Expert Systems with Applications 41 (2014) 3777–3798

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Automated instructional design for CSCL: A hierarchical task network planning approach

Geiser Chalco Challco [a], Marco Aurélio Gerosa [a], Ig Ibert Bittencourt [b], Seiji Isotani [c,*]

[a] *Department of Computer Science, Institute of Mathematics and Statistics, University of Sao Paulo, Rua do Matao, 1010, Cidade Universitaria, CEP 05508-090 Sao Paulo, SP, Brazil*
[b] *Computing Institute, Federal University of Alagoas, Campus A.C. Simoes, BR 104, Norte, km 97, Cidade Universitaria, CEP 57072-970 Maceio, AL, Brazil*
[c] *Department of Computer Systems, Institute of Mathematics and Computional Sciences, University of Sao Paulo, Avenida Trabalhador Sao-carlense, 400 Centro, CEP 13566-590 Sao Carlos, SP, Brazil*

ARTICLE INFO

Keywords:
Collaborative learning
Instructional design
Automated planning
Hierarchical task network planning
CSCL script design pattern
Unit of learning
Courseware generator

ABSTRACT

In Computer Supported Collaborative Learning (CSCL), one of the most important tasks for instructional designers is to define scenarios that foster group learning. Such scenarios, defined as Units of Learning (UoLs), comprise different components and are organized according to pedagogical approaches to orchestrate group learning processes. Examples of UoL components are learning objects, student roles, student characteristics (e.g., background, preferences, learning styles, etc.), instructional/learning goals, and activities, among others. Thus, the instructional design (ID) of a proper UoL for CSCL is a complex task that requires practice and experience. This is particularly true when designing, developing, adapting, and customizing UoLs, taking into consideration different instructional/learning goals and individual preferences of students. This paper therefore proposes using a Hierarchical Task Network (HTN) planning approach to automate and optimize the tasks of designers. To accomplish that, we define an initial CSCL scenario as "an ID task" and "a set of information related to students and the domain to be taught." Then we propose a model that formally describes ID for CSCL as HTN planning, where the initial CSCL scenario is adapted and refined according to student needs. In this model, the ID strategies are defined as hierarchical tasks and methods into a planning domain definition, and the initial CSCL scenario is defined as a planning problem definition. To validate our approach, we develop a CSCL courseware generator that (i) helps designers to set up an initial CSCL scenario; (ii) automatically generates a personalized UoL based on a given initial scenario; and (iii) supports the adaptation of UoLs.

1. Introduction

The design and orchestration of effective and well thought out Collaborative Learning (CL) scenarios is a challenge that the Computer Supported Collaborative Learning (CSCL) community has faced for several decades (Dillenbourg, 2002; Kobbe et al., 2007). The goal of creating a scenario in the context of group learning is to properly structure the interactions among peers, thereby increasing the learning gains of individuals. Such a scenario, defined as a Unit of Learning (UoL), is a delimited piece of education or training, such as a course, module, or lesson, in which the elements describing "what should to be taught" are structured according to different pedagogical approaches to define "the way in which participants (students and teachers) should interact."

Development, adaptation, and customization of UoLs for CSCL requires careful instructional design (ID). Thus, some researchers developed authoring tools that use diagrams of graphs (e.g., Cool Modes (Pinkwart, 2003)) and CSCL Design Script Patterns (e.g., COLLAGE (Hernández-Leo et al., 2006), and CHOCOLATO (Isotani et al., 2010)) to better support the ID. However, the authorship of UoLs for CSCL with these tools is difficult if the goal is to develop, adapt, or customize units taking into account the individual characteristics of students. This is because the designer must decide what activities should be defined, what groups of students should be formed, what learning goals must be achieved, what roles must be played, what tools and materials must be used, and so on.

For example, to develop a UoL in which four students acquire knowledge about the mathematical concept "derivative" through exercises, an instructional designer (in the analysis phase) identifies what will occur in the CL scenario by setting instructional/learning goals. In setting of these goals, the designer defines the purpose of elements to be obtained as the desired stages of learning development (individual goals), skills, and attitudes to be achieved when the run of UoL is finished. In this example, the designer sets "the individual goals for all student as the acquisition of knowledge about derivative at level restructuring", and "the

* Corresponding author. Tel.: +55 1633738169.
E-mail addresses: geiser@ime.usp.br (G.C. Challco), gerosa@ime.usp.br (M.A. Gerosa), ig.ibert@gmail.com (I.I. Bittencourt), sisotani@icmc.usp.br (S. Isotani).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.12.016

acquisition of attitude positive interdependence" as the purpose of UoL (instructional/learning goals).

Next, the designer (in the design phase) defines how the learners attain these goals through the group formation, the selection of tools and learning materials for students, and the definition of learning plans. In the group formation, the designer may want two groups with two students or one group with four students. This decision depends on the availability of the students, teacher or the characteristics of individual students (e.g., stage of learning development related to the concept "derivative"). In the selection of learning material, the designer performs finding the proper exercise in external repositories (e.g., exercises that none of the students have previously seen). To define the learning plans in the CL scenario, the designer employs an authoring tool that uses CSCL Design Script Patterns to decide how to apply any pattern (e.g., COLLAGE or CHOCOLATO). For example, the designer applies the "jigsaw" pattern if the selected exercise can be divided into two parts and the students have experience in CL, the "pyramid" pattern if the selected exercise is difficult and the students don't have experience in CL, or the "distributed cognition" pattern if the students have experience using the knowledge related to "derivate" or they have cognitive skills. Alternatively, the designer can decide to define the learning plan by employing an authoring tool that uses diagrams of graphs (e.g., Cool Modes).

Finally, the designer (in the development phase) arranges the concrete learning plans and develops all media and any supporting documentation that will be used in the CL scenario. The designer must define roles according to the ability of a student to perform a role and the behavior the role a player performs. In this example, if the designer decides to use the pattern "distributed cognition," for a group with all students, he will set roles "instructor" and "learner", and he will define of transmission/reception messages for each student. At this stage of the scenario it becomes clear what learners should participate, what roles they play, what behavior they perform, what learning materials they learn, and what educational benefits they are expected to acquire. Next, the designer will develop learning environments to enable interaction among students in the CL scenario. The learning environment is represented as an arrangement of tools for the members of the group. In this example, the designer can define an environment in which the student with the role "learner" will use a tool "simulation" to participate in the exercise, and the student with the role "instructor" will use the tool "monitoring learning process." The tool "making a report paper" is set for all of the students, the tool "discussion channel" (like a chat system) is set for the student with the role "learner," and the tool "monitoring discussion" is set for the student with the role "instructor."

In Artificial Intelligence, the automated planning field studies the automatic generation of action sequences, called plans. The execution of plans leads to the satisfaction of certain goals. Therefore, there is a direct relationship between ID and automated planning, in which plans are used to generate a UoL that defines an individualized teaching–learning process for CSCL. In automated planning, HTN planning is a technique that uses hierarchical tasks and methods to represent domain-specific strategies. The methods define multiple ways to deconstruct the tasks into sub-tasks until getting a consistent and coherent plan. The ID of UoLs using CSCL Design Script Patterns can be documented as hierarchical tasks and methods. This knowledge focused on the rationale of the ID process of elements to be included into UoLs.

In this paper, we present a model that formalizes the ID for CSCL as a HTN planning approach. In the second section of the paper, we present basic information about ID using CSCL Script Design Patterns. In the third section, a brief description of HTN Planning is summarized. In the fourth section, we present our approach as a model that formalizes ID for CSCL as HTN planning. In the fifth section, we present an example that shows the formulation of a planning problem and its results. In the sixth section, we present related works and compare them with the results obtained by this research. Finally, we present conclusions and discuss future work.

2. Instructional design using CSCL Script Design Patterns

CSCL represents a multidisciplinary paradigm within Technology-Enhanced Learning, in which computers are employed to enhance various educational aspects of group learning (Stahl et al., 2006). One of the main concerns for CSCL is how to improve social interactions, an essential element of group learning. The goal in this context is to increase the probability of reaching success in CL scenarios by providing students with a set of instructions that promote fruitful collaboration through learning activities. This set of instructions is mediated by what is called a CSCL script.

To run CSCL scripts on online learning environment such as Learning Management Systems (LMS) requires a designer to understand the learning context and map this context to specific scripts. This fact limits their broad applicability and imposes significant time and cost efforts each time a new CSCL script is required. Thus, the researchers Hernández-Leo, Pérez, and Dimitriadis (2004), Hernández-Leo, Villasclaras-Fernández, Asensio-Pérez, and Dimitriadis (2009) propose the use of CSCL Script Design Patterns and IMS Learning Design (IMS-LD) specification to solve these problems. In contrast to Learning Object Metadata, which is used for delineating reusable chunks of learning materials, the IMS-LD is focused on specifying the pedagogical approach to a problem (IMS GLC, 2003; Koper and Olivier, 2004). Specifying the type of pedagogical approach to a problem enables the computational representation and automatic interpretation of CSCL scripts. To be able to run CSCL scripts in different LMSs, a UoL is represented in IMS Content Packaging (IMS-CP) specification as a package that must contain a manifest file and a collection of resources (IMS GLC, 2004; Olivier and Tattersall, 2005). The manifest file named "imsmanifest.xml" contains the description of CSCL Scripts according to IMS-LD.

The IMS-LD specification is based on a social model, where the components of a CSCL script are defined as roles and activities related to different environments. Each environment consists of a set of learning objects (e.g., exercises, examples, software simulations) and services (e.g., forums, chats) to be used during the activities. In this sense, as discussed in the literature (Dillenbourg and Jermann, 2007; Hernández-Leo, 2007), the description of script mechanisms uses a metaphor of play. Generally, a play consists of a sequence of acts, in which activities are performed in parallel by different player roles. Each act delineates the task distribution, group formation, and sequencing of activities as a set of role-parts elements. When an act is completed, the transition from one act to another serves as a synchronization point, ensuring that all participants start the activities of an act simultaneously.

2.1. CSCL Script Design Patterns

CSCL Script Design Patterns are CSCL Scripts that provide solutions to recurrent situations. These patterns capture the essence of best (or good) educational practices when creating CSCL scripts through a set of guidelines. The goal of guidelines provided by these patterns refers to foster productive interactions for CSCL scenarios at different levels. With respect to this, Hernández-Leo, Villasclaras-Fernández, Asensio-Pérez, and Dimitriadis (2009) classify these patterns using the hierarchical structure shown in Fig. 1. This structure includes: (i) patterns related to the CL flow, called Collaborative Learning Flow Patterns (CLFPs), which tackle

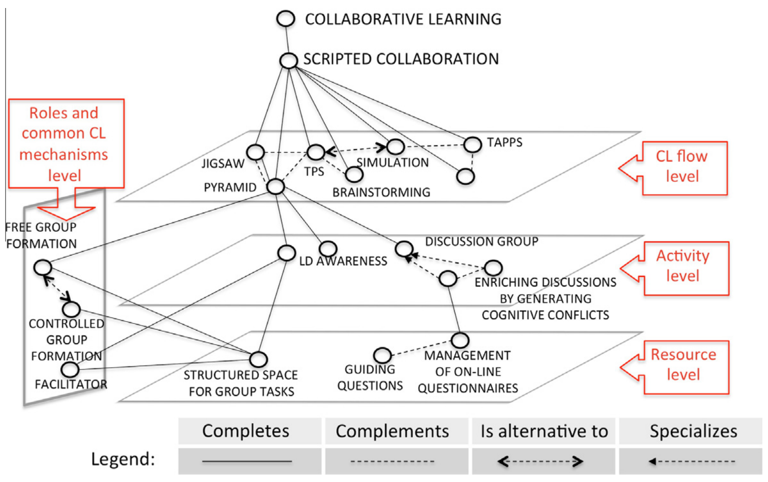


Fig. 1. Hierarchical structure of CSCL script design patterns (Hernández-Leo, 2007).

the definition of sequences of collaborative activities and grouping policies; (ii) patterns related to the activity level for the design of collaborative activities; (iii) patterns related to the resource level to configure the resources, such as documents, and software tools; and (iv) patterns related to configure roles or other CL mechanisms.

Patterns at different levels are complementary and require interaction to be complete. They can relate to one another through four different types of connections (Hernández-Leo, 2007). (1) “*Completes*” define the composition of patterns indicated by the aggregation model. Patterns at higher levels need patterns at lower levels for completeness, resources complete activities that, in turn, complete CL flows. CLFPs can be combined in such a way that a phase can be replaced with another CLFP. (2) “*Complements*” define the concatenation of patterns. Some patterns at the same level can complement each other. Two patterns related through this connection form parts of a larger whole. (3) “*Is alternative to*” define alternative solutions to the same situation. Alternative patterns are interchangeable, but they cannot be used in a complementary way. (4) “*Specializes*” define variations of identical patterns according to “degrees of freedom.” These variations formulate their use in different situations.

In this paper, we only use patterns related to CL flow level and activity level. The CLFPs used to develop CSCL Script are called macro-scripts, which delineate the organization of coarser-granularity activities, including the description of groups, roles, and flows of CL activities (e.g., Pyramid, and Jigsaw) (Dillenbourg & Jermann, 2007). Patterns related to activity level are used to develop micro-scripts, which provide detailed support within specific CL activities through the description of communication process among participants (e.g., Peer-tutoring and Discussion Group) (Weinberger et al., 2005).

2.1.1. Instructional design of macro-scripts

According to Dillenbourg and Jermann (2007) and Kobbe et al. (2007), the script mechanism of a macro- script is a linear structure of phases that specify how participants should collaborate through the definition of group activities. One group activity describes the method of distribution of a task for one group. A task is a list of learning sessions, in which each session is defined as a triple “input, activity and output.” The input is a set of resources that will be used by the group, the activity is a description of what will be done, and the output is a set of resources that will be created by the group during the learning session. A learning session is a unit with specific learning goals, and represents the abstraction of an activity whose specific communication model is not given (Villasclaras-Fernández, Isotani, Hayashi, & Mizoguchi, 2009b).

The Fig. 2 shows the use of the “jigsaw” pattern to obtain a macro-script that solves the problem of understanding a paper comprised of two sections (excluding the summary, introduction, and conclusion) by a group of four students (l1, l2, l3 and l4). This pattern, as defined in (Hernández-Leo et al., 2006), proposes that in order to solve a complex problem that can be divided into independent sub-problems, the script mechanism is composed of three phases “individual,” “expert,” and “jigsaw”. In the individual phase, each student studies or works in a learning session around a particular sub-problem “individual study of section 1” (for l1 and l3) or “individual study of section 2” (for l2 and l4). In the expert phase, the students that study the same problem meet in a group to exchange ideas until they become experts in the sub-problem. In this way, the “expert group 1”(formed by l1 and l3) performs the learning session “discussion of section 1” and the “expert group 2” (formed by l2 and l4) performs the learning session “discussion of section 2.” Finally, in the jigsaw phase, groups of students meet to contribute their expertise to solve the entire problem. Thus, in order to understand the paper, the “jigsaw group 1” (formed by l1 and l2) and “jigsaw group 2” (formed by l3 and l4) perform two tasks composed of three learning sessions: “discussion of section 1,” “discussion of section 2,” and “final discussion.” In the session “discussion of section 1” in the expert phase, the group of students l1 and l3 perform the activity “discuss with members that have read the same section to elaborate a summary,” where the input resources are the paper and one asynchronous forum, and the output resource is a summary of section 1.

2.1.2. Instructional design of micro-scripts

The CL ontology developed by Isotani and Mizoguchi (2007) provides a conceptual framework for delineating the script mechanism of a micro-script as a flow of interactions. Each interaction is represented as an IL Event. An IL event is an event that performs the description of the teaching–learning process through a learning event and an instructional event. These events include actors and actions. An actor can act as an instructor (learner doing an instructional action) or as a learner (student doing a learning action). The relations between a role and a flow of interactions are modeled as learning strategies that explain why a student with role of instructor or learner is interacting with another student (Isotani & Mizoguchi, 2008).

