is the specification itself which constraints the moment when the particularization of groups is performed. As we indicated before, an IMS-LD script does not indicate the students assigned to each group. Therefore, the population of groups must be done always after the design phase: in the instantiation phase, or even during enactment.

With respect to groups, IMS-LD provides support to their definition by means of the `role` concept (IMS Global Learning Consortium, 2003). This concept is used to assign activities to different students: the script specifies the activities that each role performs and, in a particular situation, the participants are assigned to different roles. When participants are assigned to the same role, then they form a group. In collaborative learning, we are interested in modeling the existence of different groups, among which students are distributed. In order to model these groups, we need two “levels” of group information:

- a) In a single script, different *types* of groups may exist. Usually, different types of groups intervene in different phases, and each type can be used to distribute students in groups in a different manner. In the case of IMS-LD, different types of groups can be specified in the script; therefore, they are created in the design phase. For example, in the JIGSAW shown in Figure 2, there are two types of groups: *jigsaw* and *expert groups*.
- b) For each type of group, several groups may exist. In this case, these groups of the same type represent groups of students working in parallel in the same activities. Figure 2 shows four instances of *jigsaw groups* and three instances of *expert groups*.

With respect to having different groups of the same type, IMS-LD allows two approaches:

- 1) For each group of students, a different `role` is created at the design phase. In this case, the designer needs to know the exact number of groups that will exist in the classroom. When different groups perform the same activity in parallel, IMS-LD allows assigning those groups to the same activity. Using this approach, during the instantiation phase it is only necessary to assign students to groups.
- 2) For each different type of group, only one `role` is designed. In this approach, several roles may be created to represent the different types of groups that students will take during the script. However, in order to have the necessary number of groups of each type (to allocate students in different groups), they have to be created in the instantiation phase. Therefore, in this approach the design is simpler (the exact number of groups does not need to be known in advance), but instantiation requires also the configuration of the number of groups.

While both approaches are possible according to the IMS-LD specification, it seems clear that the later is better aligned with the option of leaving the population of groups for the instantiation phase. In contrast to the first approach, the second facilitates the reutilization of the script in scenarios with different number of participants, since the number of groups is not predefined in the script.

### Issues concerning CLFP-based scripts

Continuing with the discussion of the previous section about IMS-LD, now we focus on the specific case of CLFP-based scripts (i.e., those created with the *Collage* tool).

Previously we indicated two possible approaches to create different groups (of the same type) of student in IMS-LD scripts. *Collage* follows one approach or the other depending on each CLFP. Actually, only one CLFP allows the creation of different groups during the design phase: SIMULATION. Only in this case *Collage* offers functionality to create several groups, to which different names and activities can be assigned (including, therefore, different resources). The SIMULATION has this special feature because it was considered that the definition of the simulation learning process itself needs the identification of all the intervening roles (which can represent an individual or a group).

On the contrary, in the case of all the other CLFPs *Collage* takes the approach of defining one single role for each group type, thus leaving the creation of the necessary copies for the instantiation phase. For those CLFPs, therefore, *Collage* does not offer functionality to alter the number or groups. For instance, in the case of the JIGSAW, the script defines only one *jigsaw group* and one *expert group*. The PYRAMID, on the other hand, defines exactly one group per pyramid level. In order to distribute students among groups, it is necessary to create additional *jigsaw*, *expert*, etc. groups during the instantiation phase.

Creating the necessary groups can be a challenging task. While each CLFP defines (except for the SIMULATION) a low number of group types, scripts generated with *Collage* may contain several CLFPs. CLFPs are combined hierarchically, in the following way: one CLFP (we will call it *parent CLFP*) defines the global structure of the script. Other CLFPs (*child CLFPs*) can then be used to structure the activities of each phase of the parent CLFP. When this happens, the groups defined in the child CLFP become subgroups of each group that intervenes in the affected phase of the parent CLFP. With respect to the instantiation phase, this mechanism requires that the groups defined in the child CLFP be configured as many times as the number of groups of the parent CLFP. For instance, in Figure 1 the TPS and JIGSAW are each embedded in a phase of the PYRAMID. TPS groups, therefore, would need to be configured for every group of the *level 1* of the PYRAMID.

Finally, it is important to note that each CLFP may have its own rules, derived from the learning strategy on which the pattern is based. Figure 2 shows the rule implicit in the JIGSAW CLFP: groups formed in one phase must split so that each member should join a different group in the following phase. Another example is the Thinking Aloud Pair Problem Solving (TAPPS) (Lochhead & Whimbey, 1987), which requires creating pairs and, for each pair, distributing *solver* and *listener* roles explicitly. Assigning these roles is necessary, since they determine the activities performed by each member of the pairs.

In order to take all the above issues into account, and especially in the case of complex scripts (for instance, when several CLFPs are used), the person responsible for the particularization of groups could benefit from information about the exact purpose of each group in the script, including the CLFP to which each group is related, as well as the activities assigned to the group. Next section explains how *InstanceCollage* provides users with this information in the same tool that is used to configure groups.

### Particularization of groups in InstanceCollage

Based on the positive evaluation of *Collage*, the importance of group formation, and the relation between groups and the learning strategy of CLFPs, *InstanceCollage* was proposed to facilitate the task of group particularization. While *Collage* supports teachers in the creation of CLFP-based scripts, *InstanceCollage* is focused on the instantiation phase, aiming at providing an effective and efficient solution for this complex task. This solution is based on the fact

that CLFPs also contain information about group formation policies, and thus they can be used to provide valuable support for the instantiation phase.

