T2/ediT2 : a flexible and easy-to-use model/system for editing and operationalizing learning scenarios

Péricles de Lima Sobreira

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

Péricles DE LIMA SOBREIRA

Thèse dirigée par **Pierre TCHOUNIKINE**

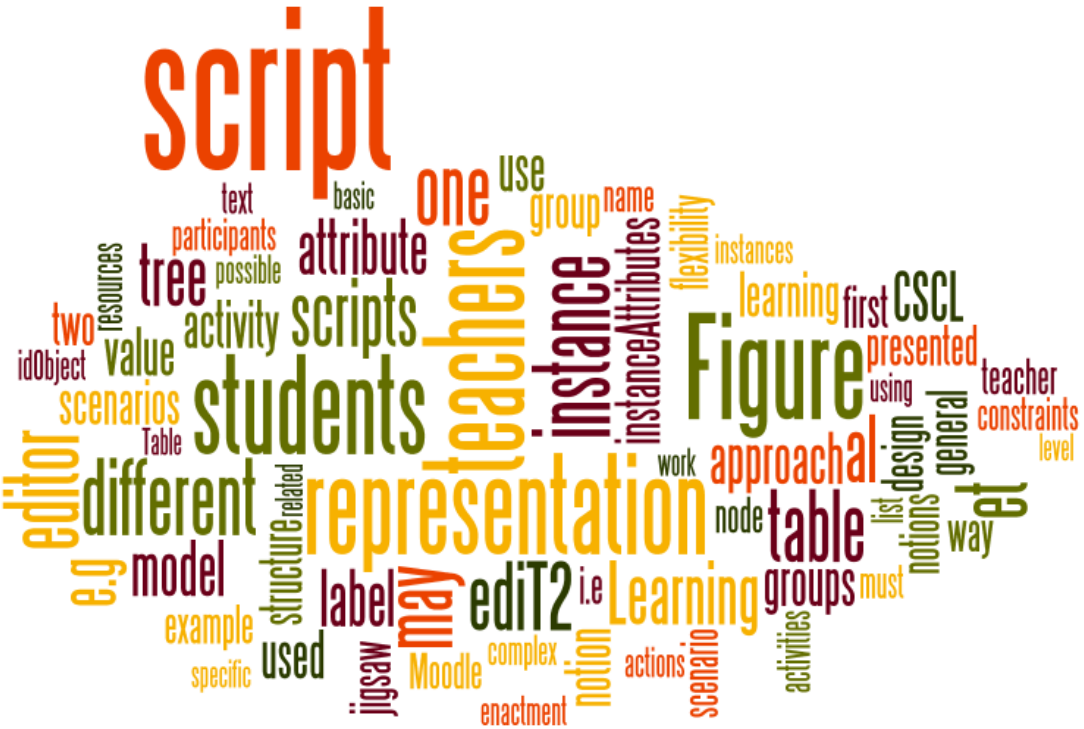
préparée au sein du **Laboratoire d'Informatique de Grenoble** dans **l'École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique**

T^2/ediT2 : Un modèle/système flexible et facile à utiliser pour l'édition et mise en œuvre de scénarios d'apprentissage

Thèse soutenue publiquement le **26 juin 2014**,
devant le jury composé de :

M. Christophe CHOQUET
Professeur des Universités. Université du Maine, **Rapporteur et Président du jury**
M. Serge GARLATTI
Professeur des Universités. Télécom Bretagne, **Rapporteur**
M. Franck BELLEMAIN
Professor Adjunto IV. Universidade Federal de Pernambuco, **Examinateur**
M. Jean-Charles MARTY
Maître de Conférences. Université de Savoie, **Examinateur**
M. Pierre TCHOUNIKINE
Professeur des Universités. Université Joseph Fourier, **Directeur de thèse**

Université Joseph Fourier / Université Pierre Mendès France / Université Stendhal / Université de Savoie / Grenoble INP


script
one
use
group
name
text
basic
participants
attribute
tree
possible
resources
instances
flexibility
instanceAttributes
learning
first
CSCL
two
activity
scripts
teachers
instance
Figure
presented
idObject
value
using
teacher
scenarios
students
constraints
Table
general
level
editor
different
representation
approach
al
work
design
et
related
structure
node
table
list
notions
way
e.g
model
label
may
ediT2
i.e
Learning
groups
must
example
used
Moodle
complex
notion
actions
scenario
activities
specific
jigsaw
enactment

Résumé

La question générale envisagée dans cette recherche est le développement d'une représentation de scénarios d'apprentissage adaptable et facile à utiliser sous la forme d'une table (considéré comme un artefact de facile manipulation par les enseignants), associée à un modèle informatique sous la forme d'en arbre (comme un moyen d'intégrer des services avancés). Cette représentation permet à des enseignants sans entraînement méthodologique et ayant des compétences technologiques de base d'éditer et mettre en œuvre des scénarios d'apprentissage à partir d'une interface graphique intuitive et flexible.