Fig. 3 shows the micro-script obtained using the pattern “peer-tutoring” on the learning session “discussion of section 1 for student l1 and *l*2” defined in “jigsaw phase” (shown in Fig. 2). In

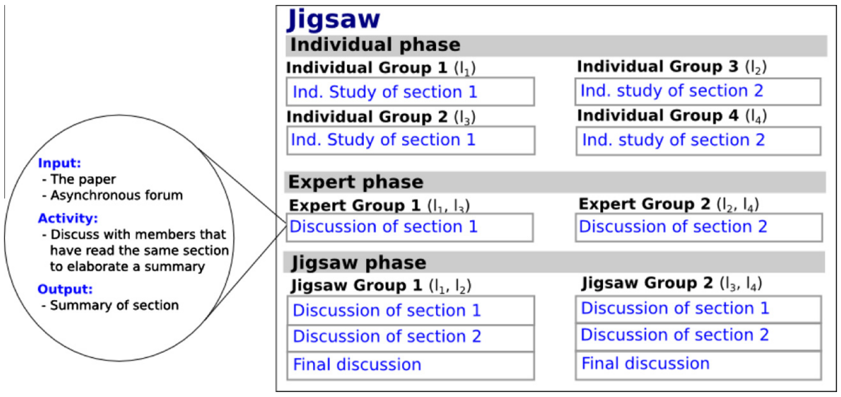


Fig. 2. Example of a macro-script obtained using the "jigsaw" pattern.

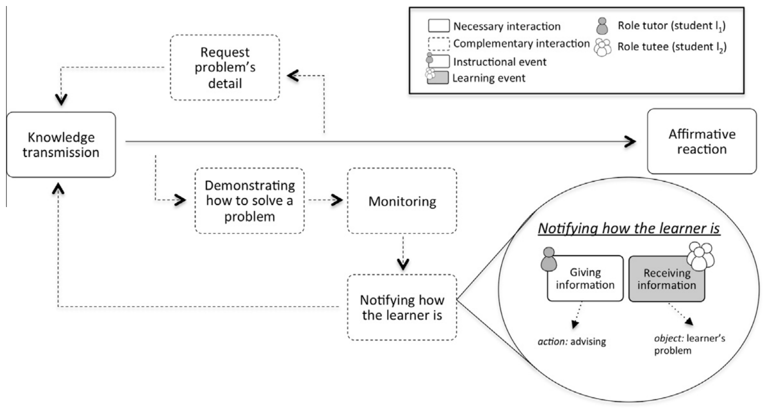


Fig. 3. Example of a micro-script obtained using the "peer-tutoring" pattern.

the pattern "peer-tutoring," the students who do not have "complete" knowledge of the event content can play the role of "tutor" or "tutee." The tutor must acquire more knowledge in order to teach and organize his/her thoughts in an understandable manner, and by being helped the tutee will acquire or construct his/her knowledge as well. In this sense, the student l1 plays the tutor role because he/she has more knowledge, while the student l2 play the tutee role because he/she is less knowledgeable about "section 1." The flow of interactions of micro-scripts begins with the "knowledge transmission" and ends with "affirmative reaction." Alternatively, if the tutee has questions then the tutor and tutee perform the interaction "request problem detail" in order to address the problematic area. To solve the problematic area of understanding, the tutor and tutee perform the interactions "demonstrating how to solve a problem," "monitoring," and "notifying how the learner is." The interaction "notify how the learner is" is modeled as an IL event composed by an instructional event "giving information" and a learning event "receiving information." Finally, the action of the instruction event is "advising," while the learning object related to the learning event is "learner's problem" as defined in the environment.

2.2. Instructional design strategies

In this work, the ID process of UoLs using CSCL Script Design Patterns is based on the instructional planning approach. This approach has been proposed by Wasson (1996) and consists of mapping out a global sequence of instructional goals and actions that enables the system to provide consistency, coherence, and continuity throughout an instructional session, and enables this global sequence to then be interspersed with local goals generated when instructional opportunities arise. Since the 1980s, various Intelligent Tutoring Systems (ITSs) were developed using this approach (Peachey & McCalla, 1986;Ullrich, 2008; Van Marcke, 1998; Vassileva, 1998; Vrakas et al., 2007). All endorse a planning approach to managing instructional interactions with the students through a component that is responsible for determining "what to do next at each point in an instructional interaction," hence an approach that controls the system's behavior.

An interaction in instructional planning is any form of interaction between students and instructional materials, or computer-mediated interaction between collaborating students or between students and teachers(Wasson, 1996). In this sense, we define an ID task as a goal or action that defines "what a designer must do." This task is an activity that must be performed for the creation or configuration of the elements of UoL. The ID tasks are defined at various levels of granularity, and the minimal activity that must be performed is called the primitive ID task (or ID action). To obtain local goals when opportunities arise, each non-primitive ID task has a set of ID methods that describe "how to achieve" a task. An ID method has an explicit description of guidelines "what explain the rationale" for the creation or configuration of an element of UoL using CSCL Script Design Patterns. This kind of formalization makes it possible to describe alternative ways to achieve an ID task.

Fig. 4 shows the representation of an ID Strategy related to the ID task "create a macro-script." In this model, the task has four methods that define the creation of a macro-script using one CSCL Script Design Pattern. Thus, the first method uses the CLFP "jigsaw" to create a macro-script according to IMS-LD specification.

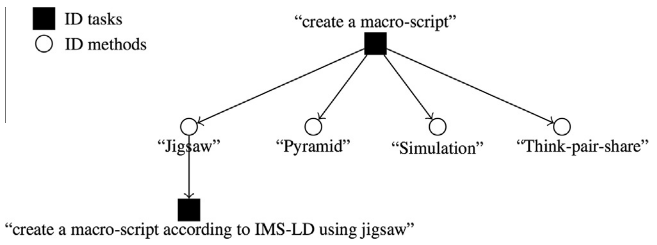


Fig. 4. Example of ID Strategy related to the "create macro-script" ID task.

In this model, the knowledge relates the ID using CSCL Script Design Pattern and the IMS-LD specification is represented as a set of ID strategies. Thus, we define six guidelines to perform the conversion of this knowledge into HTN planning knowledge.

3. Hierarchical task network planning

In HTN planning, the generation of action sequences (called plans) that leads to the satisfaction of certain goals by agents is represented as sets of tasks (task network), and methods that decompose non-primitive tasks into sub-tasks until reaching a level of primitive tasks which can be solved by operators (Garrido, Onaindia, & Sapena, 2008; Ghallab, Nau, & Traverso, 2004). Our courseware web service uses the HTN planner JShop2ip (*Java Simple Hierarchical Planner Order 2 for Instructional Planning*), a version of JShop2 (Nau et al., 2003) developed to support the instructional planning in an Adaptive and Intelligent Educational System for CL and non-CL domains that others HTN planners cannot solve.

JShop2ip takes a "planning domain definition" and a "planning problem definition" as inputs to solve a planning problem. The planning domain definition is composed of operators, methods, axioms (horn-clause-like statements for inferring conditions that are not explicitly define in the problem planning definition), and external functions (code calls to external procedure or software) to represent the human expert knowledge as heuristic knowledge in the form of decomposition methods. The planning problem definition is composed of an initial state (represented as a set of logical atoms that are assumed to be true at the time when the planner will begin the planning process) and an initial task network (a set of tasks to be performed) that describe the information of scenario.

The JShop2ip formalism uses different kinds of *terms* as a number, a constant symbol, a variable symbol, a list term, or a call term to represent the domain and problem definitions. A constant symbol begins with a letter or an underline. A variable symbol begins with a question mark "?". A list term is represented by the form (*t*1 *t*2 ... *tn*), where each *ti* is a term. A call term is represented by the form (call *f t*1 *t*2 ... *tn*) where *f* is a function that executes an attach procedure and each *ti* is a term. The logical atoms are represented as n-tuples (*p t*1 ... *tn*) where each *ti* is a term.

Axioms are horn-clause-like statements for inferring conditions that are not mentioned explicitly in the current state. An axiom is represented by the form (:- *a* [*n*1] *L*1 ... [*nm*] *Lm*) where "*a*" is the axiom's head represented by the logical atom, each *Li* is a logical precondition, and each *ni* is a optional symbol referring to the name of each *Li*. The axiom *a* is true if *L*1 is true, or if *L*1 is false but *L*2 is true, or if all of *L*1, ... *Ln*1 are false but *Ln* is true.

An operator represents a primitive task and has the form (:operator *h pre del add*), where: *h* is an atomic task (*p v*1 ... *vn*) that begins with an exclamation mark "!" in the symbol task *p* and each *vi* is a term. "*pre*" is a logical expression; while "*del*" is a list of logic atoms to be removed from the current state; and "*add*" is a list of logical atoms to be added. A method has the form (:method *h pre* [*rel*] *t*) where: "*h*" is an atomic task that represents a non-primitive task as a expression of the form (*p v*1 ... *vn*), in which "*p*" is a task symbol and each *vi* is a term; "*pre*" is a logical precondition; "*rel*" is an optional relative condition; and "*t*" is a list that defines the subtasks into which "*h*" can be decomposed. A logical precondition is either a logical expression or a first satisfier precondition represented by the form (:first *L*) where "*L*" is a logical expression, such a precondition causes JShop2ip to consider only the first binding that satisfies "*L*". A relative condition has the form (:relative *a L*), where "*a*" is a numerical value that represents the value of applicability to be increased or decreased, and "*L*" is a logical expression. Fig. 5(a) shows part of planning domain definition that details the method "transport a package ?p with transport ?t from location ?l1 to ?l2" (transport ?p ?t ?l1 ?l2) and the operator "move a transport ?t from location ?l1 to ?l2" (!move ?t ?l1 ?l2).

Fig. 5(b) shows the HTN planning process for initial task "transport a package p from location l1 to l2" (transport p t l1 l2), where the initial task is decomposed into subtasks "dispatch t to l1" (dispatch t l1), "load package p on t" (!load t p), "move t from l1 to l2" (!move t l1 l2), and "return t from l2" (return t l2). The task "dispatch t to l1" is decomposed into subtasks "reserve t" (!reserve t) and "move t from home to l1" (!move t home l1), and the task "return t from l2" is decomposed into subtasks "unload package p from t" (!unload t p) and "move t from l2 to home" (!move t l2 home). Thus, the solution plan is (!reserve t), (!move t home l1), (!unload t p) and (!move t l2 home).

Because HTN planning enables the representation of human expert knowledge as hierarchical tasks and methods, it is a suitable technique for representing ID strategies using CSCL Script Design Patterns. In the next section, we will describe the guidelines to make this representation and the proposed model to automated ID for CSCL.

4. Instructional design for cscl as htn planning

Fig. 6 shows our approach to automated ID for CSCL as HTN planning. This model is based upon the classical ITS model (Kaplan and Rock, 1995; Woolf, 2010). Where the *student model* and the *domain model* are defined in external repositories, the *pedagogical model* is defined in planning domain definition as ID strategies. To start the ID, the designer (through an *authoring tool*) defines the initial CSCL scenario as a planning problem definition, and (optionally) can also define the information of "student model" and "domain model" in external repositories. During the planning process, the HTN Planner uses a set of axioms (A) to make queries in external repositories. Once the HTN planner has finished the planning process, the actions of the plan are translated into UoLs through a *conversion component*. Finally, the participants will run these UoLs on any LMS, so the students are able to achieve the instruction/learning goals.

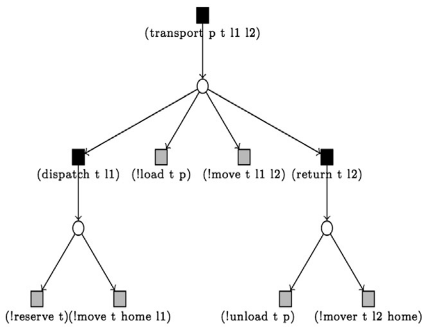
The ID for CSCL faces several challenges that must be resolved by HTN Planner "JShop2ip," including: (a) the representation of relative conditions in the ID methods, (b) queries about the information in external repositories, (c) the definition of a mechanism for group formation, and (d) the definition of changes in the current state of student model (stages of learning development).

4.1. Planning problem definition: the initial CSCL scenario

The planning problem definition has the form (defproblem ald-problem ald-domain *s*0 (*t*0)), where the initial state *s*0 is a set of logical atoms (*a*1 *a*2 ... *an*) that represent information of student model (e.g., CL experience, motivation) and of domain model (e.g., difficult, context or prerequisite of an exercise). The initial task *t*0 is an ID task which represents the element of UoL to be

```
(:method (transport ?p ?t ?l1 ?l2)
   ((available ?t)        ;; precondition
    (at ?p ?l1))
   ((dispatch ?t ?l1)     ;; sub-tasks
    (!load ?t ?p)
    (!move ?t ?l1 ?l2)
    (return ?t ?l1)))
...
(:operator (!move ?t ?l1 ?l2)
   ((at ?t ?l1))
   ((at ?t ?l1))
   ((at ?t ?l2)))
```

(a) The representation of planning domain definition



(b) The planning process

Fig. 5. Example of HTN Planning for “transport a package p from location l1 to l2.”.

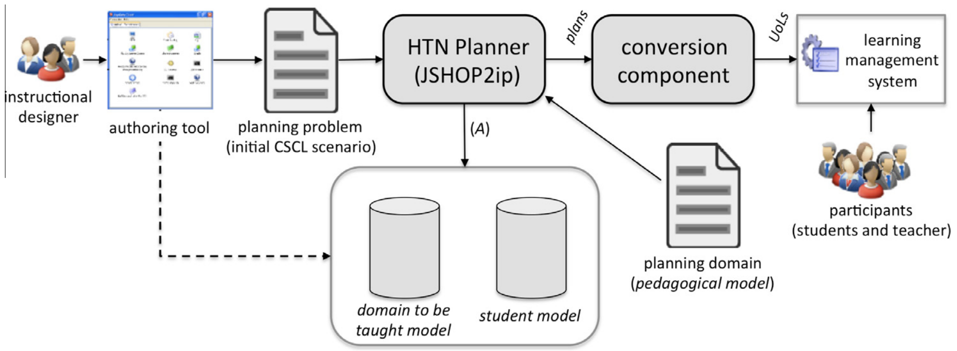

instructional designer
authoring tool
planning problem (initial CSCL scenario)
HTN Planner (JSHOP2ip)
plans
conversion component
UoLs
learning management system
(A)
domain to be taught model
student model
planning domain (pedagogical model)
participants (students and teacher)

Fig. 6. The automated ID model for CSCL as HTN planning.

obtained, and the purpose and expected benefits of this element (defined as a set of learning goals that can be achieved by students after running this element in any LMS).

4.1.1. Initial state (s0): the student model and the domain model

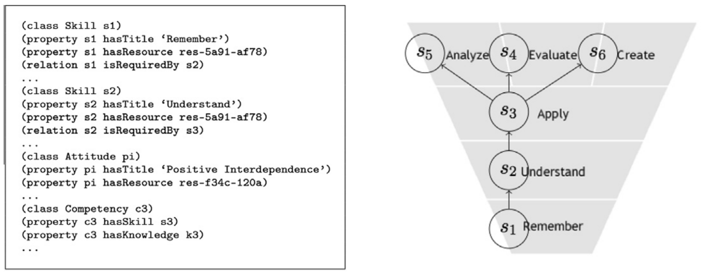
The student and the domain models are based on Sicilia (2005), Wilkinson (2001) and Melis, Faulhaber, Eichelmann, and Narciss (2008), which define competence as the multidimensional characteristics composed of skills, knowledge, and attitudes held by students. The domain model is a set of instructional resources defined into the competency structure or the knowledge structure. The student model is a set of records that overlap the domain model. Thus, in the initial state *s*0, each element ei is represented by a set of logical atoms (class *c ei*), (property *ei* hasResource *r*), (property *ei prop v*), and (relation *ei rel ej*), where “*c*” denotes the classification of element, “*r*” is the resource associated with the element, “*prop*” is property name, “*v*” is value of property, and “*rel*” specifies the relation between elements “*ei*” and “*ej*”.