*InstanceCollage* produces a description of the groups and their population, which is applicable only for the case of the specific instantiated CSCL script. The description is stored in XML format. Additionally, *InstanceCollage* can upload the particularization information to an LMS. Currently *InstanceCollage* is capable of publishing a CSCL script in *CopperCore*, creating a session and particularizing groups according to the configuration created by the user. In this way, the tool is integrated more transparently in the life-cycle of the script. Additionally, interoperability with another IMS-LD interpreter, named *GRAIL* (del Cid, de la Fuente-Valentín, Gutiérrez, Pardo, & Kloos, 2007), embedded in the open-source .LRN platform, is currently underway.

## Rationale

*InstanceCollage* facilitates the particularization of groups by representing them in relation to their function in the script. To achieve this, *InstanceCollage* uses the same graphical metaphors employed by *Collage* to represent each CLFP. Figure 4 shows, for instance, the graphical representation of the THINK PAIR SHARE CLFP. In this way, the user may be familiarized with the metaphors of each CLFP. If this is not the case, still the graphical representation is expected to be easy to understand. The metaphors are intended to be simple and, rather than displaying complete information about the script, they provide a hint about the structure of the collaborative activities.

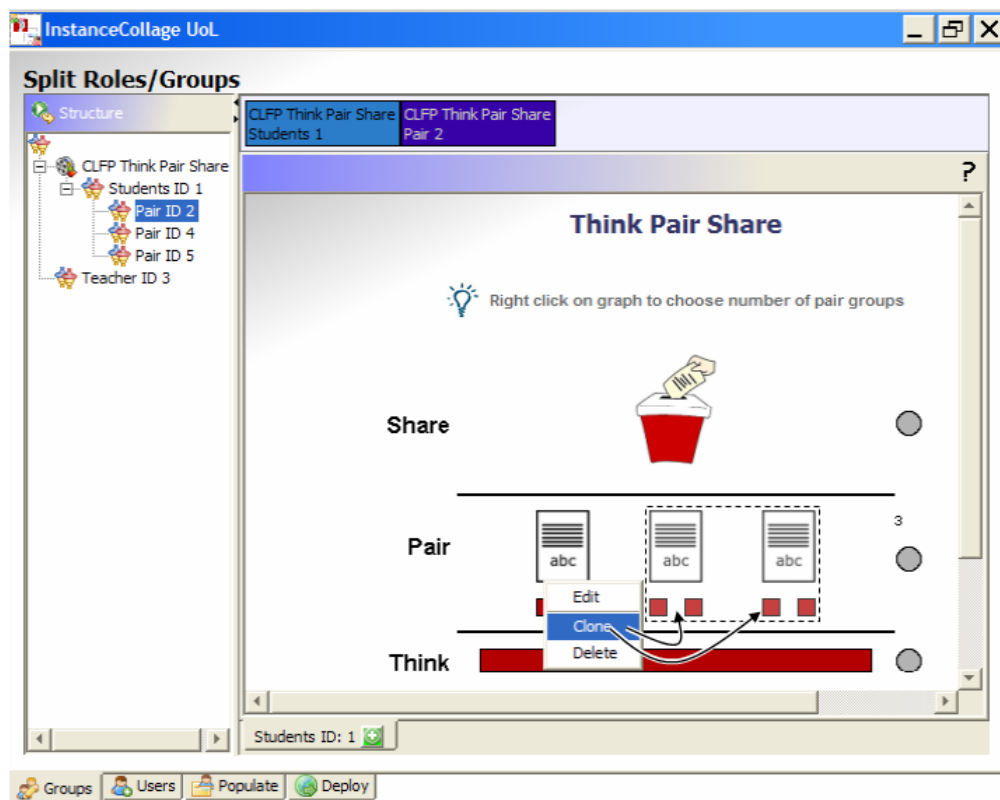


Figure 4. Creation of groups in *InstanceCollage*

The graphical representations for CLFPs depict, at least, phases and groups. Phases are typically represented vertically, separating the different steps of each CLFP. Additionally, the graphical representation uses specific symbols to represent the groups that intervene in each phase. In the case of the JIGSAW CLFP, for instance, colored jigsaw pieces represent different types of groups; in the SIMULATION CLFP, different icons represent each intervening role. In addition to using a meaningful representation for each group, the graphical interface locates

groups within the activity flow, indicating the activities in which each group intervenes. In this way, *InstanceCollage* facilitates identifying each group's function in the script, and understanding the relation between each group and the rules for group formation derived from CLFPs. In this way, *InstanceCollage* connects the learning strategy of the script with concrete information (i.e., the identity of participants) of the scenario in which that script will be enacted.