Bien que cette thèse soit centrée sur des scénarios collaboratifs, l'approche basée sur un modèle table-arbre (nommé T^2) que nous proposons présente un intérêt plus général. Dans une première phase, nous avons développé à partir de ce modèle un éditeur de scénarios d'apprentissage (nommé ediT2) proposant des notions de modélisation utilisées dans les scénarios collaboratifs. Dans une seconde phase, nous avons considéré des questions de généralisation à travers l'extension de l'implémentation initiale, de telle manière à permettre aux utilisateurs d'éditer les notions et leurs attributs.

Nous avons examiné à travers des études et expériences comment des enseignants ont utilisé notre proposition en tenant en compte comme objectifs/critères d'évaluation: (1) son expressivité pédagogique, i.e., si des éditeurs basés sur tables peuvent représenter une large gamme de scénarios d'apprentissage ; (2) sa facilité et son intuitivité ; (3) son expressivité informatique, i.e., si l'approche permet l'implémentation de services demandant des manipulations informatiques complexes ; et (4) sa flexibilité informatique, i.e., s'il est facile d'adapter l'éditeur à des besoins locaux.

MOTS CLÉS : Environnements Informatiques pour l'Apprentissage Humain ; Environnements Informatiques pour l'Apprentissage Collaboratif ; Scénarisation ; Langage de Modélisation Pédagogique ; Conception/Edition/Mise en Œuvre/Surveillance de scénarios d'apprentissage.

Abstract

The general issue considered in this research is the development of an adaptable and easy-to-use representation of learning scenarios in the form of a table (considered as an artefact of easy manipulation by teachers) associated with a computational model as a tree (as a way to integrate advanced services). In this way, teachers with basic technological skills and without methodological training can edit and operationalize learning scenarios from flexible and friendly graphical interfaces.

Although this thesis has its focus on CSCL scripts, the table-tree-based approach (named T^2) presents a more general interest. In a first moment, we implemented from this model a learning scenario editor (named ediT2) using notions from CSCL scripts. In a second moment, we considered generalization issues through the extension of the initial implementation, in order to allow teachers to edit their own notions and respective attributes.

We investigated from different studies and experiments how teachers used our proposal considering as objectives/evaluation criteria the following features: (1) pedagogical expressiveness (can table-based editors represent a wide range of learning scenarios?); (2) usability (do teachers find the editor easy to use and intuitive?); (3) computational expressiveness (does the approach allow implementation of advanced services?), and; (4) computational flexibility (is the editor easy to adapt to local needs?).

KEYWORDS: Technology Enhanced Learning; Computer-Supported Collaborative Learning; Scripting; Educational Modeling Language; Design/Edition/ Operationalization/Monitoring of Learning Scenarios.

Remerciements

Comment remercier en quelques paragraphes à tous ceux qui sont passés dans ma vie et qui m'ont amené jusque-là ? Impossible, n'est-ce pas ? Mais je vais essayer…

Tout d'abord, je voudrais remercier à mon bon Dieu, de m'avoir donné les moyens et les forces d'y arriver. Merci, my Lord.

Bien sûr, à toute ma famille, spécialement à ma mère (Neuza), mes frères (Tarita et Rhadamés), mon épouse (Joelma) et mes enfants (Hannah, Isabelle et Péricles Filho), de toujours m'avoir soutenu et dit les bons mots pour m'encourager. Merci à tous d'avoir été à mes côtés pendant toutes ces années, dans les bons comme dans les mauvais moments, à me soutenir dans la concrétisation de ce rêve (sans vous je n'aurais jamais réussi). Merci de tout mon cœur à vous tous.

Si je regarde en arrière, je me souviens de mes maîtres et de mes collègues de l'école primaire, du collège et du lycée, et en particulier de mon professeur de mathématiques à Recife qui m'a toujours démystifié les chemins de l'Algèbre et de la Géométrie, Professeur Reginaldo Gomes. A vous tous, merci beaucoup. Aussi, à mes collègues de Universidade Federal de Pernambuco, tout comme à mes professeurs à qui je remercie sincèrement, plus particulièrement à Franck Bellemain, qui m'a introduit dans le domaine de l'apprentissage assistée par la technologie, qui m'a aidé à avancer vers le chemin de l'investigation scientifique et qui finalement m'a recommandé à mon Directeur de Recherche Pierre Tchounikine, lors de ma démarche d'admission dans l'Ecole Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique de l'Université de Grenoble. Je ne saurais jamais assez te remercier Franck.