The *competency structure* $CS = \langle S,A,C,RS,RA \rangle$ is composed of skills (*S*), attitudes (*A*), cognitive competencies (*C*), relations among skills (*RS*), and relations among attitudes (*RA*). The skills $s \in S$ are considered “abilities that have been acquired by training.” Some skills are of a generic nature like “discussion” or “negotiation,” but many others refer to concrete knowledge elements, such as “solve problems,” “understand,” etc. Thus, a competency cognitive $c \in C$ that defines the association between a skill (non-generic) $s \in S$ and a knowledge element *k* is defined through logical atoms (class Competeny *c*), (property *c* hasSkill *s*) and (property *c* hasKnowledge *k*). Finally, the attitudes $a \in A$ are considered “a set of complex mental state involving beliefs and feelings and values and dispositions to act in certain ways.”

Example 1. Fig. 7(a) shows the representation of a competency structure, where the skills “remember” and “understand” are extracted from categories in the cognitive domain of Bloom’s Taxonomy (Fig. 7(b)). This example also shows the representation of an attitude “positive interdependence” and a cognitive competence “apply the method of reduction to absurdity.”

The *knowledge structure* $KS = \langle K,P,R \rangle$ defines the information contained in books, summaries, and web pages. This information defines the domain model as knowledge elements (*K*), their properties (*P*), and the relations (*R*) among these elements. In this structure, we distinguish between the class element fundamental and auxiliary. A fundamental element $ki \in K$ defined through a logic atom (class Fundamental *ki*) is a central piece of information (e.g., concepts) that learners should learn during the learning process. Examples of fundamental elements are theorems, proofs, and definitions. An auxiliary element $kj \in K$ represents additional information about fundamental elements as well as examples, exercises, and problems through an logic atom (class Auxiliary *kj*). Table 1 shows the properties of knowledge elements, their data types, and their possible values. Finally, a relation $r \in R$ between two knowledge elements can be a prerequisite (isRequiredBy), an aggregation (isPartOf) or a variation (isVariantOf).

The relationships between a knowledge element $ki \in K$ of type “auxiliary” and a cognitive competency $cj \in C$ is defined through a set of properties “prerequisites” (hasPrerequisite) and “learning objectives” (hasLearningObjective). The prerequisites are the minimum competency levels that the participants must have to take advantage of an element, while the learning objectives are the competency levels that will be achieved after use of this element. Using the *Learner’s Growth Model*, a competency level cl is defined as a stage of skill development and a stage of knowledge acquisition (Inaba et al., 2003; Isotani and Mizoguchi, 2007). Thus, we



```
(class Skill s1)
(property s1 hasTitle ‘Remember’)
(property s1 hasResource res-5a91-af78)
(relation s1 isRequiredBy s2)
...
(class Skill s2)
(property s2 hasTitle ‘Understand’)
(property s2 hasResource res-5a91-af78)
(relation s2 isRequiredBy s3)
...
(class Attitude pi)
(property pi hasTitle ‘Positive Interdependence’)
(property pi hasResource res-f34c-120a)
...
(class Competency c3)
(property c3 hasSkill s3)
(property c3 hasKnowledge k3)
...
```

(a) The representation of structure in JSHOP2ip (b) Categories in the cognitive domain

Fig. 7. Competency structure that uses the cognitive domain of Bloom's taxonomy to define non-generic skills.

Table 1
Properties of a knowledge elements, their data types, and their possible values.

Property (*prop*)	Data types	Possible values (v)
hasTitle	String	
hasResource	Resource	
hasContext	Vocabulary	training, higher-education, school, other
hasDifficult	Vocabulary	very-easy, easy, medium, difficult, very-difficult
hasLearningResourceType	Vocabulary	Auxiliary: explanation, introduction, remark, conclusion, interactivity, exercise, exploration, invitation, real-world-problem, evidence, proof, demonstration, illustration, counter-example, example. Fundamental: law, theorem, law-of-nature, definition, fact, process, policy, procedure
hasPrerequisite	n-tuple (c,cl)	c is a cognitive competence and cl is a competency level
hasLearningObjective	n-tuple (c,cl)	c is a cognitive competence and cl is a competency level

defined 20 competency levels sxky, where x is a level of skill development that can be (0) nothing, (1) rough cognitive, (2) explanatory cognitive, (3) associative, or (4) autonomous; and y is a level of knowledge acquisition that can be (0) nothing, (1) accretion, (2) tuning, or (3) restructuring.

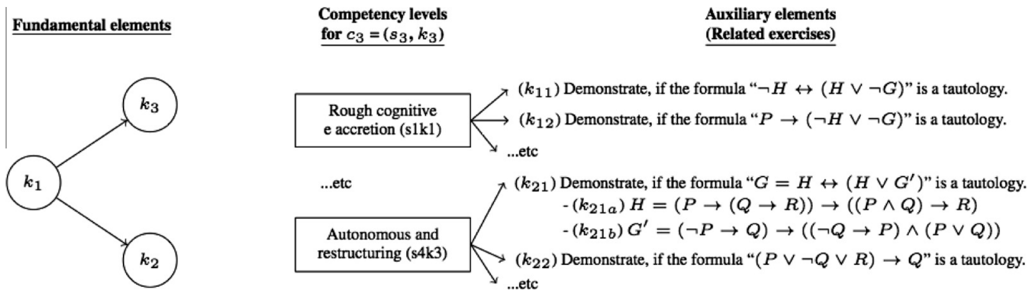
Example 2. Fig. 9 shows the representation of knowledge structures shown in Fig. 8. This figure shows the representation of a knowledge element of the type fundamental "method of reduction to absurdity" (k3), and the representation of two knowledge elements of type auxiliary "demonstrate, if the formula H is a tautology" (k21a) and "demonstrate if the formula G' is a tautology" (k21b).

The *student model* $LM = \langle L, CR, HR, PR \rangle$ is composed of learners (L), competency records (CR), history records (HR), and preference records (PR). For a learner $l \in L$, a competency record $cr \in CR$ defines the general skills $s \in S$ and attitudes $a \in A$ possessed by the learner through the logical atoms (property l hasSkill s) and (property l hasAttitude a). Also, a competency record $cr \in CR$ defines the current stage of learning development in a competency $c \in C$ through logical atoms (property l hasCompetencyLevel (c cl)). A history record $hr \in HR$ defines "what knowledge element $k \in K$ was used by a learner" through a logical atom (property l hasAlreadySeen k). A preference record $pr \in PR$ defines the characteristics of a learner through a logical atom (property l $prop$ v), where the preferences properties, their data types, and their possible values v are shown in Table 2.

Example 3. The set of logical atoms L = (class Learner l1), (property l1 hasAttitude pi), (property l1 hasCompetencyLevel (c3 s0k1)), (property l1 hasAlreadySeen k11), (property l1 hasMotivation (c3 high)) in the student model defines the information related to learner l1. Thus, the learner l1 has the attitude "positive interdependence" (pi), and the current stages of learning development "nothing and accretion" (s0k1) in the competence "apply the method of reduction to absurdity" (c3). The history record shows that the learner used the "exercise k_{11}" and the preference records shows that the learner has a "high" level of motivation to learn how to "apply the method of reduction to absurdity" (c3).

4.1.2. Initial task (t0): the initial instructional design task

In the planning problem definition, the initial ID task $t0$ defines the instructional goal as the element to be obtained and the

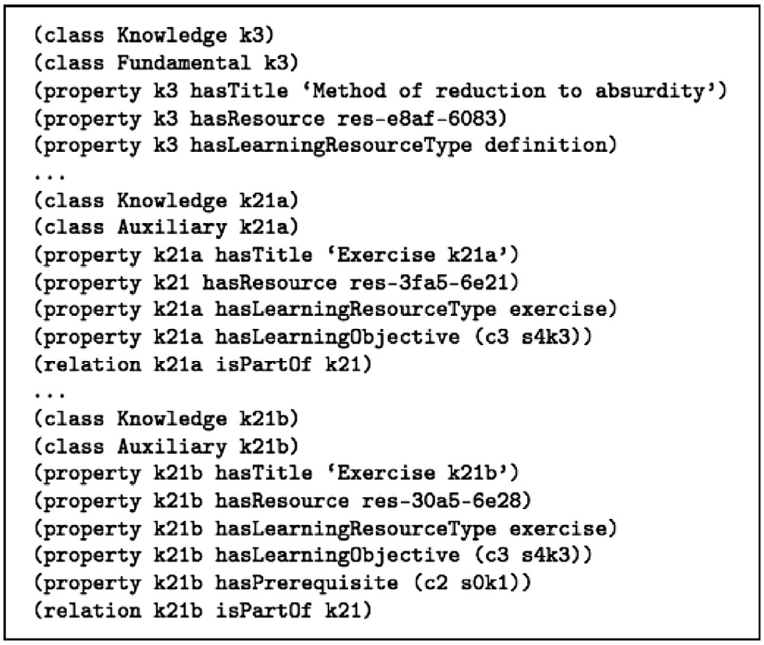


Fundamental elements
k_3
k_1
k_2
Competency levels for $c_3 = (s_3, k_3)$
Auxiliary elements (Related exercises)
Rough cognitive e accretion (s1k1)
...etc
Autonomous and restructuring (s4k3)

(k_{11}) Demonstrate, if the formula "$\neg H \leftrightarrow (H \vee \neg G)$" is a tautology.
(k_{12}) Demonstrate, if the formula "$P \rightarrow (\neg H \vee \neg G)$" is a tautology.
...etc
(k_{21}) Demonstrate, if the formula "$G = H \leftrightarrow (H \vee G')$" is a tautology.
- (k_{21a}) $H = (P \rightarrow (Q \rightarrow R)) \rightarrow ((P \wedge Q) \rightarrow R)$
- (k_{21b}) $G' = (\neg P \rightarrow Q) \rightarrow ((\neg Q \rightarrow P) \wedge (P \vee Q))$

(k_{22}) Demonstrate, if the formula "$(P \vee \neg Q \vee R) \rightarrow Q$" is a tautology.
...etc

Fig. 8. Example of knowledge structure.



```
(class Knowledge k3)
(class Fundamental k3)
(property k3 hasTitle 'Method of reduction to absurdity')
(property k3 hasResource res-e8af-6083)
(property k3 hasLearningResourceType definition)
...
(class Knowledge k21a)
(class Auxiliary k21a)
(property k21a hasTitle 'Exercise k21a')
(property k21 hasResource res-3fa5-6e21)
(property k21a hasLearningResourceType exercise)
(property k21a hasLearningObjective (c3 s4k3))
(relation k21a isPartOf k21)
...
(class Knowledge k21b)
(class Auxiliary k21b)
(property k21b hasTitle 'Exercise k21b')
(property k21b hasResource res-30a5-6e28)
(property k21b hasLearningResourceType exercise)
(property k21b hasLearningObjective (c3 s4k3))
(property k21b hasPrerequisite (c2 s0k1))
(relation k21b isPartOf k21)
```

Fig. 9. The knowledge structure representation in JShop2ip.

Table 2
The learners preferences, their data types, and their possible values.

Properties	Data types	Possible values (*val*)
hasPersonality	Vocabulary	introversion, extraversion, ambiversion
hasCLExperience	Vocabulary	very-low, low, medium, high, very-high
hasEducationalLevel	Vocabulary	training, higher-education, school, other
hasMotivation	*n*-tuple: (*c*, *cl*)	*c* is a cognitive competence and *l* is very-low, low, medium, high, or very-high
hasAnxiety	*n*-tuple: (*c*, *cl*)	*c* is a cognitive competence and *l* is very-low, low, medium, high, or very-high

learner's purpose and expected benefits through an expression of form (*t E* [*S A*] *G*), where:

- "*t*" is a task symbol that defines the instructional goal as the element to be obtained after the ID process. Table 3 shows a list of task symbols and elements to be obtained.
- "*E*" is a list of learning goals (*e*1 ... *en*) that defines the expected benefits of the elements to be obtained. Each learning goal *ei* defines the desired stage of learning development to be achieved by each student of group $gi \in G$, this learning goal is an expression of the form (*c cl*), where "*c*" is a cognitive competence and "*cl*" is a competency level.
- "*S*" is an optional list of general skills (*s*1 ... *sm*) to be acquired by all students.
- "*A*" is an optional list of attitude (*a*1 ... *am*) to be acquired by all students.
- "*G*" is a list of groups (*g*1 ... *gn*), where each *gi* is represented by a list of students (*l*1 ... *lm*).

Example 4. The initial ID task used to "create a macro-script according to IMS-LD specification for four students (with identifiers l1, l2, l3 and l4)" is represented through the initial task:

*t*0 = (createLDScript((c3 s2k3)(c3 s4k3))(ne)()((l1 l2)(l3 l4))).

One purpose of this task is the acquisition of a non-generic skill "negotiation" (ne) by all students. Another purposes of the competence "apply the method of reduction to absurdity" (c3) is to achieve the stages of learning development: "explanatory and restructuring" by students l1 and l2, and "autonomous and restructuring" by students l3 and l4.

4.2. Planning domain definition: instructional design strategies

The planning domain definition has the form (defdomain domain (*d*1 *d*2 ... *dn*)), where each item *di* is an operator, a method, or an axiom. A representation of these elements in JSHOP2ip is based on JSHOP2 (Nau et al., 2003), we restrict to provide the description of features actually used in this paper. The JSHOP2 manual (Ilghami, 2006) describes the complete set of features.

The following section describes the representation of ID strategies as hierarchical methods within the planning domain definition. This process represents these strategies through the six guidelines detailed below. The guidelines show how these ID strategies are extracted from CSCL Script Design Patterns and IMS-LD specification.

Table 3
Task symbols that can be used in the definition of initial ID tasks.

Task symbols	Description
createLDFundamentalUoL	Create a UoL according to IMS-LD specification
createLDScript	Create a macro-script according to IMS-LD specification
createLDCLScenario	Create a micro-script according to IMS-LD specification

4.2.1. First guideline: mapping of instructional design tasks as hierarchical tasks

The non-primitive ID tasks are represented as expressions (p ?goals ?groups) or (p ?skills ?attitudes ?goals ?groups), where "p" is a symbol defined through the mapping function $f: T \rightarrow P$ that associates each name of an ID task $t \in T$ with a task symbol $p \in P$.

Example 5. The ID task used to "create an individual phase according to IMS-LD" is represented as an expression (createLDIndividualPhase ?goals ?groups), where "f" defines the association between this task and a symbol createLDIndividualPhase.

A primitive ID task is represented as an operator in the domain planning definition. Table 4 shows the set of operators used to represent ID tasks and their corresponding descriptions. The operators that begin with two exclamation mark "!!" are used to define actions in the plan that will not be interpreted by conversion component (called non-translate actions).

4.2.2. Second guideline: mapping between instructional design strategies and IMS-LD elements

Each ID strategy has an associated hierarchical task of high-level to define the mapping between it and an element of IMS-LD specification. This association is defined by the scheme shown in the Fig. 10, in which the ID strategy "create*TaskName*" is mapped out with an IMS-LD element *tag* that has the identifier ?id.

Table 5 shows the mapping between an ID strategy and an IMS-LD element. This mapping is extracted from CSCL Script Design Patterns related to CL flow level and activity level. In addition, we added the modeling of patterns defined by Ullrich (2008). These patterns formalize the descriptions of courses through the definition of structured sections. For example, the "discover" pattern structures a course through sections "introduction," "development," "practice," and "show relations."

The operators (!startLDElement *tag*-ref ((ref ?id))) and (!endLDElement *tag*-ref) in the mapping between a ID strategy and an IMS-LD element (shown in the Fig. 9) are optional – we can use both or none. Each ID task of set *InstructionalDesignTask*1... *InstructionalDesignTaskn* can be an operator or any of basic ID task (detailed in Table 6).