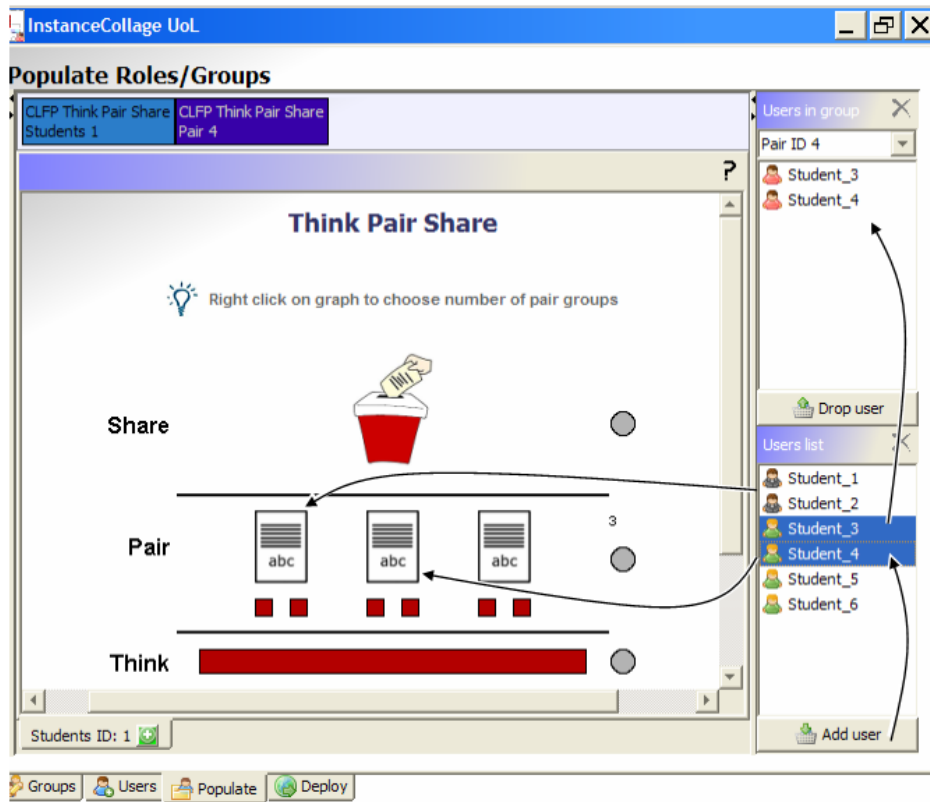


Figure 5. Population of groups in *InstanceCollage*

## Usage

In order to complete the task of group particularization, *InstanceCollage* guides its users through four steps, that are presented in different tabs after loading in the tool an IMS-LD script created with *Collage*. The first step, shown in Figure 4, consists in the creation of the required groups. To do this, the user can browse through the different CLFPs and their phases. After locating the group that intervenes in each phase, the user can create the number of groups necessary; when the configuration begins, each group has exactly one copy. Figure 4 represents the creation of two *pair* groups (for a total of three), using the menu option “clone”. On the left side of the figure, a tree shows the CLFP structure and the intervening; three *pairs* can be seen both in the graphical representation and the tree structure.

The second step (the first and second steps can actually be performed in any order), involves simply retrieving the names of the participants, including teachers and students. Currently *InstanceCollage* loads this information from a text file provided by the user. Once *InstanceCollage* has information about groups and students, the user can assign students to groups. The third step corresponds thus with the population of groups, and it is represented in Figure 5. The presentation is similar to that used in the first phase, but now additional information is shown concerning the users that belong to each group. In addition to intuitive means to add or drop users to the different groups, for each phase the tool indicates which users have been already assigned to a group, and those who have not, with the objective of preventing mistakes. In Figure 5, the graphical interface shows that *Student\_1* and *Student\_2* have already been assigned to a *pair* (using a different color in the user list on the right-bottom); the user then adds the next two students to the second *pair*.

In order to facilitate the first and third steps, the graphical representation is interactive. Each type of group is represented with a different icon, and the number of icons represents the number of groups for each type. By interacting with the different elements of the representation, groups can be created or deleted (thus modifying the number of groups of the same type), and students can be assigned to these copies (thus populating groups).

The final step involves the deployment of the script in the LMS. *InstanceCollage* automatically uploads the script; then, groups are created and populated for the particular scenario, according to the group particularization created by the user. After this step has concluded, the script is ready to be played by participants in the LMS. The interoperability between *InstanceCollage* and the LMS can be bidirectional: first, *InstanceCollage* might retrieve the list of participants, thus simplifying the second step described here. Second, *InstanceCollage* needs to connect with the LMS to upload the information about groups.

## Case Studies

This section discusses two case studies in which CSCL scripts created with *Collage* and formalized with IMS-LD were applied in learning scenarios. These case studies represent the whole life-cycle of real CSCL scripts in LMSs, including the particularization of groups depending on the participants (and their characteristics, such as their physical location) of authentic learning scenarios. At the moment in which each of these cases studies were carried out, different software tools were used to complete the group particularization. Afterwards, we carried out the particularization phase again using *InstanceCollage*. Our objective is, first, to evaluate whether *InstanceCollage* is capable of replicating the group particularization as was required in each case study. Additionally, this section examines the difficulties in the group particularization caused both by script complexity and the educational context. These difficulties are discussed with respect to the tools used in the case studies and the potential benefits of *InstanceCollage*.

### Case Study A: CTM2

The first case deals with a fifth year undergraduate course of Telecommunication Engineering, at the University of Valladolid, with 12 students attending the course. The case study comprises two two-hour sessions in which the students have to read a complex article on the topic of Network Management and analyze the main ideas presented in it (Hernández-Leo et al., 2007b).