En France, je voudrais remercier tous mes collègues qui m'ont chaleureusement accueilli au sein de l'équipe MeTAH : mes jours sont devenu meilleurs avec vos mots de réconfort et de motivation. Un grand merci à toutes ces personnes spéciales : Anne, Aurélie, Ben, Cédric, Celso, Chrysanthi, Claire, Hamid, Isabelle, Jacques, Jean-Michel, José-Luis, Marjolaine, Muriel, Nachoua, Nadine, Patricia, Patrick, Pierre, Rachel, Reinaldo, Viviane et Zilora.

Je voudrais enregistrer mes éternels remerciements à mon directeur de thèse, Pierre Tchounikine, pour m'avoir accepté, avoir cru en moi et en mon potentiel dès nos premiers échanges alors que j'étais encore au Brésil. Je m'en souviens aussi des premiers mots que nous avons échangé dès que je suis arrivé en France. Il m'a demandé « Comment est-ce que vous vous sentez ? », et je lui ai répondu « J'ai peur », et il m'a soudainement dit : « Ne vous inquiétez pas, nous sommes là pour vous soutenir ». Aujourd'hui je peux te dire une chose, le doctorat a été un stage très important dans ma vie, qui s'est très bien passé à tes côtés. Merci beaucoup pour tout Pierre.

Mes sincères remerciements à mon jury de thèse d'avoir accepté d'en faire partie: Christophe Choquet (Université du Maine), Serge Garlatti (Télécom Bretagne), Franck Bellemain (Universidade Federal de Pernambuco) et Jean-Charles Marty (Université de Savoie). Vous m'avez montré qu'il existe encore des étapes à suivre au-delà de mon horizon. Merci aux Messieurs Choquet et Garlatti, de m'avoir fait l'honneur d'accepter d'être rapporteurs de cette thèse : merci pour les commentaires et suggestions faites avant et pendant la soutenance. Merci à Monsieur Marty, pour montrer des points d'instigation à travailler entre notre recherche et la sienne. Et un autre merci à Franck Bellemain, qui était encore une fois présent dans cette étape de ma formation et de ma vie.

Merci aux personnes qui ont participé aux expérimentations faites pour la validation de nos idées. Un merci très spécial à Christine Tchounikine, Hamid Chaachoua et Philippe Dessus, qui nous ont permis de travailler avec plusieurs enseignants et de mettre au point nos idées pendant le déroulement de cette recherche. Merci également au groupe de recherche GSIC de Universidad de Valladolid, qui a collaboré dans plusieurs points de ce travail. Merci à vous, Chus, LP, Asensio et Dimitriadis pour tout le soutien donné.

Finalement, un remerciement très spécial à mes amis de Grenoble et du Brésil : Jai, Guga, Vitão, Marcão, Afonso Henriques, Jauberth, Suely et Bringel, Bia et Cesar. Vos encouragements m'ont beaucoup aidé. Je garderais de très bons souvenirs des journées et des nuits où nous parlions de notre avenir en France, des fou rires, et surtout quand vous essayiez de m'apprendre quelques jeux de cartes. Merci à vous tous. Je voudrais finaliser ces remerciements à mes amis qui ont eu la patience de revoir et corriger la version finale de cette thèse : Brigitte, Jack et Adriano, merci pour toute votre aide et amitié.

"Don't gain the world and lose your soul.
Wisdom is better than silver and gold"

Bob Marley

Contents

1 Introduction 21
1.1 General context 24
1.2 Thesis organization 25

2 Representation and edition of CSCL scripts 27
2.1 CSCL scripts 30
2.2 Review of activities 31
2.3 Learning scenarios languages and platforms 33
2.3.1 General modeling-based perspective 33
2.3.2 CSCL modeling-based perspective 34
2.3.3 Pattern-based approaches 36
2.3.4 Operationalization-based approaches 36
2.4 Flexibility in CSCL scripts 37
2.5 Review of matters of concern and objectives 39

3 Problematic, adopted approach and methodology 41
3.1 Considered issue 43
3.2 Research question 43
3.3 Adopted approach 44
3.4 Historical dimensions and methodology 45

4 The T^2 model and the ediT2 editor 47
4.1 General considerations 50
4.2 The table/tree structure 50
4.3 Proof of concept – the ediT2 editor 53
4.4 Semantics 56
4.5 Offered flexibility 59
4.6 Constraints 60
4.6.1 The allocation of items into table cells 60
4.6.2 The script-structure 61
4.6.3 The 1-n relationship 61
4.7 Adding control rules 64
4.8 T^2 tree manipulation algorithms 65