Example 6. Fig. 11 shows the mapping between the ID strategy used to "create a individual phase" (createIndividualPhase) and the element act of IMS-LD specification. This strategy was extracted from CLFPs "jigsaw" and "pyramid" and its mapping includes the operator !!changeCLGrouping (line 4) to avoid group formation, thereby forcing individual work in learning sessions.

To define changes in the current stage of learning development of students (challenge "d" in ID for CSCL), we replace the operator !startLDElement with the hierarchical task startLDElement! in the scheme definition (shown in Fig. 9) that defines the mapping between an instructional design strategy and an IMS-LD element. The hierarchical task startLDElement! detailed in Fig. 12 uses the operator !!changeIndGoals (line 4) to change individual goals of students in the current IMS-LD element ?id. In association with the operator !endLDElement, the changes in the current stage of learning development of students are defined "how the substitution of current competency levels of students by levels defined in each individual goal."

Example 7. Suppose that the initial state $s0$ is (property *l*1 hasCompetencyLevel (*c*1 s1k0)), (property l3 hasCompetencyLevel (*c*1 s1k1)) and the initial task $t0$ is (createLDDiscussionSession ((c1 s1k2) (c1 s1k1)) ((l1) (l3 l4))). Using the ID strategy shown in Fig. 13 that defines the mapping between this strategy and the element "learning-activity, " we obtain the plan P, where

$$\begin{aligned}
\mathrm{P} = (&(\text{!tartLDElement learning} - \text{activity}((\text{identifier la} - \text{a23f} - \text{df23}))\\
&((\text{Discussion Session})\text{la} - \text{a23f} - \text{df23}((\text{c1})\text{s1k2})((\text{c1 s1k1}))((\text{l1})(\text{l3 l4}))))\\
&(\text{!!changeIndGoals}((\text{l1}(\text{c1s1k0s1k2}))(\text{l3}(\text{c1 s1k1s1k1}))(\text{l4}(\text{c1 s0k0s1k1}))))\\
&\ldots\\
&(\text{!endLDElement learning} - \text{activity la} - \text{a23f} - \text{df23}))
\end{aligned}$$

The result state *sn* to apply the plan P in $s0$ is (property l1 hasCompetencyLevel (c1 s1k2)), (property l3 hasCompetencyLevel (c1 s1k1)), (property l4 hasCompetencyLevel (c1 s1k1)).

4.2.3. Third guideline: representation of instructional design rules as logical preconditions and relative conditions in hierarchical methods

The guidelines in each ID method that define the learner's proper selection is a set of conditions called ID rule. These rules explain the rationale for their selection through evaluating the purpose and expected benefits of each element according to CSCL Script Design Patterns. The ID rules are represented as logical preconditions and relative conditions in hierarchical methods using: general purpose axioms; axioms for queries in domain to be taught and student models; and ID axioms.

To solve the challenge "a" (in the ID for CSCL), a method (:method *h pre* [*rel*] *t*) in JSHOP2ip has a special type of condition, named relative conditions [*rel*]. These conditions do not need to be satisfied in the process of planning as an logical precondition "*pre*".

Table 4
The primitive instructional design tasks as operators.

Operator	Description
`(!startLDElement ?tag ?params·(?t ?id ?goals ?groups))`	Create a start tag with identifier ?id and parameters `?params` according to IMS-LD. The variable symbol `?t` is a list of terms that defines the classification of element `?id`, while the variable `?goals` defines the expected benefits to be acquired by students `?groups`
`(!endLDElement ?tag ?id)`	Create an end tag for an element with identifier `?id` according to IMS-LD
`(!!changeIndGoals ?indGoals)`	Change individual learning goals for the students
`(!!changeCurrentLDElement ?elements)`	Change the current IMS-LD element to be obtained
`(!!changeLearningResourceType ?types)`	Change the learning resource types to be used in the definition of elements to be obtained
`(!!changeFundCompetency ?comp)`	Change the knowledge fundamental to be learn through modifying the competence
`(!!changeCLGrouping (?t ?sgPs)`	Change informations about the group formation
`(!addUserToRole ?user ?role)`	Assign a role to an user
`(!removeUserFromRole ?user ?role)`	Remove an user from a role
`(!addUserToGroup ?user ?group)`	Add an user to a group
`(!removeUserFromGroup ?user ?group)`	Remove an user from a group
`(!text ?types ?fparams ?sparams)`	Insert a plain-text, where `?types` is a list that defines the source of text
`(!insertElement ?e ?learners)`	Change the history record of learners, set the element `?e` as already seen
`(!insertResource ?r ?attribs)`	Create a tag "<resource>" for ?r with attributes `?attribs` according to IMS-CP

```
(:method (createLDTaskName ?goals [?skill ?attits] ?groups)
  ((assign ?id (call GetUUID tag)))
  ((!startLDElement tag-ref ((ref ?id)))
   (!startLDElement tag ((identifier ?id) params) ((Class) ?id ?goals [?skill ?attits] ?groups))
   InstructionalDesignTask_1
   ...
   (createTaskName ?goals ?groups)
   ...
   InstructionalDesignTask_n
   (!endLDElement tag ?id)
   (!endLDElement tag-ref)))
```

Fig. 10. The scheme of mapping between an instructional design strategy and an IMS-LD element.

Table 5
The mapping between instructional design strategies and IMS-LD elements.

Task symbols	Description (<tag>)
(0) Mapping using patterns related to pedagogical level	
createLDFundamentalUoL	Create a UoL as an element < *learning-design*>
createLDFundamentalScript	Create a pedagogical scenario as an element < *play*>
createLD*Name*Phase	Create a section in a scenario as an element < *act*>
createLD*Name*GroupActivity	Create an association between learners and learning sessions as an element < *role-part*>
createLD*Name*Sessions	Create a set of learning sessions as an element < *activity-structure*>
createLD*Name*Session	Create a learning session as an element < *learning-activity* > ou < *unit-of-learning-ref*>
(1) Mapping using patterns related to CL flow level	
createLD*Name*Script	Create a macro-script as an element < *play*>
createLD*Name*Phase	Create a phase in a macro-script as an element < *act*>
createLD*Name*GroupActivity	Create a group activity as an element < *role-part*>
createLD*Name*Sessions	Create a set of learning sessions as an element < *activity-structure*>
createLD*Name*Session	Create a learning session as an element < *learning-activity* > or < *unit-of-learning-ref*>
(2) Mapping using patterns related to activity level	
createLD*Name*CLScenario	Create a micro-script as an element < *play*>
createLD*Name*Phase	Create a phase in a micro-script as an element < *act*>
createLD*Name*Strategy	Create a learning strategy as an element < *role-part*>
createLD*Name*Interactions	Create a cyclical or directed interaction as an element < *activity-structure*>
createLD*Name*ILEvent	Create an IL event as an element < *learning-activity*>
createLD*Name*ILEventDescription	Create an action in IL event defined as an element < *activity-description*>

These conditions increase or decrease the preference on selection of a method through the value of applicability. When more than one method can be applied in an existing state, the value of applicability defines the selection order of methods, so the first method to be selected in the planning process is the one that has the highest value of applicability.

Example 8. The representation of ID rules associated with the ID method that defines the creation of a macro-script using the CLFP "Jigsaw" is shown in Fig. 14. These rules evaluate "what the problem to be solve can be divided into sub-problems" (lines 2–4) and "what the number of students is enough to solve the sub-problems" (lines 5–6). Within these rules, there are two relative conditions that evaluate the CL experiences of students and the values of elements that will be obtained. The preference on selection is increased if "the majority of students have high level of CL experience" (lines 9–11), and "one purpose is to promote the positive interdependence" (lines 12–14).

To enable the reasoning about the informations of "student model" and "domain model" contained in external repositories (challenge "b" in the ID for CSCL), the axioms for queries in "student model" and "domain model" use the following call terms: (i) (call GetType ?e) to obtain the classifications of element ?e; (ii) (call GetRelated ?e ?rel) to obtain a list of elements which are connected to the element ?e by the relation ?rel; (iii) (call GetPropertyValue ?e ?prop [?d]) to obtain the value of a property ?prop of element ?e that is optional related with ?d; and (iv) (call GetElements ?q) to obtain a list of elements that fulfill the given mediator query ?q.

For example, the call term GetPropertyValue is used in the axiom getPropertyValue shown in Fig. 15 to obtain the value of a property firstly in the current state (line 2), and next in external repositories (line 3). This axiom is used in the representation of ID rules shown in the Fig. 13 to obtain an auxiliary element ?a related with the competence ?c (line 3).

The mechanism of group formation is defined through the call term (call GetCLGrouping ?indGoals) to solve the challenge "c" (in the ID for CSCL). This call term is used only in ID strategies that are mapped to an IMS-LD element " < act > ", because the group formation is defined only in this strategy through the definition of elements " < role-part > ". For example, the ID strategy used to "create a phase of practice with exercises" shown in Fig. 16 use the call term GetCLGrouping (line 3) to obtain the groups in the symbol variable ?clgrouping.

4.2.4. Fourth guideline: definition of mandatory and optional instructional design tasks

In the definition of ID strategies for CSCL, there exist a number of ID tasks that should be achieved if possible, but failing to do so should not cause backtracking; these tasks are called optional ID tasks. Other ID tasks are mandatory, and these are defined as hierarchical tasks through the suffix "!". These ID tasks cause backtracking if no method can be applied in the current state.

Therefore, in the representation of the ID strategy, there is an equivalent option task for all critical ID tasks. The scheme shown in Fig. 17 is used to represent mandatory and optional tasks. The first method encapsulates the mandatory ID task "create*TaskName*!"

in an optional ID task "create*TaskName*." The second method is used to define a mandatory ID task, and in case this method cannot be applied in the current state, the "fallback" method is applied.

Example 9. Fig. 18 shows the representation of mandatory and optional ID tasks "create a description for IL event that show solution" (createShowSolutionILEventDescription). The mandatory task (lines 5–9) performs the search of descriptions by the query axiom getElement, while the *fall-back* method (lines 11–15) performs the creation of description through the call term (call BuildElement ...).

4.2.5. Fifth guideline: definition of repetitive task decomposition method into instructional design strategies

In the representation of many of ID strategies as hierarchical methods, some ID tasks are further delineated into a given number of repetitive ID sub-tasks. This process is called repetitive task decomposition method, and it uses a set of term list to determinate the ID sub-tasks.

Example 10. Fig. 19 shows the definition of a repetitive task decomposition method used to break down an ID task into sub-tasks "create a individual group activity using the IMS-LD specification" (createLDIndividualGroupActivity). This hierarchical method decomposes the task distributeIndGroupActivityByComps through two list terms ?comps and ?groups, these terms are used to define the variable symbols ?c and ?g.

4.2.6. Sixth guideline: definition of relationships among CSCL Script Design Patterns

According to the relationship "completes" defined in the hierarchical structure of CSCL Script Design Patterns (shown in Fig. 1), a CLFP can be completed by another CLFP or a pattern at activity level. This means that two macro-scripts can be combined through the substitution of activity in one learning session by another macro-script. It also means that one micro-script can be integrated into another macro-script through substituting of an activity in one learning session by one micro-script. In this sense, the relationship "completes" is defined in all ID strategies that are used to "create an activity in any learning session" (create*ActivityName*) through the schema shown in Fig. 20. In this schema, the substitution is defined in the mandatory ID task through the hierarchical task(createLDScriptCLScenario!). If the substitution fails then the fall-back method defines the activity as an environment and a description.

The hierarchical task (createLDScriptCLScenario!) shown in Fig. 21 defines the substitution of activity in one learning session. First, this task attempts to create a macro-script using any CLFPs through the mandatory task createLDScript! (lines 6–8). Next, if the mandatory task fails and none of CLFPs can be applied, the substitution attempts to create a micro-script through the ID task createLDCLScenario! (lines 13) that is defined in the *fall-back* method (lines 11–13).

Example 11. Fig. 22 shows the definition of the relationship "completes" in the ID strategy used to "create an activity for the discussion session." This ID strategy was extracted from the definition of the learning session in CLFPs "jigsaw" and "pyramid."

The relationships "is alternative to" and "specializes" are naturally represented through the logical preconditions in methods used to "create a macro-script using CLFPs" *(createLDScript!)* and "create a micro-script using patterns at activity level" *(createLDCLScenario!)*.

Example 12. Fig. 23 shows the relationship "is alternative to" between two CLFPs. In the strategy used to "create a macro-script using the jigsaw pattern" (lines 1–15), the logical preconditions consist of the comparison of "the number of knowledge sub-elements associated with the competence ?c" (?nSubs, lines 2–4) and "the number of learners" (?nLearners, line 5). According this pattern, the number of learners must be greater than double the number of sub-elements, and the number of sub-elements must be greater than 1 (line 6). According to "pyramid," the logical precondition used to "create a macro-script" (lines 17–30) using this pattern consist of "the difficulty value of knowledge element associated with the competence ?c, this value must be very-difficult" (lines 18–20) and "the number of learners ?nLearners must be greater than 8" (line 21).

The relationship "complements" among CSCL Script Design Patterns are not defined in the planning domain definition. In our approach, these relationships are defined manually through the authoring tool. Thus, before performing the conversion of a plan named "main" into UoLs, the instructional designer defines the concatenation among CSCL script at the same level (macro-scripts or micro-scripts) through the definition of a new planning problem definition. After the planning process, the newly obtained plan named "sub plan" is added into the "main" plan.

4.3. Conversion component: the translation of instructional design actions into UoLs

Once the plan is obtained by the JShop2ip planner, the conversion components performs their translation into files "*imsmanifest.xml*" that represent the CSCL scripts using the IMS-LD specification. The manifest files are packaged into a set of ZIP packages with the necessary resources using the IMS-CP specification. Thus, the function "*convert*(*hi*)" uses the plan p = (*h*1 *h*2 ... *hn*) to perform:

- the conversion of each action (!startLDElement *tagname* [((*attrib*1 *v*1) ... (*attribn vn*))) into a start tag < imsld:*tagname attrib*1 = "*v*1" ... *attribn* = "*vn*">;
- the conversion of each action (!endLDElement *tag* [*id*]) into a end tag </imsld:*tag*>;
- the conversion of each action (!text (*t*0 ... *tn*) (*f*0 ... *fm*1) (*s*0 ... *sm*2)) into a text *t*0...*tn* with substitution of {f0} ... {fm1}, {s0} ... {*sm*2} by terms *t*0 ... *tn*, *s*0 ... *sm*2; and

Table 6
The basic instructional design tasks used in the mapping between ID strategies and IMS-LD elements.