In order to foster discussion among the students about the paper, a CSCL script was planned. This script aims at guiding students along a series of activities that involve individual learning, discussion, giving explanations to peers, or reaching agreement. The CSCL script was created with *Collage*, based on three patterns: JIGSAW, PYRAMID and TPS, organized as shown in Table 1. On the other hand, the *Gridcole* system (Bote et al., 2008) was employed for the CSCL script enactment, using a player based on the *CopperCore* IMS-LD engine. *Gridcole* implements a service oriented architecture, which enables the tailorability of the technological support for the learners by the integration of third-party tools (collaborative or not) on-the-fly. Other IMS-LD players, such as *SLeD* (Weller, 2006) could have been used. *SLeD* actually provides a graphical interface for the group particularization, more user-friendly than the tool provided in *CopperCore*. However, the case study was also used to evaluate the integration in *Gridcole* of third-party services; this was the main reason for the staff to choose this player. Actually, several tools, offered as services, were used by the participants in the case study: *Synergeia*, a shared, structured, web-based work space for collaborative learning, which includes tools for document sharing and asynchronous discussion; a chat, for synchronous communication; and *Quest*, a tool to publish and answer questionnaires on-line. As discussed below, there is a strong relationship between service allocation and provision and group particularization. This issue is currently being researched in *Gridcole*, which supports the binding of different instances of these services-like tools to each group in each activity. In this way, each group is provided with its own private work space for each tool. This binding is, therefore, based on the group particularization. For this reason, it was necessary to configure all groups before enacting the activities.

At the time when the case study took place, groups were particularized using *Clicc*, the tool provided with *CopperCore*. The particularization required 56 actions; being console-based, each action in *Clicc* is performed by means of a textual command. In spite of the low number of students, a total of 13 groups in total had to be created,

and each student had to be assigned to 4 groups (of different type), besides the *class* group that included all students. Figure 6 represents the group structure created for this experience. The usage of *Clicc* presented important difficulties. In addition to requiring the user to learn a series of text commands, the information provided by the tool concerning the particularization of groups as it was completed was difficult to read: it is presented in a XML-like text description, heavily based on the IMS-LD specification, which might not be readable for non-expert users. Actually, the configuration of groups was decided before using the tool with “pen and paper”, in order to understand the 56 commands necessary.

The same group particularization, with the same students, was carried out afterwards with *InstanceCollage* (Figure 7 shows the creation of the necessary groups). Loading the resulting particularization created with *InstanceCollage* in *Gridcole* yields the same result as the particularization with *Clicc*. However, the benefit of *InstanceCollage* lies in the guidance provided by the graphical interface. This tool provides comprehensive information to the user, which can be used to analyze and decide the number of groups and their population. In contrast to *Clicc*, *InstanceCollage* does not only enable the configuration of groups, but aims at supporting the user in the process of taking the decisions that complete this task.

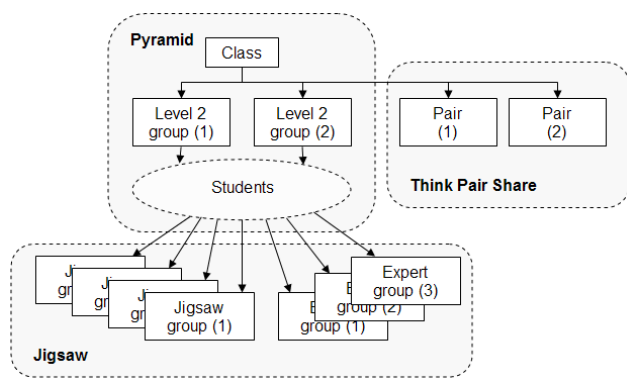


Figure 6. Group structure and relationship with CLFPs in Case Study A

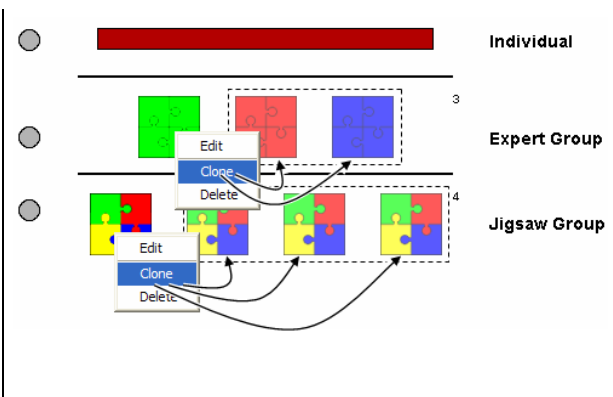


Figure 7. Creation of groups for the JIGSAW in *InstanceCollage*

### Case Study B: Mosaic

The second case was carried out in a coordinated activity between three universities, in the context of graduate courses on the topic of grid computing. Twelve students from the University of Valladolid, the Open University of Catalonia and Carlos III University of Madrid (four from each institution), participating on independent doctoral courses, worked together following a collaborative script. Given that each student worked from their own institution premises, computer support was particularly important to carry out the collaborative activities, which included both synchronous and asynchronous ones. The CSCL script was enacted using *GRAIL* (del Cid et al., 2007). One teacher from each institution monitored the activities locally.

The script was, similarly to that of Case Study A, composed by the PYRAMID and JIGSAW CLFPS, and PEER REVIEW activities; group particularization was done almost in the same way, but without configuring TPS groups. In this case, PEER REVIEW activities were included in different phases: first individually, during the JIGSAW (in the first level of the PYRAMID), and later in groups, during the second level of the PYRAMID.