5 Validation of ease of table-tree representation, perceived flexibility and pedagogical expressiveness 67
5.1 Initial considerations 69
5.2 Is the model/system easy to use? 69
5.2.1 First usability test 69
5.2.2 Second usability test 71

5.2.3 Analysis of the technical competences required to use the editor with respect to teachers' ICT skills 72
5.2.4 Discussion 73
5.3 Do teachers avail themselves of the flexibility provided? 74
5.4 Does a table notation allow representation of a wide range of scenarios? 78

6 Enhancing the editor/model through the development of advanced functionalities 80
6.1 General considerations 83
6.2 Instantiation support 83
6.3 Respect of constraints 86
6.4 Representation of complex scheduling structures 88
6.5 Simulation 92

7 Operationalization 95
7.1 General considerations 98
7.2 Middleware-based strategy 98
7.3 *Ad hoc* strategy 101
7.4 Focus on technical representation transformations 103
7.5 Discussion 104

8 Editor generalization 106
8.1 Initial considerations 109
8.2 Developing a specific variant: the MediaWiki example 109
8.3 A generic editor 112
8.3.1 Making the editor generic 112
8.3.2 Example #1: using an ediT2 variant to represent preparation sheets 114
8.3.3 Example #2: using an ediT2 variant to represent SceDer scenarios 115
8.3.4 Example #3: introducing monitoring information in the scenario design 116

9 Discussion, conclusions and perspectives 119
9.1 Discussion 122
9.1.1 Originality 122
9.1.2 Application scope 123
9.1.3 Relation with other works 124
9.2 Conclusions 126
9.3 Perspectives and exploratory work 127
9.3.1 General perspectives 127
9.3.2 Monitoring perspective: exploratory work 128

Appendix A – Approach's architectural considerations 132
Appendix B – t2ML..... 134
B.1 t2ML Schema 134
B.2 t2ML Document 135
Appendix C – Trace files considerations 146
Appendix D – Using ediT2 to represent CSCL scripts 147
D.1 MURDER 148
D.2 Social..... 148
D.3 Pyramid 149
D.4 ConceptGrid..... 149
D.5 Universanté..... 150
D.6 MagicBook 150
D.7 RSC..... 151
D.8 Signifié-Signifiant Collaborative Play 151
D.9 Scaffolding Group Feedback and Explanation During Practice Time 152
D.10 ArgueGraph 153
Appendix E – Analysis of the work..... 154
Bibliography 158

List of Figures

Figure 2.1 A reciprocal-teaching script, adapted from (Palincsar and Brown, 1984 *apud* Kollar et al., 2006) 31
Figure 2.2 A jigsaw script, adapted from (Hernández-Leo et al., 2006) 31
Figure 4.1 A jigsaw script (Figure 2.2) as a table 51
Figure 4.2 The jigsaw script (Figure 4.1) as a tree 51
Figure 4.3 The ediT2 general interface (jigsaw script (Figure 2.2) from the first usability test, Chapter 5) 54
Figure 4.4 Using ediT2 to represent the reciprocal-teaching script (from the first usability test, Chapter 5) 55
Figure 4.5 “Split a cell” algorithm 65
Figure 4.6 “Move left/right a column” algorithm 66
Figure 6.1 Possible interfaces to inform jigsaw (top) and reciprocal-teaching (bottom) scripts’ data configuration 85
Figure 6.2 Instantiation mechanisms for jigsaw scripts 85
Figure 6.3 Automatic generation of a jigsaw script (the tree (left) and parts of the table presentation (right)) 86
Figure 6.4 Firing out the pattern constraints verification module (to the jigsaw pattern with “22 students shared in extensible groups of 4 students” as configuration parameters) 87
Figure 6.5 Interoperating ediT2 and jBPM 90
Figure 6.6 A generated jBPM skeleton (from the jigsaw script presented in Figure 4.3) 90
Figure 6.7 Completed workflow 92
Figure 6.8 Completed workflow + simulation 93
Figure 7.1 ediT2-GLUE!-PS mapping (from the jigsaw script presented in Figure 4.3) 99
Figure 7.2 GLUE!-PS–Moodle mapping from the jigsaw script presented in Figure 4.3 (administrative (teacher) view) 99
Figure 7.3 A possible design of a simple role-play script from ediT2 100
Figure 7.4 A simple role-play script deployed into Moodle via GLUE!-PS (from the script presented in Figure 7.3) 101
Figure 7.5 New ediT2 play service (top). List of Moodle courses that a specific student (“e1”) is enrolled (middle). List of resources of a course assigned to such student (bottom) 103
Figure 7.6 Transforming the representation from one script-structure to another 104
Figure 8.1 “Brazil” scenario example modeled from the ediT2-MediaWiki instantiation 110
Figure 8.2 The “Brazil” scenario (Figure 8.1) when deployed into MediaWiki 111
Figure 8.3 User rights in the edition of wiki pages (“Debie” and “John” accessing the “Brazil’s history” Topic) 111