Basic task	Description
`(createLDTitle ?t ?goals [?e ?texts])`	Create a label for an IMS-LD element using how parameters a list of elements `?e` and a list of constant symbols `?texts`. A list `?t` defines the source of text
`(createLDItem ?e ?learners)`	Create an IMS-LD element < item > to insert the resource related with the element `?e` defined in the student model or the domain model
`(createLDInstructItem ?e ?learners)`	Create an IMS-LD element < item > in an instructional event for students `?learners`
`(createLDLearningItem ?e ?learners)`	Create an IMS-LD element < item > in a learning event for students `?learners`

```
(:method (createLDIndividualPhase ?goals ?groups)
   ((assign ?id (call GetUUID as)))
   ((!startLDElement act ((identifier ?id)) ((Individual Phase) ?id () ?groups))
    (!!changeCLGrouping ())
    (createLDTitle (Individual Phase) ?goals)
    (createIndividualPhase ?goals ?groups)
    (!endLDElement act ?id)))
```

Fig. 11. Mapping between the strategy "create individual phase" and the element "`act`".

```
(:method (startLDElement! ?tag ?params (?types ?id ?goals ?groups))
   ((getIndGoals ?indGoals ?goals ?groups))
   ((!startLDElement ?tag ?params (?types ?id ?goals ?groups))
    (!!changeIndGoals ?indGoals)))
```

Fig. 12. Task `startLDElement!` that creates a start tag and changes stages of learning development.

```
(:method (createLDDiscussionSession ?goals ?groups)
  ((assign ?id (call GetUUID la)))
  ((!startLDElement learning-activity-ref ((ref ?id)))
   (startLDElement! learning-activity ((identifier ?id)) ((Discussion Session) ?id ?goals ?groups))
   (createDiscussionSession ?goals ?groups)
   (!endLDElement learning-activity ?id)
   (!endLDElement learning-activity-ref)))
```

Fig. 13. Mapping between the strategy "create discussion session" and the element "`learning-activity`".

```
(:method (createLDScript! ?goals ?groups)
  ((getCompFromGoals ?c ?goals) (hasKnowledgeType ?c Auxiliary)
   (getPropertyValue ?a ?c hasKnowledge)
   (getRelateds ?subs (?a) 1 inverseIsPartOf) (length ?nSubs ?subs) (call > ?nSubs 1)
   (length ?nLearners (call ConcatList ?groups))
   (call >= ?nLearners (call * 2 ?nroSubs))
   (getElement ?e ((class CurrentLDElement)))
   (getRelateds ?lds (?e) -1 isPartOf ((class Script) (class Jigsaw))) (same ?lds ()))
  (:relative 3 ((assign ?learners (call ConcatList ?groups)) (length ?nroLearners ?learners)
                (filterByQuery ?experts ?learners ((property hasCLExperience (high very-high))))
                (length ?nExperts ?experts) (call > ?nExperts (call / ?nLearners 2))))
  (:relative 1 ((getRelateds ?pis (pi) -1 isVariantOf)
                (getElement ?e ((class CurrentLDElement))) (getPropertyValue ?u ?e hasCurrentUoL)
                (getPropertyValue ?a1 ?u hasAttitude) (or (same ?a1 pi) (exist ?a1 ?pis)))))
  ((createLDJigsawScript ?goals ?groups)))
```

Fig. 14. The representation of instructional design rules into the method "create a macro-script using jigsaw".

```
(:- (getPropertyValue ?result ?e ?property)
    (:first (property ?e ?property ?result))
    ((assignIterator ?result (call GetPropertyValue ?e ?property))))
```

Fig. 15. The axiom to get the value of an property in the current state and external repositories.

```
(:method (createPracticeWithExercisePhase ?goals ?groups)
   ((getIndGoals ?indGoals ?goals ?groups)
    (assign ?clgrouping (call GetCLGrouping ?indGoals)))
   ((distributePracticeWithExerciseGroupActivityByCLGroups ?clgrouping)))
```

Fig. 16. The instructional design strategy used to "create a phase of practice with exercises".

```
(:method (createTaskName ?goals [?skill ?attitudes] ?groups)
   ()
   ((createTaskName! ?goals [?skill ?attitudes] ?groups)))

;; mandatory
(:method (createTaskName! ?goals [?skill ?attitudes] ?groups)
   ()
   ())

;; fall-back
(:method (createTaskName ?goals [?skill ?attitudes] ?groups)
   ()
   ())
```

Fig. 17. The scheme for definition of mandatory and optional tasks.

```
(:method (createShowSolutionILEventDescription ?goals ?groups)
   ()
   ((createShowSolutionILEventDescription! ?goals ?groups)))

(:method (createShowSolutionILEventDescription! ?goals (?instructors ?learners))
   ((getElement ?iact ((class Explanation) (class ILEventDescription))
    (getElement ?lact ((class RecExplanation) (class ILEventDescription))))
   ((createLDInstructItem ?iact ?instructors)
    (createLDLearningItem ?lact ?learners)))

(:method (createShowSolutionILEventDescription ?goals (?instructors ?learners))
   ((assign ?iact (call BuildElement ((class Explanation) (class ILEventDescription))))
    (assign ?lact (call BuildElement ((class RecExplanation) (class ILEventDescription)))))
   ((createLDInstructItem ?iact ?instructors)
    (createLDLearningItem ?lact ?learners)))
```

Fig. 18. The mandatory and optional ID tasks "create an description for IL event that show solution".

```
(:method (distributeIndGroupActivityByComps () ()) () ())

(:method (distributeIndGroupActivityByComps (?c . ?comps) (?g . ?groups))
   ()
   ((createLDIndividualGroupActivity ((?c s0k0)) (?g))
    (distributeIndGroupActivityByComps ?comps ?groups)))
```

Fig. 19. The decomposition method used to define the distribution of task `createLDIndividualGroupActivity`.

```
(:method (createActivityName ?goals ?groups)
   ()
   ((createActivityName! ?goals ?groups)))

;; mandatory
(:method (createActivityName! ?goals ?groups)
   ()
   ((createLDScriptCLScenario! ?goals ?groups)))

;; fall-back
(:method (createActivityName ?goals ?groups)
   ()
   ((createLDActivityNameEnvironment ?goals ?groups)
    (createLDActivityNameSessionDescription ?goals ?groups)))
```

Fig. 20. The scheme to define the relationship "completes" among CSCL script design patterns.

```
(:method (createLDScriptCLScenario! ?goals ?groups)
   ()
   ((createLDScriptCLScenario!! ?goals ?groups)))

;; mandatory
(:method (createLDScriptCLScenario!! ?goals ?groups)
   ()
   ((createLDScript! ?goals ?groups)))

;; fall-back
(:method (createLDScriptCLScenario! ?goals ?groups)
   ()
   ((createLDCLScenario! ?goals ?groups)))
```

Fig. 21. The hierarchical methods associated with the task `createLDScriptCLScenario!`.

```
(:method (createDiscussionActivity ?goals ?groups)
   ()
   ((createDiscussionActivity! ?goals ?groups)))

;; mandatory
(:method (createDiscussionActivity! ?goals ?groups)
   ()
   ((createLDScriptCLScenario! ?goals ?groups)))

;; fall-back
(:method (createDiscussionActivity ?goals ?groups)
   ()
   ((createLDDiscussionEnvironment ?goals ?groups)
    (createLDDiscussionSessionDescription ?goals ?groups)))
```

Fig. 22. The definition of relationship "competes" that is defined in the activity of learning session "discussion".

- the conversion of each action (!insertResource *id* ((type *t*) (href *h*))) into a IMS-CP element < imscp:resource identifier = "*id*" type = "*t*" identifier-ref = "*h*" />.

Example 13. The plan P and function *convert*(*hi*) shown below define the generation of IMS-LD and IMS-CP elements shown in Fig. 24.

```
P = ((!startLDElement play((identifier play – 29241d03 – c533) (isvisible true)))
  ...
  (!startLDElement act((identifier act – ace2f2d3 – d348)))
  (!startLDElement title)
  (!text (Individual Phase)()())
  (!endLDElement title)
  (!startLDElement role – part)
  (!startLDElement role – ref ((ref rp – 84516a75 – a565)))
  (!endLDElement role – ref)
  ...
  (!endLDElement role – part)
  (!endLDElement act)
  (!endLDElement play))
```

5. Research application

To evaluate the validity of our approach, we developed a CSCL courseware generator employing the reference model detailed in previous section. This courseware generator obtains personalized UoLs for CSCL. These units are adapted from the students' characteristics through of a set of plans of sections, plans of practice sessions, plans of learning sessions, and plans of interactions. The plans of sections are obtained through the sequencing of learning goals, the selection of a *pattern related to pedagogical level* for each learning goal, and the definition of sections (e.g., introduction, development, and practice) using this pattern. The plans of practice sessions (for each practice section) are obtained through the group formation, and the definition of a sequence of practice sessions using this group formation. The plans of learning sessions for each practice session are obtained through the selection of a proper auxiliary knowledge element, the selection of a *pattern related to CL flow level*, and the definition of learning sessions using this pattern. The plans of interactions for each learning session are obtained through the selection of a *pattern related to activity level* that define the communication model to be applied, and the definition of directed and cyclical interactions as IL events using this pattern.

We modeled four pattern related to pedagogical levels detailed on the *website* http://www.ime.usp.br/~geiser/dissertacao/scenarios/. We modeled three patterns related to CL flow level with four strategies for learning session, eight strategies for group activity, and ten strategies for phase detailed on the *website* http://www.ime.usp.br/~geiser/dissertacao/macrorroteiros/. We also modeled six patterns related to activity level with twelve strategies for IL event, sixteen strategies for cyclical and directed interactions, six strategies for interactions patterns, nine strategies for learning strategies, and ten strategies for instructional and learning roles detailed on the *website* http://www.ime.usp.br/~geiser/dissertacao/microrroteiros/.

```
(:method (createLDScript! ?goals ?groups)
  ((getCompFromGoals ?c ?goals) (hasKnowledgeType ?c Auxiliary)
   (getPropertyValue ?a ?c hasKnowledge)
   (getRelateds ?subs (?a) 1 inverseIsPartOf) (length ?nSubs ?subs) (call > ?nSubs 1)
   (length ?nLearners (call ConcatList ?groups))
   (call >= ?nLearners (call * 2 ?nroSubs))
   (getElement ?e ((class CurrentLDElement)))
   (getRelateds ?lds (?e) -1 isPartOf ((class Script) (class Jigsaw))) (same ?lds ()))
  (:relative 3 ((assign ?learners (call ConcatList ?groups)) (length ?nroLearners ?learners)
                (filterByQuery ?experts ?learners ((property hasCLExperience (high very-high))))
                (length ?nExperts ?experts) (call > ?nExperts (call / ?nLearners 2))))
  (:relative 1 ((getRelateds ?pis (pi) -1 isVariantOf)
                (getElement ?e ((class CurrentLDElement))) (getPropertyValue ?u ?e hasCurrentUoL)
                (getPropertyValue ?a1 ?u hasAttitude) (or (same ?a1 pi) (exist ?a1 ?pis)))))
  ((createLDJigsawScript ?goals ?groups)))

(:method (createLDScript! ?goals ?groups)
  ((getCompFromGoals ?c ?goals) (hasKnowledgeType ?c Auxiliary)
   (getPropertyValue ?a ?c hasKnowledge)
   (getPropertyValues ?ds ?a hasDifficult) (exist very-difficult ?ds)
   (length ?nLearners (call ConcatList ?groups)) (call >= ?nLearners 8)
   (getElement ?e ((class CurrentLDElement)))
   (getRelateds ?lds (?e) -1 isPartOf ((class Script) (class Pyramid))) (same ?lds ()))
  (:relative 3 ((assign ?learners (call ConcatList ?groups)) (length ?nLearners ?learners)
                (filterByQuery ?experts ?learners ((property hasCLExperience (medium low))))
                (length ?nExperts ?experts) (call > ?nExperts (call / ?nLearners 2))))
  (:relative 1 ((getRelateds ?pis (pi) -1 isVariantOf)
                (getElement ?e ((class CurrentLDElement))) (getPropertyValue ?u ?e hasCurrentUoL)
                (getPropertyValue ?a1 ?u hasAttitude) (or (same ?a1 pi) (exist ?a1 ?pis))))
  ((createLDPyramidScript ?goals ?groups)))
```

Fig. 23. Representation of relationship "is alternative to" between "jigsaw" and "pyramid" pattern.

The CSCL courseware generator was developed as a Web Service which will be used to support the instructional planning in an Adaptive and Intelligent Educational System for CSCL (e.g., ITSs and authoring tools). We developed an authoring tool of UoLs, named ALD (*Automated Learning Design*). The ALD employs the services provided by the courseware generator to obtain UoLs that are adapted the students' characteristics.

Fig. 25 shows the user interfaces that allow definition of an initial CL Scenario. Fig. 25(a) shows the definition of an initial ID task. The editor uses a set of user interfaces shown in Fig. 25(b) to define the information related to student model and domain model as a set of forms. The file tree (on the left menu of the form used to define an initial CL Scenario shown in Fig. 25(a) shows a set of elements that are obtained after the ID process.

5.1. Formulation of planning problem

Using the ALD tool, the domain model in the initial state *s*0 is defined through a competency structure shown in Example 1, and a knowledge structure shown in Example 2. The student model in the initial state is defined through the information shown in Fig. 26.

```
<imsld:act identifier="act-ace2f2d3-d348">
    <imsld:title>Individual Phase</imsld:title>
    <imsld:role-part>
        <imsld:title>Individual Task</imsld:title>
        <imsld:role-ref ref="rp-84516a75-a565" />
        ...
    </imsld:role-part>
</imsld:act>
```

Fig. 24. The obtained IMS-LD and IMS-CP elements by plan $\mathbb{P}$ and function *convert*(*hi*).

The initial ID task *t*0 is defined as "create of a UoL according to IMS-LD specification" with a learning goal "apply the method of reduction to absurdity" (c3) into "autonomous and restructuring" level and an attitude "positive interdependence" (pi) for eight students (with identifier "l1, l2 . . ., l8") is represented as the hierarchical task:

$$t0 = (\text{createLDFundamentalUoL } ((\text{c3 s4k3}))()(\text{pi}) \times ((\text{l1 l2 l3 l4 l5 l6 l7 l8}))).$$

5.2. Results of planning process

Using this formulation of the planning problem, we obtain the results shown in Fig. 27 which consists of a plan of sections with four acts "introduction," "development," "practice," and "show relations." The section "introduction" consists of two sessions "show problem" (la-1) and "illustrate example" (la-2). The development section consists of one session "show definition" (la-3). The practice section consists of one session "solve exercises" (la-4). The show relations section consists of one session "show connections" (la-5).

The session "solve exercises" (la-4) in the section "practice" is defined as a plan of practice sessions of three phases that enables learners to achieve the cognitive competence *c*3 at level s4k3. In the first phase (act 01), Group 2 (formed by learners *l*1 and *l*2) performs the session la-6 to practice the competence *c*3 at level s4k3, Group 3 (formed by learners *l*3 and *l*4) performs the session la-7 to practice the competence *c*3 at level s4k2, and Group 4 (formed by learners *l*5, *l*6, *l*7 and *l*8) performs the session la-8 to practice the competence *c*3 at levels s3k1 and s3k2. In the second phase (act 02), Group 5 (formed by learners *l*1, *l*2, *l*3 and *l*4) performs the session la-9 to practice the competence *c*3 at level s4k3, Group 6 (formed by learners *l*5 and *l*6) performs the session la-10 to practice the competence *c*3 at level s4k2, and Group 7 (formed by learners *l*7 and *l*8) performs session la-11 to practice the competence *c*3 at level s3k1. In the last phase (act 03), Group 8 (formed by learners *l*1, *l*2, ... and *l*8) performs the session la-12 to practice the competence *c*3 at level s4k3.

The practice session "practice *c*3 at level s4k3" (la-12) is defined as the selection of exercise *k*21 "demonstration of formula *G*," and the definition of a plan of learning sessions using the pattern "*jigsaw*." In the individual phase (act 01), Group 9 (formed by learners *l*1, *l*2, *l*3 and *l*4) performs the session la-13 to solve the exercise *k*21*a*, and Group 10 (formed by learners *l*5, *l*6, *l*7 and *l*8) performs the session la-14 to solve the exercise *k*21*b*. In the expert phase (act 02), Group 9 performs the learning session la-15 "specialization of exercise *k*21*a*," and Group 10 performs the learning session la-16 "specialization of exercise *k*21*b*." In the *jigsaw* phase (act 03), Group 11 (formed by *l*1, *l*2, ... and *l*8) performs the learning session la-17 "demonstration of formula *G*".

The learning session "demonstration of formula *G*" (la-17) is defined as a plan of interactions using the pattern "*distributed cognition*." This plan defines a micro-script through the definition of roles "instructional full-participant" (for students *l*1, *l*2, *l*3 and *l*4) and "learner full-participants" (for students *l*5, *l*6, *l*7 and *l*8). In the cyclical interaction as-1, the plan defines two IL events la-18 and la-19. In the IL event la-18, the action "demonstration" is defined for student with role "instructional full-participants," and the action "obs. demonstration" is defined for student with role "learner full-participants". In the IL event la-19, the action "req. opinion" is defined for students with role "instructional full-participants," and the action "exp. opinion" is defined for student with role "learner full-participants."