The pedagogical context highlights the importance of providing information about the learning strategy of the script to facilitate group population. One of the distinguishing characteristics of this case study is the involvement of three teachers in the activities. After one of the teachers created the script with *Collage*, a second teacher configured the LMS to support the document flow between participants, which was done semi-automatically: the tool provided personalized instructions to each student about how to retrieve and upload the documents. Configuring the support for document flow requires information about groups and the activity flow. In different phases, the roles of *authors* and *reviewers* were distributed in different manners; for instance, exchange of documents happened between individual students or whole groups, depending on the phase. Therefore, document flow had to be configured taking into account not only the groups, but also the related activities. Moreover, the population of groups was decided

considering the physical location of each student, in order to foster students to collaborate both with students of the two other institutions and their own.

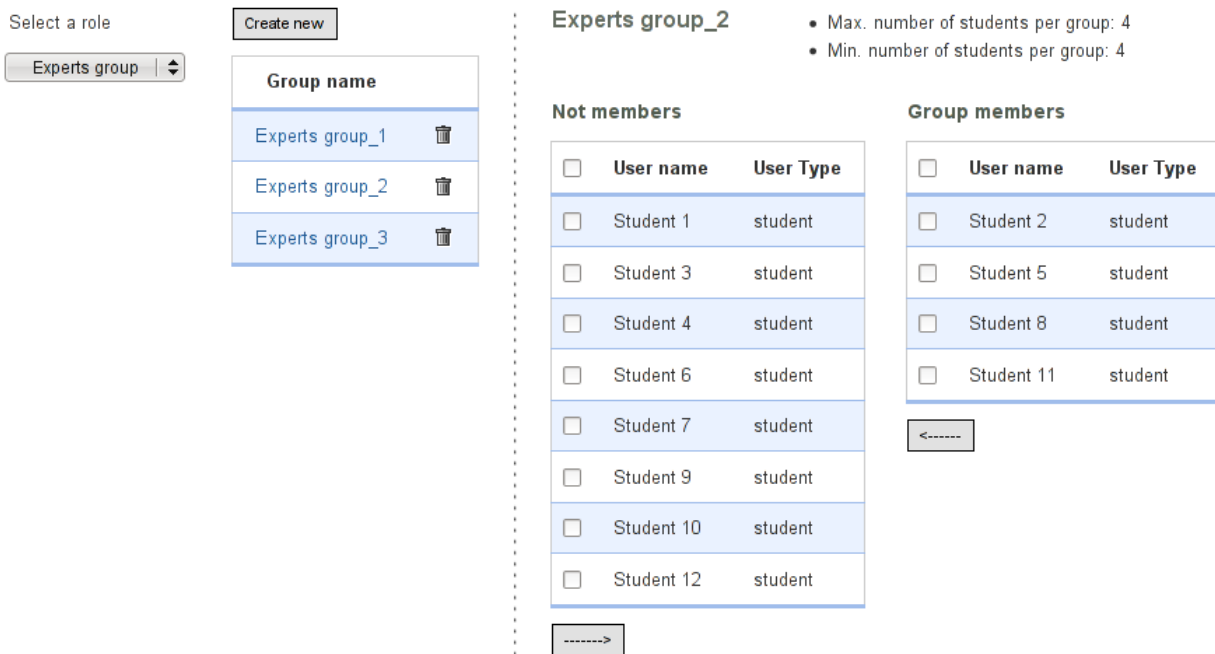


Figure 8. GRAIL interface for the population of groups

In the case study, group particularization was performed with the tool provided by *GRAIL*. Figure 8 shows a screenshot of this tool, in which the graphical interface can be seen. By using a web interface, the tasks of creating and populating groups are more intuitive than in *Clicc*. However, some problems were detected:

- When configuring different group “copies” (for instance, the three *expert groups* shown in Figure 8), the tool does not offer information about which participants have already been assigned to one of the groups of the same type. This can increase the chances of making the mistake of assigning the same student to several groups of the same type.
- In the case of group hierarchies (i.e., when parent groups contain child groups), the tool does not allow the assignment of students to the parent groups automatically by assigning them directly to child groups. In addition, there are no visual indications about the complete group structure.

Both issues are tackled in *InstanceCollage*. The particularization of the script used in this case study in *InstanceCollage* shows that it is possible to replicate the same group structure and population. In addition, *InstanceCollage* indicates, for each type of groups, what students have already been assigned to one group, in order to prevent the error of assigning one student to, for instance, several *expert groups*.

### Discussion and comparison with other tools

In the two cases, the collaborative activities were predefined in a CSCL script, based on CLFPs. An LMS was used to support the enactment of the activities. Complementing Figure 6, Table 1 shows the CLFP structure that defined the sequencing of activities. In the original cases, *Clicc* and *GRAIL* were used to particularize groups.

For both cases, *InstanceCollage* has shown to be capable of replicating the result of the group configuration and population phase. Since the number of students is low, further tests with the tool were carried out, showing that *InstanceCollage* supports the particularization of scenarios with a higher number of students. Up to 100 students were assigned to groups during the particularization of a test script; the group structure and population was later successfully uploaded in *CopperCore*.

Table 1: Necessity of support for group configuration and population.