Figure 8.4 Instantiation mechanism for a wiki-based pattern 112
Figure 8.5 Defining the general and instance attributes for the “Task” notion 113
Figure 8.6 Learning scenario designed with general and instance lists of attributes 113
Figure 8.7 “Classroom preparation sheet” scenario 114
Figure 8.8 The “Chem Drawing” scenario (extract copied in extenso from Niramitranon et al., 2010) 116
Figure 8.9 The “Chem Drawing” scenario (Figure 8.8) designed from the ediT2-SceDer tool 116
Figure 9.1 ediT2 used as a script conducting cockpit (from the jigsaw script presented in Figure 4.3): the first activity is run out (top); the first activity is monitored (bottom) 129
Figure A.1 ediT2 general architecture 132
Figure D.1 The “MURDER” script narrative (top) and representation (bottom) (Dansereau et al., 1979 apud Kobbe et al., 2007) 148
Figure D.2 The “Social” script narrative (top) and representation (bottom) (Weinberger et al., 2005 apud Kobbe et al., 2007) 148
Figure D.3 The “Pyramid” script narrative (top) and representation (bottom; instanced for 6 students) (adapted from Karakostas and Demetriadis, 2009) 149
Figure D.4 The “ConceptGrid” script narrative (top) and representation (bottom; instanced for 2 groups composed by pairs) (adapted from Dillenbourg et al., 2009) 149
Figure D.5 The “Universanté” script narrative (top) and representation (bottom; instanced for triplets from 3 different countries) (Dillenbourg and Jermann, 2007 apud Kobbe et al., 2007) 150
Figure D.6 The “MagicBook” script narrative (top) and representation (bottom; instanced for 3 students) (adapted from Dillenbourg, 2002) 150
Figure D.7 The “RSC” script narrative (top) and representation (bottom; instanced for 3 students) (Betbeder and Tchounikine, 2003 apud Dillenbourg and Tchounikine, 2007 (adapted)) 151
Figure D.8 The “Signifié-Signifiant Collaborative Play” script narrative (top) and representation (bottom; here, the change of the teams’ roles is not modeled for the figure readability sake) (Ioannidou and Dimitracopoulou, 2003, 2004 apud Van de Velde et al., 2004 (adapted)) 152
Figure D.9 The “Scaffolding Group Feedback and Explanation During Practice Time” script narrative (top) and representation (bottom; instanced for 3 students and considering that in the second activity all students have agreed on a same correct answer) (adapted from Roschelle et al., 2009) 152
Figure D.10 The “ArgueGraph” script narrative (top) and representation (bottom; instanced for 4 students) (Dillenbourg and Jermann, 2007 apud Kobbe et al., 2007) 153

List of Tables

Table 4.1 A basic interpretation of the is-associated-with generic relationship. 57
Table 4.2 Basic actions to adapt a script (examples from the script presented in Figure 4.4) 57
Table 4.3 Rationale for the 1-n with n ≥ 1 relationship 62
Table 4.4 Examples of manipulations of script representations and propagations 64
Table 5.1 The French usability test protocol 70
Table 5.2 Questions related to the editor and number of answers 71
Table 5.3 The Spanish usability test protocol 72
Table 5.4 Script skeletons 75
Table 5.5 Number of different involved items 76
Table 5.6 Modifications of script-structures 76
Table 5.7 Users' actions 77
Table 6.1 Examples of pattern constraint-checking mechanisms (to the jigsaw pattern) 87

Abbreviations

Application Programming Interface	API
Computer Based Pedagogical Setting	CBPS
Computer Supported Collaborative Learning	CSCL
Educational Modeling Language	EML
Information and Communication Technology	ICT
Instructional Management Systems – Learning Design	IMS-LD
Learning Management System	LMS
Learning Scenarios Editor	LSE
Not Applicable	N/A
Rich Desktop Application	RDA
Rich Internet Application	RIA
T^2 model	Table-Tree model
Technology Enhanced Learning	TEL

1 Introduction

1.1 General context

1.2 Thesis organization

Résumé en français

Les outils informatiques pour la gestion de scénarios d'apprentissage peuvent offrir différents services en fonction de leur publics cibles : outils orientés vers le design pour les ingénieurs pédagogiques ; outils orientés vers l'édition pour les enseignants ; outils orientés vers la mise en œuvre pour le personnel technique ; ou outils orientés vers l'accompagnement pour les enseignants/tuteurs.