5.3. Other results of the planning process

Our approach enables the generation of UoLs that are adapted to different inputs. The adaptations are obtained through setting up information values in the student model.

5.3.1. Another section plan

If we change the competency levels of students *l*7 and *l*8 (at level s3k1), the plan of sections shown in Fig. 28 is obtained through the application of the pattern "rehearse." The plan of sections consists of five sections: "develop," "show connections," "practice," "illustrate," and "practice". Other patterns related to pedagogical level that can be applied to the initial task *t*0 are "guided-tour" and "train."

5.3.2. Another practice session plan

Fig. 29 shows a plan of a practice session that is obtained through the application of another way to form groups. In session la-4 of scenario "discover" (shown in the Fig. 27), this grouping consists of four practice phases that enables learners to achieve the cognitive competence *c*3 at level s4k3.

In the first phase (act 01), Group 2 (formed by learners *l*1, *l*2, *l*3 and *l*4) performs the session la-6 to practice the competence *c*3 at level s4k3 and s3k3 (by cog. flexibility); Group 3 (formed by learners *l*5, *l*6, *l*7 and *l*8) performs the session la-7 to practice the competence *c*3 at level s3k2 and s3k1 (by peer-tutoring); Group 4 (formed by learners *l*3, *l*4, *l*5 and *l*6) performs the session la-8 to practice the competence *c*3 at level s3k3 and s4k2 (by cog. apprenticeship), and the session la-10 to practice the competence *c*3 at level s4k3 (by distribute cognition); Group 5 (formed by learners *l*7 and *l*8) performs the session la-9 to practice the competence *c*3 at level s3k2 (by anchored instruction); Group 6 (formed by learners *l*1, *l*2, *l*7 and *l*8) performs the session la-11 to practice the competence *c*3 at level s4k3 and s3k3 (by distributed cognition); and Group 7 (formed by student *l*1, *l*2, ..., *l*8) performs the session la-12 to practice the competence *c*3 at level s4k3 (by distributed cognition).

This method of forming groups uses individual goals and learning theories, but there are many others ways to form groups (e.g., free group formation, controlled group formation or a grouping tool), thus the figure shows others plans of learning sections (p2*a*, p2*b*, ...) that can be applied to session la-4 (CSCL script).

5.3.3. Another learning session plan

Changing the history and preference records of students *l*3, *l*5 and *l*7 to values shown in Fig. 30, we obtain the definition of learning session "practice *c*3 at level s4k3" (la-12) through the application of the pattern "*pyramid*" in the plan of learning sessions p2*a* (shown in Fig. 27). The result of the planning process consists of three phases of "discussion."

In the first phase (act 01), Group 9 (formed by *l*1 and *l*2), Group 10 (formed by *l*3 and *l*4), Group 11 (formed by *l*5 and *l*6) and Group 12 (formed by *l*7 and *l*8) perform the solution of exercise *k*22 in the learning sessions la-13, la-14, la-15 and la-16. In the second phase (act 02), Group 13 (formed by *l*1, *l*2, *l*3 and *l*4) and Group 14 (formed by *l*5, *l*6, *l*7 and *l*8) perform the solution of exercise *k*22 in the learning sessions la-17 and la-19. In the third phase (act 03), Group 15 (formed by *l*1, ..., and *l*8) performs the solution of exercise *k*22 in the learning session la-19. Others CSCL scripts patterns that can be applied to the learning session la-12 are "jigsaw" and "distributed cognition." Because most learners have low levels of CL experience and the selected exercise (*k*23) can not be divided, the pattern "jigsaw" is used to define the content of session la-12.

5.3.4. Another plan of interactions

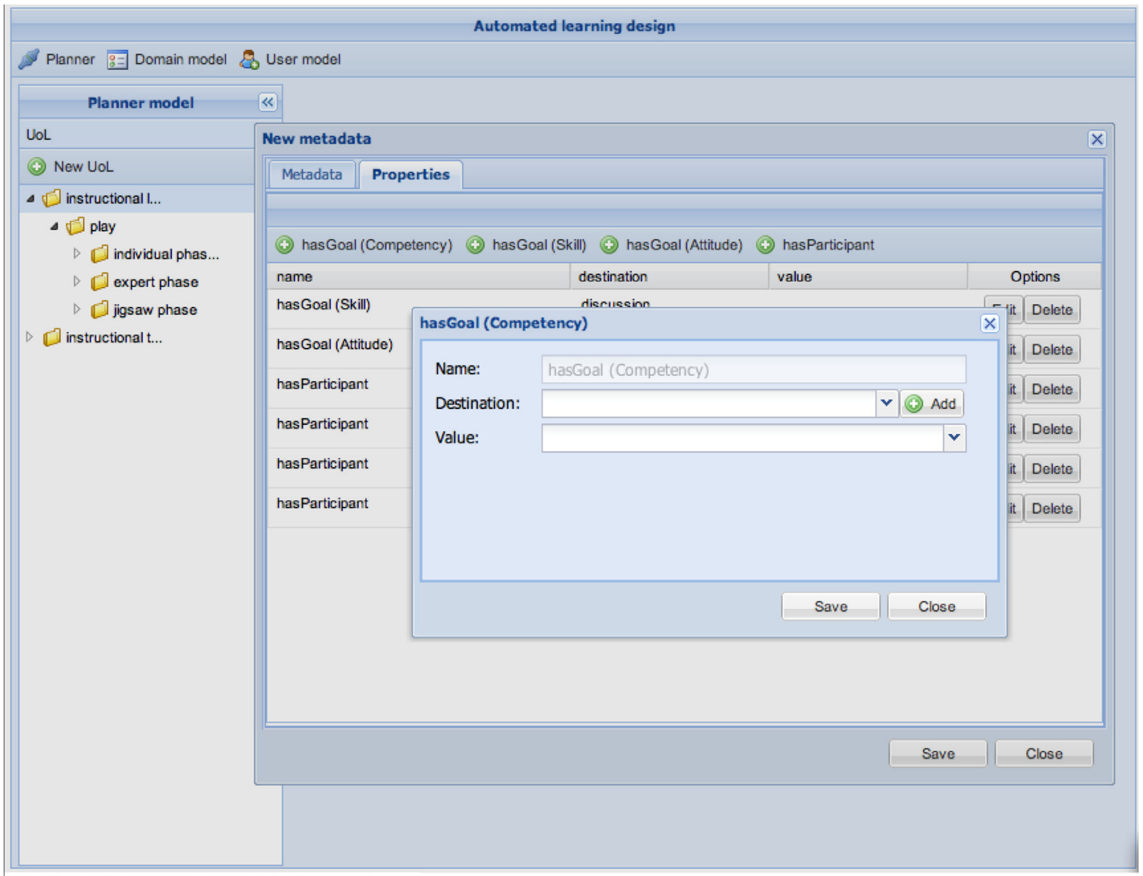
If we change the group information of learning session "demonstrate G" (la-17) in the plan "p3*a*" defined through of pattern "jigsaw" (shown in Fig. 26) to values that define the use of "cognitive apprenticeship" as CL scenario, then, the result of planning process (shown in Fig. 30) consists of one IL event (la-18) and three cyclical interactions (as-1, as-2 and as-3) that define the learning process as a "master" role for learners *l*5 and *l*6, and a "apprenticeship" role for learners *l*1, *l*2, *l*3 and *l*4. To obtain this result, it is necessary to add the atom (clscenario cog-apprenticeship ((master (l5 l6) apprenticeship (l1 l2 l3 l4)))) in the initial state *s*0.(See Fig. 31)

5.4. Technical evaluation of the research application

The results of the planning process shown in this section employed a dummy model that returned only values defined in the initial state for queries in the student model and the domain model. In addition, all tests were performed with a pre-filled result of group formation to ignore the time of group formation. This result is pre-filled through of term call (GetCLGrouping ...).

The test (results shown in Table 7) was performed by a standard PC with 2.4 GHz Intel Pentium 4 CPU with 4 GB RAM using the initial task "createLDFundamental". This task is the most abstract that involves all pattern related to pedagogical level (including of the scenarios "discover," "rehearse," "guide-tour," and "train"). The data was collected using 2, 4, 8, 16 and 32 learning goals, with 32, 64, 128 and 256 students. Each generation was repeated 10 times and the data was averaged.

Table 8 describes the number of actions that were defined by the generation of UoLs using the initial task "createLDFundamental." It shows the average size of the solution plan obtained by our approach, thus, a UoL for 2 learning goals and 32 students has 3528 actions that represents on average the ID of 352 elements (we considered the ID of an element to be obtained by ten ID actions). If the average design time of each element is one minute, then the ID of the UoL would require 352 min. These results show that our approach enables the generation of UoL for CSCL more quickly than the manual method of ID.

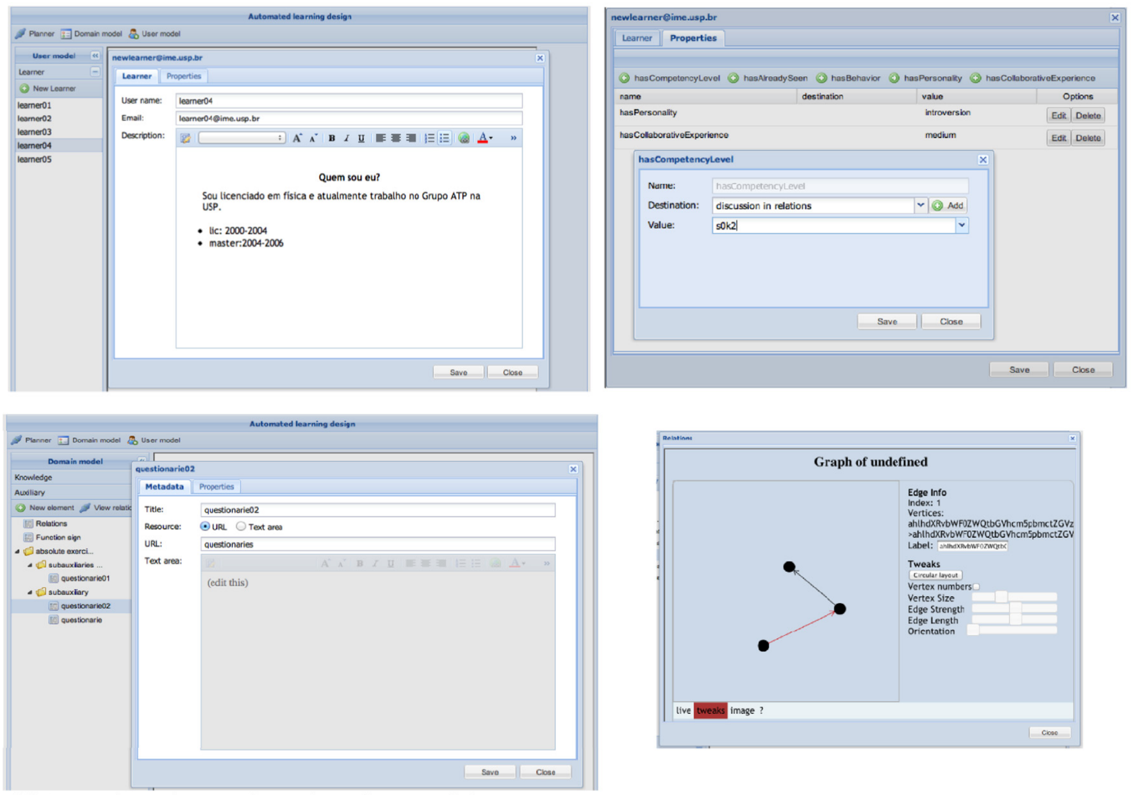


name	destination	value	Options
hasGoal (Skill)	discussion		Delete
hasGoal (Attitude)			Delete
hasParticipant			Delete
hasParticipant			Delete
hasParticipant			Delete
hasParticipant			Delete

hasGoal (Competency)

Name: hasGoal (Competency)
Destination:
Value:

(a) Initial instructional design task



newlearner@ime.usp.br

User name: learner04
Email: learner04@ime.usp.br
Description:

Quem sou eu?

Sou licenciado em física e atualmente trabalho no Grupo ATP na USP.

- lic: 2000-2004
- master:2004-2006

newlearner@ime.usp.br

name	destination	value	Options
hasPersonality		introversion	Edit Delete
hasCollaborativeExperience		medium	Edit Delete

hasCompetencyLevel

Name: hasCompetencyLevel
Destination: discussion in relations
Value: s0k2

questionarie02

Title: questionarie02
Resource: [✓ URL] Text area
URL: questionaries
Text area: (edit this)

Edge Info
Index: 1
Vertices:
ahlhdXRvbWF0ZWQtbGVhcm5pbmctZGVz
>ahlhdXRvbWF0ZWQtbGVhcm5pbmctZGV
Label: ahlhdXRvbWF0ZWQtb(

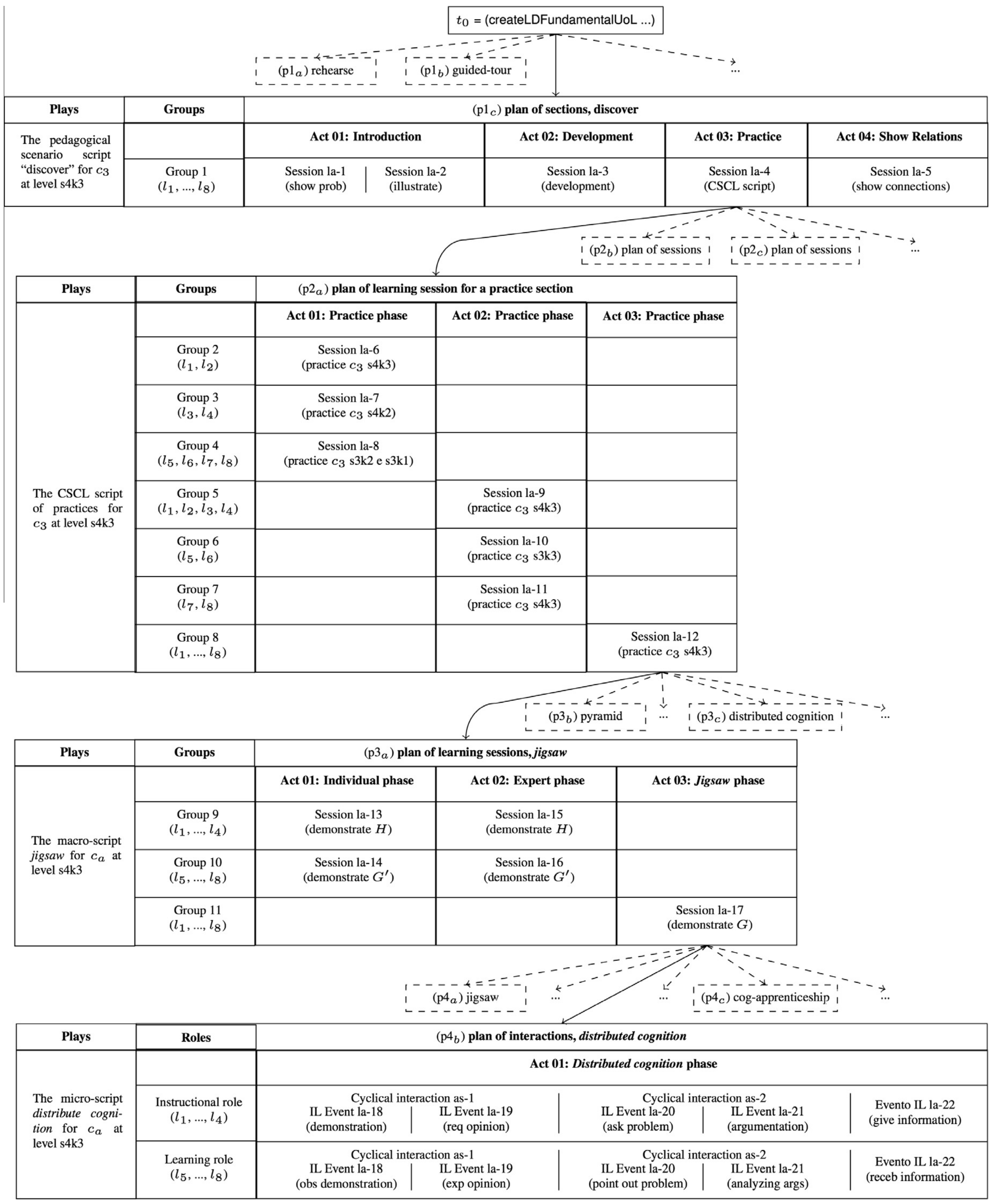
(b) Domain to be taught and student models

Fig. 25. User interfaces in the authoring tool ALD that are used to define an initial CL Scenario. In (a) a designer can define the instructional tasks; and in (b) the designer can define/check the domain and the student model.

	competence		historic		motivation	
id	c_2	c_3	r_{21}	r_{23}	c_2	c_3
l1	s0k3	s4k2	no	no	medium	high
l2	s2k2	s4k2	no	no	medium	high
l3	s2k2	s3k2	no	yes	high	high
l4	s2k2	s3k2	no	no	low	medium

	competence		historic		motivation	
id	c_2	c_3	r_{21}	r_{23}	c_2	c_3
l5	s0k3	s3k1	no	yes	low	high
l6	s1k2	s3k1	no	no	high	high
l7	s1k2	s3k0	no	yes	medium	low
l8	s0k3	s3k0	no	yes	medium	low

Fig. 26. The information of student model used in the application example.


t_0 = (createLDFundamentalUoL ...)
($p1_a$) rehearse
($p1_b$) guided-tour
...
($p2_b$) plan of sessions
($p2_c$) plan of sessions
...
($p3_b$) pyramid
...
($p3_c$) distributed cognition
...
($p4_a$) jigsaw
...
...
($p4_c$) cog-apprenticeship
...