Case study	Script	Need to show the learning strategy	Technological issues
<b>A) CTM2</b>	PYRAMID	Complexity of the script (group hierarchies).	Several tools provided by the LMS. Group resource allocation depends on the group configuration.
	Level 1		
	JIGSAW		
	Level 2		
	Level 3		
	TPS		
<b>B) Mosaic</b>	PYRAMID	Complexity of the script (group hierarchies).	Document flow depends on group formation: - Types of groups (individual or group roles) - Specific participants. - Activities in which document flow takes place.
	Level 1		
	JIGSAW		
	Ind. PEER REVIEW		
	Level 2	Different actors might be involved in the group configuration task.	
	Group PEER REVIEW		
	Level 3		

The tools originally used at the time of the case studies, not being specific of collaborative learning, show problems when configuring scripts in which students form complex group structures. In comparison with other tools, *InstanceCollage* is the only one that provides visual information about the moment in which groups intervene and, therefore, their function in the learning strategy of the script.

*Clicc* offers an unintuitive interface, which requires the user to enter a series of text commands. The information that this tool provides to describe the group particularization as it is been configured is presented in XML format, which makes its understanding difficult. On the other hand, the application embedded in *GRAIL* offers a more user-friendly interface. This tool implements easy-to-use group creation and population functionality. However, in the case of CSCL scripts in which both the activity sequence and group structure are relatively complex, the tool lacks the functionality of automatic assignment of participants in child groups to the parent group and of providing information about student distribution among groups during the population task.

Finally, there are other available tools for the instantiation of IMS-LD scripts, typically integrated with IMS-LD interpreters. *SLeD*, another player based on the *CopperCore* IMS-LD engine, also provides a graphical interface to particularize groups. In contrast with *GRAIL*, *SLeD* shows in a tree representation of the group hierarchical structure, facilitating the overview of the groups (types and copies of groups). However, similarly to *GRAIL*, it does not provide information about the participants already assigned to groups of a certain type. The *ReCourse* IMS-LD editor includes a *CopperCore* manager that enables the population of roles (i.e. groups). However, this tool does not allow the creation of different copies for each group type during instantiation. Therefore, this tool cannot instantiate scripts generated with *Collage*.

In contrast to other tools, *InstanceCollage* acknowledges the complexity of collaborative learning settings, such as the difficulty and importance of distributing students in groups adequately, and in accordance to the learning strategy, especially when groups are used as a means to improve the chances of social interactions. Information from CLFPs is used to display and communicate information about the purpose of groups within the script, not only during design, but also in the instantiation phase.

However, *InstanceCollage* presents a series of limitations. The first and main limitation is derived from the usage of CLFPs, which determine that only CLFP-based IMS-LD scripts can be processed by this tool. This limitation is shared with the related authoring tool, *Collage*. Therefore, *InstanceCollage* is not intended to substitute completely other group particularization tools, which are still needed to instantiate scripts generated with other authoring tools. However, we argue that *InstanceCollage* provides a more intuitive graphical interface and functionality in the case of CLFP-based scripts. CLFPs have prove useful to facilitate the access to collaborative learning scripts for non-expert users (for instance, teachers), and therefore *InstanceCollage* can help such users to close the life-cycle of CSCL scripts.

Moreover, a second limitation appears as the number of student increases. While *InstanceCollage* can handle a relatively large number of students, group population may become cumbersome, since this task needs to be done for each student (each of them may be assigned to several groups). Actually, this problem is shared by all the role/group particularization tools known to the authors. *InstanceCollage*, however, opens the possibility of implementing algorithms for automated group population that are aware of the pedagogical method of the script. Actually, there exist in the literature several approaches to automated grouping of students, based on different grouping criteria, for instance those proposed by Ikeda et al. (1997) and Ounnas et al. (2008). In addition to such strategies, CLFPs provide information about how to configure groups; see for instance the case of the JIGSAW in Figure 2. This information can be used to implement algorithms that automatically create groups that respect the pedagogical method of the script, depending of the CLFPs employed.

A third limitation is a technical one: in order to deploy the script, *InstanceCollage* needs to interact with the IMS-LD player. Currently *InstanceCollage* can automatically upload script (including the creation and population of groups) only in *CopperCore* and *Gridcole*. As mentioned before, the interoperability between *InstanceCollage* and other players, such as .LRN, is under development.

## Conclusions

This paper has described *InstanceCollage*, a tool for the configuration and population of groups in CSCL settings. Focused on IMS-LD scripts, *InstanceCollage* adopts a pattern-based approach, in which CLFPs are used to provide users (instructional designers or collaborative learning practitioners) with information about the pedagogical method of CSCL macro-scripts. *InstanceCollage* is intended to process scripts generated with the authoring tool *Collage*. In spite of the limitations due to the usage of CLFPs, these patterns and their combination offer important advantages to support the particularization of groups for non-expert users.

IMS-LD has shown to provide adequate support for collaborative learning scenarios. To that end, the life-cycle of scripts formalized with this specification requires further configuration of groups after the design phase, as this configuration will affect the development of activities. The two case studies discussed in this paper show that group formation is critical in CSCL settings. Besides guiding the enactment of activities, computer support includes the provision of software tools that enable collaboration, such as communication tools, and management of document flow. These technical issues justify the interest in creating detailed descriptions of group configuration and population, since implicit grouping may not be sufficient for the LMS to manage the delivery of meaningful collaborative activities.

Therefore, tools that enable users to particularize groups are needed. The case studies show that *InstanceCollage* is capable of configuring two CSCL macro-scripts which have been used in authentic situations. These scripts, based on CLFPs, are characterized by a complex pedagogical method. CSCL scripts may require assigning each student to several groups, using group changes between phases to create conditions that promote interactions. In addition, considering the technical issues and the whole life-cycle of CSCL scripts, it is possible that several actors (teachers, pedagogy experts, or other staff) are involved in the process of applying a script to a learning scenario. These different actors need to exchange information about, among other things, the learning strategy of the script and group formation decisions concerning the particular scenario. This issue reinforces the need to support the understanding of this information in the instantiation phase, for which *InstanceCollage* was developed.