Si nous considérons des pratiques générales, c'est-à-dire, celles des enseignants ayant des compétences technologiques de base et sans entraînement méthodologique, l'utilisation de ces outils est peu développée. L'une des raisons est que des langages/plateformes devraient prendre en compte le fait que les enseignants espèrent que les outils pédagogiques soient faciles à utiliser et adaptables à leurs besoins.

Dans ce contexte, la problématique générale envisagée dans cette thèse est de comprendre comment offrir à ces utilisateurs des outils pour la représentation et la mise en œuvre de scénarios collaboratifs et plus généralement, de scénarios d'apprentissage. L'objectif est de proposer un éditeur flexible, qui offre une interface intuitive pour des enseignants avec des compétences technologiques de base et sans entraînement particulier, et qui peut offrir, si/quand nécessaire, des services additionnels similaires à ceux de l'état de l'art.

La contribution que nous proposons est une approche basée sur la relation entre : (1) une représentation de scénarios d'apprentissage adaptable et facile à utiliser sous la forme d'une table (qui est connue comme un artefact de manipulation facile), et (2) un modèle informatique sous la forme d'un arbre, qui permet notamment d'intégrer des services avancés. Cette proposition ne vise pas à remplacer les approches classiques basées sur un méta-modèle détaillé de scénario et une représentation en graphe. Son but est plutôt d'offrir un modèle alternatif simples, flexible et intuitif, à utiliser si/quand nécessaire.

Notre approche a comme inconvénients le fait que son modèle de base offre une expressivité limitée et qu'elle est permissive (i.e., ne contrôle pas les représentations des utilisateurs autant qu'une méta-modélisation le permet). Cependant, ces inconvénients peuvent être surmontés avec l'amélioration de ce modèle de base par des extensions. Dans cette recherche, nous avons considéré comme des extensions de notre éditeur les services suivants : l'instanciation de patterns de scénarios ; des mécanismes de simulation et de vérification de contraintes ; l'interopération de l'éditeur avec d'autres moyens spécialisés, comme les moteurs de workflow (pour représenter des structures complexes de planification) ou les plateformes de mise en œuvre (pour l'opérationnalisation et le suivi de sessions de scénarios).

De façon à étudier cette approche, nous avons développé un éditeur basé sur ces principes (l'éditeur ediT2) et avons effectué des expérimentations et des études exploratoires pour analyser comment les enseignants perçoivent et utilisent cet outil, en prenant comme objectifs/critères d'évaluation : (1) son expressivité pédagogique, i.e., le fait qu'un éditeur basé sur une table peut représenter une large gamme de scénarios d'apprentissage ; (2) sa facilité et son intuitivité ; (3) son expressivité informatique, i.e., si l'approche permet l'implémentation de services demandant des manipulations informatiques complexes, et finalement ; (4) sa flexibilité informatique, i.e., s'il est facile d'adapter l'éditeur à des besoins locaux. ediT2 a été initialement conçu de façon à offrir un ensemble standard de notions collaboratives. Dans une deuxième étape, nous avons considéré un contexte plus général pour augmenter ses possibilités d'intégration avec d'autres plateformes, pas forcément collaboratives, en permettant l'édition des noms des notions et de leurs attributs.

1.1 General context

A Computer Supported Collaborative Learning (CSCL) script is a learning scenario designed to enhance collaborative activities.

Computer-based means to manage learning scenarios can offer different services depending of their target public: design-oriented tools for instructional designers; editing-oriented tools for teachers; implementation-oriented tools for technical staff; and conducting-oriented tools for tutors.

If we consider general practices, i.e., teachers with basic technological skills and no particular CSCL methodology training, the use of these tools has not been really developed. One of the reasons is that languages/platforms should take into consideration that teachers expect educational tools to be easy to use and adaptable to their contexts (Williams et al., 2004).

The general issue we have considered in this thesis is how to offer to not-specifically-trained teachers tools to edit and operationalize CSCL scripts and, more generally, learning scenarios.

The contribution we propose is an approach based on relating: (1) an easy-to-use end-user representation in the form of a table, known as a device of easy manipulation by teachers, with; (2) a machine model of the script as a tree, as a way to reach advanced services.

The table-tree model introduces structural expressiveness and semantics which are limited but straightforward and intuitive. The table presentation enables direct interaction through natively-simple mouse-manipulations as in office suites (e.g., inserting a new activity/task in a script is achieved by the insertion of a new row in the table).

This proposal is not meant to replace the mainstream approach, which consists in offering a graph-based representation of scripts based on a meta-model representing in detail the different conceptual aspects of scripts. Rather, the proposed approach is meant to offer an alternative featuring simplicity, easiness and flexibility, to be used if and when needed.