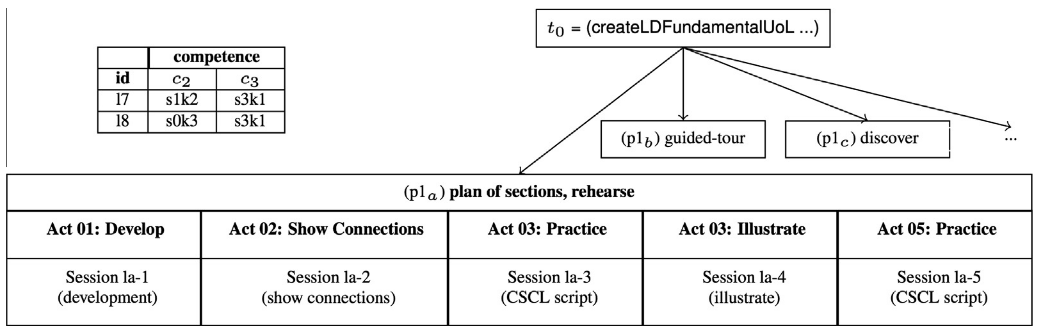
Plays	Groups	($p1_c$) plan of sections, discover				
The pedagogical scenario script "discover" for c_3 at level s4k3		Act 01: Introduction		Act 02: Development	Act 03: Practice	Act 04: Show Relations
	Group 1 ($l_1, ..., l_8$)	Session la-1 (show prob)	Session la-2 (illustrate)	Session la-3 (development)	Session la-4 (CSCL script)	Session la-5 (show connections)

Plays	Groups	($p2_a$) plan of learning session for a practice section		
The CSCL script of practices for c_3 at level s4k3		Act 01: Practice phase	Act 02: Practice phase	Act 03: Practice phase
	Group 2 (l_1, l_2)	Session la-6 (practice c_3 s4k3)		
	Group 3 (l_3, l_4)	Session la-7 (practice c_3 s4k2)		
	Group 4 (l_5, l_6, l_7, l_8)	Session la-8 (practice c_3 s3k2 e s3k1)		
	Group 5 (l_1, l_2, l_3, l_4)		Session la-9 (practice c_3 s4k3)	
	Group 6 (l_5, l_6)		Session la-10 (practice c_3 s3k3)	
	Group 7 (l_7, l_8)		Session la-11 (practice c_3 s4k3)	
	Group 8 ($l_1, ..., l_8$)			Session la-12 (practice c_3 s4k3)

Plays	Groups	($p3_a$) plan of learning sessions, *jigsaw*		
The macro-script *jigsaw* for c_a at level s4k3		Act 01: Individual phase	Act 02: Expert phase	Act 03: *Jigsaw* phase
	Group 9 ($l_1, ..., l_4$)	Session la-13 (demonstrate H)	Session la-15 (demonstrate H)	
	Group 10 ($l_5, ..., l_8$)	Session la-14 (demonstrate G')	Session la-16 (demonstrate G')	
	Group 11 ($l_1, ..., l_8$)			Session la-17 (demonstrate G)

Plays	Roles	($p4_b$) plan of interactions, *distributed cognition*				
The micro-script *distribute cognition* for c_a at level s4k3		Act 01: *Distributed cognition* phase				
		Cyclical interaction as-1		Cyclical interaction as-2		
	Instructional role ($l_1, ..., l_4$)	IL Event la-18 (demonstration)	IL Event la-19 (req opinion)	IL Event la-20 (ask problem)	IL Event la-21 (argumentation)	Evento IL la-22 (give information)
		Cyclical interaction as-1		Cyclical interaction as-2		
	Learning role ($l_5, ..., l_8$)	IL Event la-18 (obs demonstration)	IL Event la-19 (exp opinion)	IL Event la-20 (point out problem)	IL Event la-21 (analyzing args)	Evento IL la-22 (receb information)

Fig. 27. Result of the planning process.



	competence	
id	c_2	c_3
l7	s1k2	s3k1
l8	s0k3	s3k1

($p1_a$) plan of sections, rehearse				
Act 01: Develop	Act 02: Show Connections	Act 03: Practice	Act 03: Illustrate	Act 05: Practice
Session la-1 (development)	Session la-2 (show connections)	Session la-3 (CSCL script)	Session la-4 (illustrate)	Session la-5 (CSCL script)

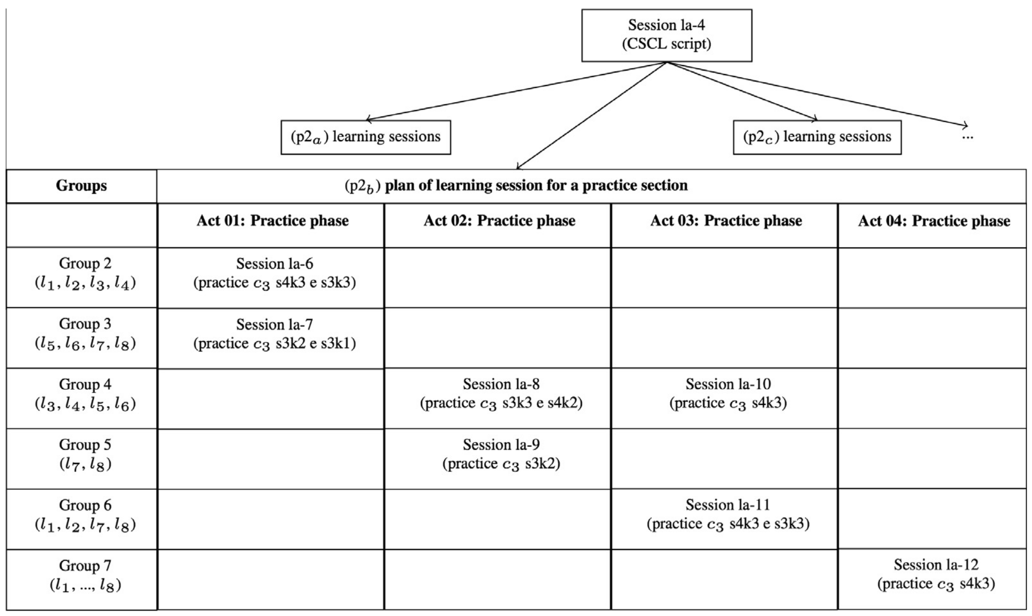
Fig. 28. The plan of sections obtained by the planning process with changes in competency levels.

6. Related works

Since the first approach for integration of automated planning in ITSs many researches employ different techniques to obtain plans tailored to each student (Peachey and McCalla, 1986). For example, the *Generic Tutoring Environment* (GTE) developed by Van Marcke (1998) allows to declaratively define the pedagogical model as a set of tasks and methods, and them use it for the planning process. The pedagogical model defined in GTE was evaluated and utilized on different ITSs such as NOBIT (cerri1992nobile), EPOS (Erol, 1990), Toska (1991) and DCG (Vassileva, 1998). Nevertheless, despite the large amount of modeled strategies with GTE, none of them can be used in collaborative learning settings.

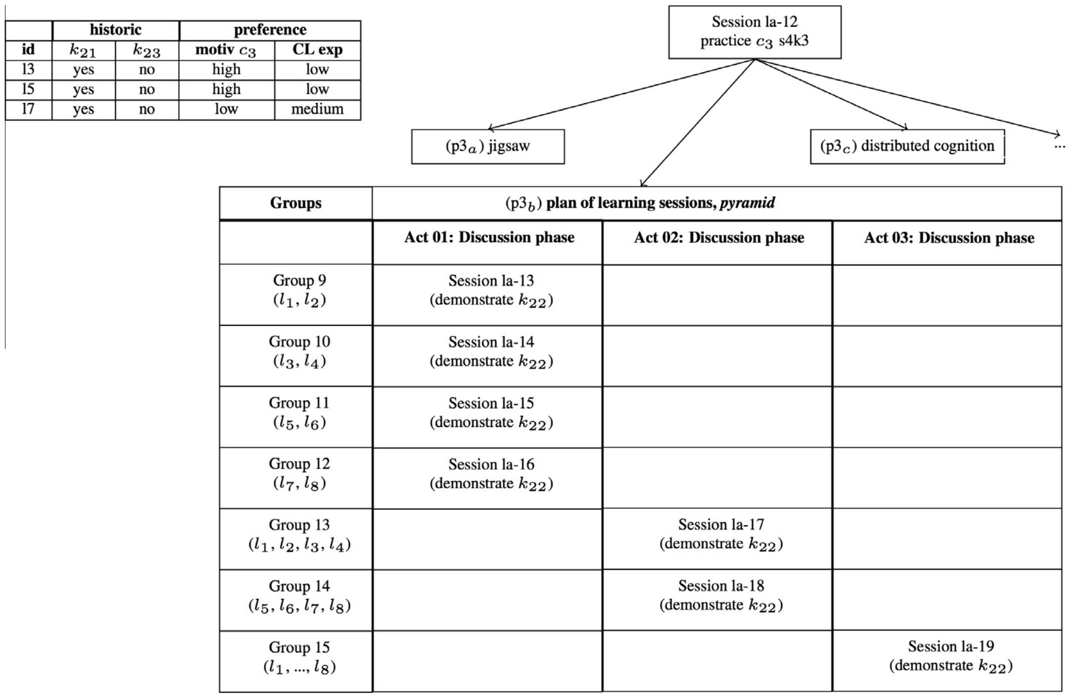
More recently, Ullrich (2008) has developed PAIGOS, a course generator that uses the JShop2 to make the instructional planning based on the representation of pedagogical model as HTN tasks and methods. Despite some similarities with our approach, PAIGOS does not contain strategies to support the creation of collaborative learning-based instructional plans. Our representation of ID strategies is based on collaborative learning theories that added more complexity to the pedagogical model that needs to represent students, their needs and the variety of interactions that can occur during the learning process. Furthermore, in our approach, the structure of concepts and its relationships with instructional resources is represented as an ontological structure separating knowledge elements (fundamental and complementary) from their properties (prerequisites and objectives).

In order to facilitate the design and adaptation of online collaborative learning activities, Karakostas, Papamitsiou, and Demetriadis (2012) created two tools, referred to as FlexCoLab and PPR, that support teachers in the process of developing flexible CSCL designs by reusing and customizing adaptation patterns according to the learning scenario. In a similar approach, Villasclaras-Fernández, Hernández-Leo, Asensio-Pérez, and Dimitriadis (2013) developed Web Collage, an online and extended version of Collage, a tool that supports the use, reuse and adaptation of CSCL design patterns and best practices to create effective learning scenarios. Both works have successfully addressed the problems of supporting the process of creating personalized well-though-out CSCL scenarios. Nevertheless, they do not support the automation of such a process. To do so, it is necessary to formalize the representation of these patterns and practices. In this direction Palomino-Ramírez et al. (2013) has proposed a formal representation of sequence of collaborative learning activities to orchestrate the execution of these activities and the evocation of educational resources. Furthermore, Rius, Conesa, García-Barriocanal, and Sicilia (2014) defined a metamodel to unambiguously specify learning processes that helps to create automatically partial implementations of learning processes from the metamodel. Those approaches addressed the problem of data flow management during the execution of CSCL activities


Session la-4 (CSCL script)
($p2_a$) learning sessions
($p2_c$) learning sessions
...

Groups	($p2_b$) plan of learning session for a practice section			
	Act 01: Practice phase	Act 02: Practice phase	Act 03: Practice phase	Act 04: Practice phase
Group 2 (l_1, l_2, l_3, l_4)	Session la-6 (practice c_3 s4k3 e s3k3)			
Group 3 (l_5, l_6, l_7, l_8)	Session la-7 (practice c_3 s3k2 e s3k1)			
Group 4 (l_3, l_4, l_5, l_6)		Session la-8 (practice c_3 s3k3 e s4k2)	Session la-10 (practice c_3 s4k3)	
Group 5 (l_7, l_8)		Session la-9 (practice c_3 s3k2)		
Group 6 (l_1, l_2, l_7, l_8)			Session la-11 (practice c_3 s4k3 e s3k3)	
Group 7 ($l_1, ..., l_8$)				Session la-12 (practice c_3 s4k3)

Fig. 29. The plan of learning sessions obtained by other way of grouping.



historic
preference
id
k_{21}
k_{23}
motiv c_3
CL exp
l3 yes no high low
l5 yes no high low
l7 yes no low medium
Session la-12
practice c_3 s4k3
(p3$_a$) jigsaw
(p3$_c$) distributed cognition
...
Groups
(p3$_b$) plan of learning sessions, *pyramid*
Act 01: Discussion phase
Act 02: Discussion phase
Act 03: Discussion phase
Group 9 (l_1, l_2)
Session la-13 (demonstrate k_{22})
Group 10 (l_3, l_4)
Session la-14 (demonstrate k_{22})
Group 11 (l_5, l_6)
Session la-15 (demonstrate k_{22})
Group 12 (l_7, l_8)
Session la-16 (demonstrate k_{22})
Group 13 (l_1, l_2, l_3, l_4)
Session la-17 (demonstrate k_{22})
Group 14 (l_5, l_6, l_7, l_8)
Session la-18 (demonstrate k_{22})
Group 15 ($l_1, ..., l_8$)
Session la-19 (demonstrate k_{22})

Fig. 30. The plan of learning sessions obtained using pattern "*pyramid*".