In order to facilitate the particularization of groups according to the learning strategy of the script, the responsible staff or teacher should be aware of the relation between groups, activity design and the pedagogical method of the script, since grouping policies and related mechanisms (e.g., group changes) are key elements of the activity flow. Providing this information, in an intuitive graphical interface, is the main contribution of *InstanceCollage* with respect to other existing tools. This is achieved by means of CLFPs and their pattern-specific graphical representation. In spite of the limitations of *InstanceCollage*, the approach implemented in this tool for group particularization can be specially beneficial in the case of non-expert users working with complex collaborative scripts.

Further evaluation will be aimed at investigating the intuitiveness and ease of use of the tool. On the other hand, future work will be aimed at implementing automatic group population, which may be especially valuable in cases

with a large number of students. Automatic group population, however, needs to take into account the pedagogical model of each CLFPs. JIGSAW and SIMULATION patterns feature complex group changes; in the PYRAMID, on the other hand, appropriate group size in each phase is a relevant factor. Rules derived from CLFPs can be combined with other group formation policies based on student characteristics. Finally, it is worth noting that current work deals with the integration of *InstanceCollage* with LMSs capable of interpreting IMS-LD scripts, focusing on the automation of collaborative tools management.

## References

- Aronson, E., Blaney, N., Stephan, C., Sikes, J., & Snapp, M. (1978). *The jigsaw classroom*, Sage.
- Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., & Angel, S. (1977). *A Pattern Language: Towns, Buildings, Construction*, New York, USA: Oxford University Press.
- Bote-Lorenzo, M. L., Gómez-Sánchez, E., Vega-Gorgojo, G., Dimitriadis, Y., Asensio-Pérez, J. I., & Jorrín-Abellán, I. M. (2008). Gridcole: a tailorable grid service based system that supports scripted collaborative learning. *Computers & Education*, 51 (1):155–172.
- del Cid, J. E., de la Fuente-Valentín, L., Gutiérrez, S., Pardo, A., & Kloos, C. D. (2007). Implementation of a Learning Design Run-Time Environment for the .LRN Learning Management System. *Journal of Interactive Media in Education*, retrieved August 1, 2009, from <http://www-jime.open.ac.uk/2007/07/>.
- Dalziel, J. (2003). Implementing learning design: the Learning Activity Management System (LAMS). *Proceedings of the ASCILITE 2003 conference*, December 7-10, Adelaide, Australia.
- Dillenbourg, P. (2002). Over-Scripting CSCL: The Risks of the provision of service instances is a critical feature: Blending Collaborative Learning with Instructional Design. In Kirschner, P. A. (Ed.), *Inaugural Address, Three Worlds of CSCL. Can We Support CSCL?* Heerlen: Open Universiteit Nederland, 61–91.
- Dillenbourg, P. (2004) Framework for integrated learning. Kaleidoscope Network of Excellence deliverable D23.5.1.
- Dillenbourg, P., & Hong, F. (2008), The mechanics of CSCL macro scripts. *International Journal of Computer-Supported Collaborative Learning*, 3 (1), 5–23.
- Goodyear, P., Avgeriou, P., Baggetun, R., Bartoluzzi, S., Retalis, S., Ronteltap, F., & Rusman, E. (2004). Towards a pattern language for networked learning. *Proceedings of the Networked learning 2004*, Lancaster, UK: Lancaster University, 449–455.
- Harrer, A., & Malzalm, N. (2006). Bridging the gap – Towards a graphical modelling language for learning designs and collaboration scripts of various granularities. *Proceedings of the 6<sup>th</sup> International Conference on Advanced Learning Technologies 2006*, Kerkrade, The Netherlands, 296–300.
- Hernández-Leo, D, Villasclaras-Fernández, E. D., Asensio-Pérez, J. I, Dimitriadis, Y., Jorrín-Abellán, I. M., Ruiz-Requies, I., & Rubia-Avi, B. (2006). COLLAGE: A collaborative Learning Design editor based on patterns. *Educational Technology & Society*, 9 (1), 58-71.
- Hernández-Leo, D., Villasclaras-Fernández, E.D., Asensio-Pérez, J.I., Dimitriadis, Y., Retalis, S. (2006b). CSCL Scripting Patterns: Hierarchical Relationships and Applicability. *Proceedings of the 6th IEEE International Conference on Advanced Learning Technologies 2006*, Kerkrade, The Netherlands, 388–392.
- Hernández-Leo, D., Burgos, D., Tattersall, C., & Koper, R. (2007a). Representing Computer-Supported Collaborative Learning macro-scripts using IMS Learning Design. *Proceedings of the Second European Conference on Technology Enhanced Learning, CEUR Workshop Proceedings*. Crete, Greece.
- Hernández-Leo, D., Bote-Lorenzo, M.L., Asensio-Pérez, J.I., Gómez-Sánchez, E., Villasclaras-Fernández, E.D., Jorrín-Abellán, I.M., & Dimitriadis, Y.A. (2007b). Free- and Open Source Software for a Course on Network Management: Authoring and Enactment of Scripts based on Collaborative Learning Strategies. *IEEE Transactions on Education*, 50 (4), 292–301.
- Hernández-Leo, D., Villasclaras-Fernández, E.D., Asensio-Pérez, J.I., Dimitriadis, Y., Bote-Lorenzo, M.L., & Jorrín-Abellán, I.M. (2008). The added value of implementing the Planet Game scenario with Collage and Gridcole. *Journal of Interactive Media in Education*, retrieved May 1, 2009, from <http://www-jime.open.ac.uk/2008/22/>.
- Ikeda, M., Go, S., & Mizoguchi, R. (1997). Opportunistic Group Formation, In B. duBoulay and R. Mizoguchi (Eds.) *Artificial Intelligence and Education, Proceedings of AIED'97*, Amsterdam: IOS Press, 167–174.