The drawbacks of the proposed approach are that its basic model offers limited expressiveness and is permissive, i.e., does not control users' representations as much as a meta-model approach allows to. However, these drawbacks may be overcome by enhancing the basic model with extensions. In this work, we considered extensions related to the following complex services: instantiation support; constraints-checking and simulation mechanisms; interoperating the editor with other specialized means

such as workflow engines (to represent complex scheduling structures) or enactment frameworks (to operationalize and monitor script sessions).

To study this approach, we have implemented an editor based on this principle (the ediT2 editor) and conducted exploratory studies and experiments to analyze how teachers perceived and used such a tool. For this, we considered the usability ("is the model/system easy to use?"), perceived flexibility ("do teachers avail themselves from the flexibility provided by the approach?") and expressiveness ("does a table notation allow representation of a wide range of scenarios?") features. For these purposes, ediT2 was designed to offer a standard set of CSCL notions, i.e., "activity", "group", "participant", "resource" and "role". Usability was evaluated by in-lab experiments. The fact that the approach allows offering similar services to state-of-the-art systems was shown by implementing proofs-of-concepts. In a second phase, we considered generalization issues. We extended the initial ediT2 implementation to allow editing the notions and respective attributes to be offered by the editor, which increases the possibility of integration with other platforms, not necessarily CSCL featured. Although this thesis has its focus on CSCL scripts, the table-tree-based approach does present a more general interest.

1.2 Thesis organization

In Chapter 2, we recall CSCL scripts basics, discuss flexibility issues, and present the main characteristics and services offered by state-of-the-art languages and platforms.

In Chapter 3, we present in more details the considered issue in this thesis: the construction of a flexible CSCL script editor offering a friendly interface for teachers with basic technological skills and without special training; and which can offer, if/when necessary, additional services similar to that of state-of-the-art systems. We conclude this chapter presenting the adopted approach and the methodology used.

In Chapter 4, we present the proposed decoupling approach within which a scenario is modeled as a tree and presented as a table (T^2 model). A possible T^2-based script editor (named ediT2) is proposed and developed. We conclude this chapter discussing the semantics and constraints of the model, and some of its algorithms of manipulation.

In Chapter 5, we present the results of some studies and experiments performed to analyze the viability of the use of ediT2 to design and operationalize CSCL scripts. Usability, perceived flexibility and expressiveness are investigated.

In Chapter 6, we discuss and extend the basic ediT2 implementation through tree manipulations (still in correspondence with the T^2 model) to support, via add-ons, some advanced services existing in other learning editors/languages: instantiation

support, respect of constraints, representation of complex scheduling structures and simulation.

In Chapter 7, we continue to describe the extension of ediT2 through the development of a new service: the operationalization of the represented scenario on LMSs such as Moodle (from two different strategies – by middleware and by *ad hoc* means).

In Chapter 8, we describe why and how we extended ediT2 (still in correspondence with the T^2 model) to allow users to edit the names and respective attributes of the notions involved in the modeling of learning scenarios. We describe how we adapted ediT2 notions/attributes to allow users to: (1) operationalize ediT2 scenarios via a wiki platform; (2) represent preparation sheets; (3) model ediT2 scenarios compliant with another CSCL learning design editor, and; (4) introduce monitoring information in the ediT2 design interface.

Finally, in Chapter 9 we discuss the originality and application scope of our proposal, as well as its relation with other learning languages/platforms. Then, we conclude the thesis outlining some perspectives for the current and future works.

2 Representation and edition of CSCL scripts

2.1 CSCL scripts

2.2 Review of activities

2.3 Learning scenarios languages and platforms
2.3.1 General modeling-based perspective
2.3.2 CSCL modeling-based perspective
2.3.3 Pattern-based approaches
2.3.4 Operationalization-based approaches

2.4 Flexibility in CSCL scripts

2.5 Review of matters of concern and objectives

This chapter builds on and extends some ideas and concepts presented in our papers published in the ijCSCL (2012) and in the Computers and Education (2015) journals.

Résumé en français

La représentation et la mise en œuvre des scenarios d'apprentissage comportent différentes activités. Celles-ci peuvent impliquer différents acteurs et être conceptualisées et articulées de différentes façons.

Une première approche est de considérer les scénarios à partir de langages de modélisation pédagogique, c'est-à-dire, en prenant en compte, comme principe de base un langage général, comme, par exemple, l'IMS-LD.

Une deuxième approche est de considérer les notions spécifiques que l'on trouve dans les scénarios collaboratifs. MoCoLADe est un exemple d'éditeur de script collaboratif dont la représentation informatisée permet la communication avec d'autres plateformes à travers différents services, comme, par exemple, l'opérationnalisation et le monitoring.