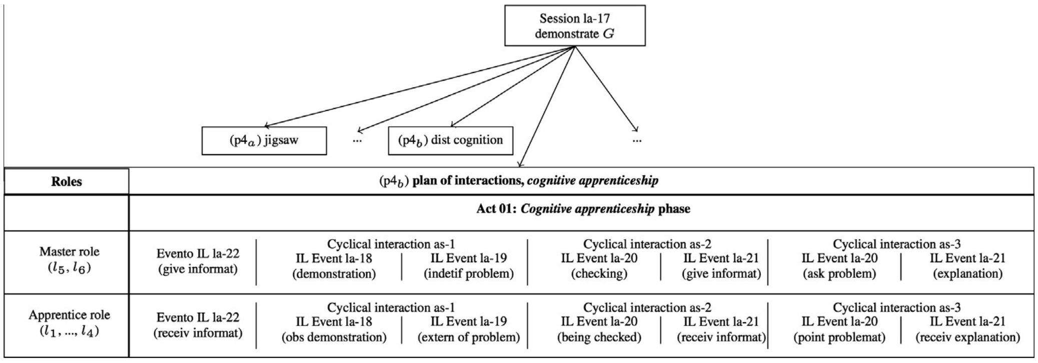
and the automation of pre-defined (i.e., static) CSCL scenarios (based on the metamodel), but they still do not support the automatic design of personalized and flexible CSCL scenarios since information about students' needs and teachers preferences are not taken into account. To address this challenge, in this work we use a holistic approach to formalize the role of all the components of a collaborative learning scenario. For example, learning objects, student roles, student characteristics (e.g., background, preferences, learning styles, etc.), instructional/learning goals, and activities, teachers' preferences, among others. Then, we propose a model to describe instructional design as hierarchical tasks and methods into a planning domain definition. Finally, we created an algorithm that uses the Hierarchical Task Network (HTN) planning approach to automate and optimize the authoring of a flexible CSCL scenarios adapted to the needs of student and teachers and the constraints of the learning environment.

Table 7
Required time (in seconds) of generation vs increasing amount of learning goals and students.

Students/Learning goals	2	4	8	16	32
32	12,17	12,25	14,46	15,13	16,79
64	17,23	27,23	35,37	37,23	47,98
128	81,51	94,85	108,81	135,36	139,21
256	154,83	297,45	328,31	371,20	417,60

7. Conclusion and future research

The design of flexible and personalized well thought out collaborative learning (CL) scenarios (i.e. UoLs) is a complex task due to the various variables that need to be considered during the designing process. To address this challenge, we proposed an approach to formally model ID using HTN planning techniques. We established a set of guidelines that convert the knowledge of ID such as CSCL scripts, design patters and best practices into HTN planning knowledge. Thus, we can use the HTN planning knowledge as a pedagogical model to create learning scenarios. We also formally describe two independent models (i.e., student model and domain model) to enable the personalization of UoLs based on students' needs and the learning objectives for a particular domain. To test our approach we create an algorithm and different uses cases to



Session la-17
demonstrate G
(p4$_a$) jigsaw
...
(p4$_b$) dist cognition
...
Roles
(p4$_b$) plan of interactions, *cognitive apprenticeship*
Act 01: *Cognitive apprenticeship* phase
Master role (l_5, l_6)
Evento IL la-22 (give informat)
Cyclical interaction as-1
IL Event la-18 (demonstration)
IL Event la-19 (indetif problem)
Cyclical interaction as-2
IL Event la-20 (checking)
IL Event la-21 (give informat)
Cyclical interaction as-3
IL Event la-20 (ask problem)
IL Event la-21 (explanation)
Apprentice role ($l_1, ..., l_4$)
Evento IL la-22 (receiv informat)
Cyclical interaction as-1
IL Event la-18 (obs demonstration)
IL Event la-19 (extern of problem)
Cyclical interaction as-2
IL Event la-20 (being checked)
IL Event la-21 (receiv informat)
Cyclical interaction as-3
IL Event la-20 (point problemat)
IL Event la-21 (receiv explanation)

Fig. 31. The plan of learning sessions obtained by *cog-appenticeship* pattern.

Table 8
Number of actions obtained by the generation of a UoL.

Student/Learning goals	2	4	8	16	32
32	3528	7240	14654	28486	56234
64	4237	8756	16458	33126	67128
128	5348	10248	20456	41245	80436
256	7796	14567	29678	61234	131146

demonstrate its validity. The results show that our algorithm can correctly use information form students (e.g., their characteristics, preferences and needs), teachers (e.g., instructional goals, teaching patters) and the resources available (e.g., learning objects) to create personalized CL scenarios. Furthermore, to enable any LMSs to use the output (scenario) of our algorithm we developed a conversion component that translates a plan (i.e. automatic generated learning scenario) into UoLs using the IMS-LD and IMS-CP specifications. Thus, any LMSs that read IMS-LD and IMS-CP standards can benefit from the results of this work. Finally, we also developed a CSCL courseware generator as a Web service to be used in the development of Adaptive and Intelligent Educational Systems to create "on-the-fly" personalized CL scenarios to help students to learn in groups a particular content. We believe the results achieved in this paper are step forwards to facilitate the inclusion of effective CL activities in online intelligent learning environments.

In the future we will continue to refine four aspects of our research. First, we are modeling other ID strategies extracted from "patterns related to resource level" and "patterns related to roles and common CL mechanisms level." Second, we are improving the conversion component of the CSCL courseware generator to enable integration with LMSs, such as Moodle. Third, we are developing new features for the ALD tool to manage (e.g., create, edit, update, and delete) the ID strategies. To develop these features in the ALD tool, recent work by Hayashi, Bourdeau, and Mizoguchi (2006) develops "an ontology of learning, instruction and instructional design" that is conceptually similar to hierarchical tasks and methods proposed in this work. In this sense, we are working to combine their complex ontology with our approach. Furthermore, we are developing an algorithm that automatically converts a set of CSCL Script Design Pattern into HTN planning knowledge. Fourth, we are developing a monitoring and evaluation component to be added into our approach model. Since the run of UoLs that are obtained through of ID for CSCL does not always proceed as planned, it is necessary to monitor the run of UoLs, and perform evaluation of expected results at each step. We are working to integrate the methods and techniques recently proposed by Isotani et al. (2013), Karakostas and Demetriadis (2011) and Villasclaras-Fernández, Hernández-Leo, Asensio-Pérez, and Dimitriadis (2009a) to identify and evaluate the impact of the run of UoLs. Finally, in future work, we will carry out experiments with real users and scenarios to compare them with conventional methods of CSCL design.

Acknowledgment

We thank the CNPq (Process: 400481/2013-8, 310204/2011-9 and 550449/2011-6) for providing support for this research. The authors also wish to express their gratitude to the reviewers and editors of Expert Systems with Applications who spent their time and effort to improve the quality of this work.

References

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

Dillenbourg, P., & Jermann, P. (2007). Designing integrative scripts. In F. Fischer, I. Kollar, H. Mandl, & J. Haake (Eds.), *Scripting Computer-Supported Collaborative Learning. Computer-Supported Collaborative Learning* (vol. 6, pp. 275–301). US: Springer.

Erol, N. (1990). Epos collaborative learning environment. DELTA Project EPOS D7002 1C 90–02-02, ECOLE, Milano: Dida*El.

Garrido, A., Onaindia, E., & Sapena, O. (2008). Planning and scheduling in an e-learning environment. A constraint-programming-based approach. *Engineering Applications of Artificial Intelligence, 21*(5), 733–743.

Ghallab, M., Nau, D., & Traverso, P. (2004). Automated planning: theory & practice. Access online via Elsevier.

Hayashi, Y., Bourdeau, J., & Mizoguchi, R. (2006). Ontological support for a theory-eclectic approach to instructional and learning design. In W. Nejdl & K. Tochtermann (Eds.), *Innovative Approaches for Learning and Knowledge Sharing. Lecture Notes in Computer Science* (pp. 155–169). Berlin Heidelberg: Springer.

Hernández-Leo, D. (2007). A pattern-based design process for the creation of CSCL macro-scripts computationally represented with IMS LD. Ph.D. thesis, ETSIT, University of Valladolid, Valladolid, Spain.

Hernández-Leo, D., Pérez, J. I. A., & Dimitriadis, Y. A. (2004). Ims learning design support for the formalization of collaborative learning patterns. In *Proceedings of IEEE International Conference on Advanced Learning Technologies* (pp. 350–354). IEEE.

Hernández-Leo, D., Villasclaras-Fernández, E., Asensio-Pérez, J., & Dimitriadis, Y. (2009). Generating cscl scripts: From a conceptual model of pattern languages to the design of real scripts. *E-learning Design Patterns Book*, 49–64.

Hernández-Leo, D., Villasclaras-Fernández, E., Jorrín-Abellán, I., Asensio-Pérez, J., Dimitriadis, Y., Ruiz-Requies, I., et al. (2006). COLLAGE, a collaborative learning design editor based on patterns. *Journal of Educational Technology and Society, 9*, 58–71.

Ilghami, O. (2006). Documentation for JSHOP2. Technical report, Department of Computer Science, University of Maryland.

IMS GLC, (2003). IMS learning design information model. Version 1.0 Final Specification. IMS Global Learning Consortium. http://www.imsglobal.org/learningdesign/. [Last visited on 01-07-2013].

IMS GLC, (2004). IMS content packaging information model. Version 1.1.4 Final Specification. IMS Global Learning Consortium. http://www.imsglobal.org/content/packaging/. [Last visited on 01-07-2013].

Inaba, A., Ikeda, M., & Mizoguchi, R. (2003). What learning patterns are effective for a learner's growth. In *Proceedings of the international conference on artificial intelligence in education*, Sydney (pp. 219–226).

Isotani, S., & Mizoguchi, R. (2007). Deployment of ontologies for an effective design of collaborative learning scenarios. In *Proceedings of the Thirteenth International Conference on Groupware: Design Implementation, and Use, CRIWG'07* (pp. 223–238). Berlin, Heidelberg: Springer-Verlag.

Isotani, S., & Mizoguchi, R. (2008). Adventures in the boundary between domain-independent ontologies and domain content for CSCL. In *Proceedings of the International Conference on Knowledge-Based Intelligent Information and Engineering Systems. Lecture Notes in Computer Science* (Vol. 5179, pp. 523–532). Berlin Heidelberg: Springer.

Isotani, S., Mizoguchi, R., Isotani, S., Capeli, O. M., Isotani, N., & Albuquerque, A. R. (2010). An authoring tool to support the design and use of theory-based collaborative learning activities. In V. Aleven, J. Kay, & J. Mostow (Eds.), *Intelligent Tutoring Systems. Lecture Notes in Computer Science* (vol. 6095, pp. 92–102). Berlin Heidelberg: Springer.

Isotani, S., Mizoguchi, R., Isotani, S., Capeli, O. M., Isotani, N., Albuquerque, A. R. P. L., et al. (2013). A semantic web-based authoring tool to facilitate the planning of collaborative learning scenarios compliant with learning theories. *Computers & Education, 63*, 267–284.

Kaplan, R., & Rock, D. (1995). New directions for intelligent tutoring. *AI Expert* (vol. 10, pp. 30–40). San Francisco, CA: CL Publications, Inc..

Karakostas, A., & Demetriadis, S. (2011). Adaptation patterns as a conceptual tool for designing the adaptive operation of CSCL systems. *Educational Technology Research and Development, 59*(3), 327–349.

Karakostas, A., Papamitsiou, Z. & Demetriadis, S. (2012) Prototype tools for the flexible design of CSCL activities based on the adaptation pattern perspective. Intelligent adaptation and personalization techniques in computer-supported collaborative learning. *Studies in Computational Intelligence* (Vol. 408, pp. 109–130). Berlin Heidelberg: Springer.

Kobbe, L., Weinberger, A., Dillenbourg, P., Harrer, A., Hämäläinen, R., Häkkinen, P., et al. (2007). Specifying computer-supported collaboration scripts. *International Journal of Computer-Supported Collaborative Learning, 2*(2–3), 211–224.

Koper, R., & Olivier, B. (2004). Representing the learning design of units of learning. *Educational Technology & Society, 7*(3), 97–111.

Melis, E., Faulhaber, A., Eichelmann, A., & Narciss, S. (2008). Interoperable competencies characterizing learning objects in mathematics. In B. Woolf, E. AÃmeur, R. Nkambou, & S. Lajoie (Eds.), *Intelligent Tutoring Systems. Lecture Notes in Computer Science* (vol. 5091, pp. 416–425). Berlin: Springer.

Nau, D., Au, T.-C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., et al. (2003). SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research, 20*, 379–404.

Olivier, B., & Tattersall, C. (2005). The learning design specification. In R. Koper & C. Tattersall (Eds.), *Learning Design* (pp. 21–40). Berlin Heidelberg: Springer.

Peachey, D. R., & McCalla, G. I. (1986). Using planning techniques in intelligent tutoring systems. *International Journal of Man–Machine Studies, 24*(1), 77–98.

Pinkwart, N. (2003). A plug-in architecture for graph based collaborative modeling systems. In *Supplementary proceedings of the eleventh conference on artificial intelligence in education*, Sydney, Australia (pp. 89–94).

Rius, A., Conesa, J., García-Barriocanal, E., & Sicilia, M. (2014). Automating educational processes implementation by means of an ontological framework. *Computer Standards & Interfaces, 36*(2), 335–348.

Sicilia, M. (2005). Ontology-based competency management: Infrastructures for the knowledge-intensive learning organization. In M. Lytras, & A. Naeve (Eds.), *Intelligent learning infrastructure for knowledge intensive organizations: A semantic web perspective* (pp. 302–324). Hershey, PA: Information Science Publishing.

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* (pp. 409–426). Cambridge University Press.

Toska (1991). Toska final report. Technical report, CEC delta project D1007, Friendrichshafe: Dornier.

Ullrich, C. (2008). *Pedagogically Founded Courseware Generation for Web-based Learning: An HTN-Planning-based Approach Implemented in PAIGOS* (vol. 5260). Springer-Verlag.

Van Marcke, K. (1998). GTE: An epistemological approach to instructional modelling. *Instructional Science, 26*(3–4), 147–191.

Vassileva, J. (1998). DCG + GTE: Dynamic courseware generation with teaching expertise. *Instructional Science, 26*(3–4), 317–332.

Villasclaras-Fernández, E. D., Hernández-Leo, D., Asensio-Pérez, J. I., & Dimitriadis, Y. (2009). Incorporating assessment in a pattern-based design process for CSCL scripts. *Computers in Human Behavior, 25*(5), 1028–1039.

Villasclaras-Fernández, E. D., Isotani, S., Hayashi, Y., & Mizoguchi, R. (2009). Looking into collaborative learning: Design from macro-and micro-script perspectives. In *Proceedings of the International Conference on Artificial Intelligence in Education* (pp. 231–238). Amsterdam,The Netherlands: IOS Press.

Villasclaras-Fernández, E. D., Hernández-Leo, L., Asensio-Pérez, J. I., & Dimitriadis, Y. (2013). Web Collage: An implementation of support for assessment design in CSCL macro-scripts. *Computers & Education, 67*, 79–97.

Vrakas, D., Tsoumakas, G., Kokkoras, F., Bassiliades, N., Vlahavas, I., & Anagnostopoulos, D. (2007). Paser: a curricula synthesis system based on automated problem solving. *International Journal of Teaching and Case Studies, 1*(1), 159–170.

Wasson, B. (1996). Instructional planning and contemporary theories of learning: Is this a self-contradiction. In *Proceedings of the European Conference on Artificial Intelligence in Education* (pp. 23–30). Lisbon: Colibri.

Weinberger, A., Fischer, F., & Stegmann, K. (2005). Computer-supported collaborative learning in higher education: scripts for argumentative knowledge construction in distributed groups. In *Proceedings of the 2005 Conference on Computer Support for Collaborative Learning: Learning 2005: The next 10 years!, CSCL '05* (pp. 717–726). International Society of the Learning Sciences.

Wilkinson, J. (2001). A matter of life or death: re-engineering competency-based education through the use of a multimedia CD-ROM. In *Proceedings of IEEE international conference on advanced learning technologies* (pp. 205–208).

Woolf, B. P. (2010). *Building Intelligent Interactive Tutors: Student-centered Strategies for Revolutionizing e-Learning*. Morgan Kaufmann.