- IMS Global Learning Consortium (2003). *IMS Learning Design Specifications*, Retrieved February 7, 2009, from <http://www.imsglobal.org/learningdesign/>.
- Isotani, S., & Mizoguchi, R. (2007) Deployment of Ontologies for an Effective Design of Collaborative Learning Scenarios. *Lecture Notes in Computer Science*, 4715, 223-238.
- Kobbe, L. (2006). Framework on multiple goal dimensions for computer-supported scripts, Kaleidoscope, D21.2.1 (Final), retrieved May 1, 2009, from <http://halshs.archives-ouvertes.fr/docs/00/19/02/97/PDF/Lars-Kobbe-2005.pdf>.
- Kobbe, L., Weinberger, A., Dillenbourg, P., Harrer, A., Hämmäläinen, R., Häkkinen, P., & Fischer, F. (2007). Specifying computer-supported collaboration scripts. *International Journal of Computer-Supported Collaborative Learning*, 2 (2-3), 211–224.
- Kollar, I., Fischer, F., & Hesse, F. W. (2006). Computer-supported collaboration scripts - a conceptual analysis. *Educational Psychology Review*. 18 (2), 159–185.
- Koper, R., & Burgos, D. (2005). Developing advanced units of learning using IMS Learning Design level B. *International Journal on Advanced Technology for Learning*, 2 (3), 189-202.
- Lochhead, J., & Whimbey, A. (1987). Teaching analytical reasoning through thinking aloud pair problem solving. *Developing critical thinking and problem-solving abilities, New Directions for Teaching and Learning*, 30, 73-92.
- Miao, Y., Hoeksema, K., Hoppe, H. U., & Harrer, A. (2005). CSCL scripts: Modelling features and potential use. *Proceedings of the Computer Supported Collaborative Learning*, Mahwah, NJ: Lawrence Erlbaum, 423–432.
- Ounnas, A., Davis, H. C., & Millard, D. E. (2008). A Framework for Semantic Group Formation. *Proceedings of the 7<sup>th</sup> IEEE International Conference on Advanced Learning Technologies*, Los Alamitos, CA: IEEE Computer Society Press, 34–28.
- Palomino-Ramírez, L., Martínez-Monés, A., Bote-Lorenzo, M.L., Asensio-Pérez, J.I., & Dimitriadis, Y. (2007). Data Flow between Tools: Towards a Composition-Based Solution for Learning Design. *Proceedings of the 7th IEEE International Conference on Advanced Learning Technologies*, Los Alamitos, CA: IEEE Computer Society Press, 354–358.
- Stahl, G., Koschmann, T., & Suthers, D. (2006). Computer-supported collaborative learning: An historical perspective. In R. K. Sawyer (Ed.), *Cambridge handbook of the learning sciences*, Cambridge, UK: Cambridge University Press, 409–426.
- Suthers, D. (2005). Technology affordances for intersubjective learning: A thematic agenda for CSCL. *International Conference of Computer Support for Collaborative Learning*, Taipei, Taiwan.
- Tattersall, C. (2006). Using IMS Learning Design to model collaborative learning activities. *Proceedings of the 7<sup>th</sup> IEEE International Conference on Advanced Learning Technologies*, Los Alamitos, CA: IEEE Computer Society Press, 1103–1104.
- Tchounikine, P. (2008). Operationalizing macro-scripts in CSCL technological settings. *International Journal of Computer-Supported Collaborative Learning*, 3 (2) 193–233.
- Vignollet, L., Ferraris, C., Martel, C., & Burgos, D. (2008). A Transversal Analysis of Different Learning Design Approaches. *Journal of Interactive Media in Education*, retrieved February 7, 2009, from <http://jime.open.ac.uk/2008/26/>.
- Weinberger, A., Fischer, F., & Stegmann, K. (2005). Computer-Supported Collaborative Learning in higher education: scripts for argumentative knowledge construction in distributed groups. *Proceedings of CSCL 2005*, Mahwah, NJ: Lawrence Erlbaum, 717–726.
- Weinberger, A., Collar, I., Dimitriadis, Y., Mäkitalo-Siegl, K., & Fischer, F. (2008). Computer-supported collaboration scripts: Theory and practice of scripting CSCL. In N. Balacheff, S. Ludvigsen, T. de Jong, A. Lazonder, S. Barnes & L. Montandon (Eds.), *Technology-Enhanced Learning. Principles and Products*: Berlin: Springer, 155-174.
- Weller, M. (2006). *The SLeD project: Investigating Learning Design and Services*, JISC E-learning Focus, Retrieved April 17, 2009, from <http://www.elearning.ac.uk/features/sledproject>.