Une autre approche est de proposer aux enseignants des patrons/gabarits de scénarios qui peuvent être instanciés et combinés. La plateforme WebCollage est un exemple classique d'un tel type de système.

Finalement, une quatrième approche possible est basée sur des langages instructionnels directement associés à des environnements spécifiques de mise en œuvre. Deux exemples sont : LAMS (qui permet aux enseignants de concevoir, partager, distribuer et monitorer des scénarios d'apprentissage à partir d'activités prédéfinies et d'outils de gestion de séquencement) et CeLS (qui permet aux enseignants de modéliser, réutiliser, opérationnaliser et monitorer des structures asynchrones de scénarios collaboratifs).

Ces schémas de représentation peuvent être analysés de différentes façons. Un regard transversal permet d'identifier les services généraux suivants :

- *Support conceptuel*. Les outils de représentation doivent permettre de travailler et de réfléchir sur le scénario (ou script).
- *Flexibilité du script*. Pendant l'édition du script, les enseignants doivent pouvoir ajouter facilement ou enlever des activités, changer l'allocation des ressources ou déplacer un étudiant d'un groupe à un autre. Cette flexibilité peut aussi être demandée en temps réel.
- *Support à l'instanciation*. Le système peut proposer un support à l'édition (semi-)automatisée à partir de patrons.
- *Respect des contraintes*. Pendant l'édition ou l'adaptation d'un script (avant et/ou après sa session), les enseignants peuvent avoir un support pour vérifier certaines contraintes.
- *Représentation de structures de planification complexe*. La plupart des scripts sont des constructions simples qui peuvent être représentées comme des listes d'étapes. Cependant, certains scripts ont besoin de

structures de planification plus complexes, comme, par exemple, dans des structures contenant des boucles ou des branchements.

- *Simulation*. Simuler un script avant sa mise en œuvre peut aider les enseignants à comprendre et à réfléchir sur la façon dont le scénario pourrait se dérouler dans des situations réelles.
- *Opérationnalisation*. La plateforme de mise en œuvre fournie aux étudiants peut être générée ou configurée à partir de la représentation du scénario.
- *Monitoring*. La représentation d'un script peut être la base pour le monitoring des actions des étudiants et pour son adaptation, si nécessaire, pour des nouveaux contextes/sessions.

2.1 CSCL scripts

Research in collaborative learning has shown that, in general, collaboration can increase group performance and individual learning outcomes. However, the fact that a setting presents a collaborative aspect is not sufficient for learning to occur (see (Slavin, 1996) for a review) because knowledge-generating interactions do not necessarily emerge spontaneously (Cohen, 1994; Salomon and Globerson, 1989).

To improve the likelihood of such interactions, two non-exclusive approaches have emerged.

One is to monitor the process of students and intervene if interactions are not occurring as expected. This may be based on launching regulative actions or interaction-analysis methods, i.e., analyzing what students are doing as they communicate and collaborate with one another as a basis for guiding collaborative behavior (Diziol et al., 2010; McLaren et al., 2010; Soller et al., 2005).

Another approach is to structure the setting by introducing the notion of scripting. A CSCL script is a learning scenario dedicated to a group of distant or co-present students whose actions or interactions are (at least partially) mediated by a computer-based system.

In the last decade, CSCL scripts have attracted considerable interest and the International Journal of Computer Supported Collaborative Learning called “Scripting in CSCL” a “flash theme” (Stahl and Hesse, 2007; Stahl et al., 2013). They have emerged as a key notion in the context of CSCL (Fischer et al., 2007), and have been supported by a broad range of studies highlighting how they foster learning activities (e.g., Baker and Lund, 1997; De Wever et al., 2009; Kollar et al., 2007; Rummel and Spada, 2005; Schellens et al., 2007; Schoonenboom, 2008; Slof et al., 2010; Stegmann et al., 2007; Weinberger et al., 2005, 2010).

Although a continuum exists, scripts are often dissociated in micro-scripts and macro-scripts (Dillenbourg and Tchounikine, 2007).

Micro-scripts are studied at a psychological level and aim at scaffolding students’ process at the interaction level, focusing on argumentation patterns and prompting students on the basis of a model of dialogue that is expected to be internalized by them (typically through the use of structured chats or argumentation graphical tools). As examples: make a student state a hypothesis and prompt a peer to produce counter-evidence; constrain interactions by prompting turn taking or imposing an argumentation grammar; etc.

Macro-scripts, on the other hand, are pedagogy-oriented large-grained scripts, based on indirect constraints generated by the definition of the sequence of activities or

the characteristics of the groups. Typically, to a group of students is given a task; this overall task is broken down into subtasks to be shared; to students are given different roles and resources; these subtasks and division-of-labor are studied to create a context within which students have to interact.

CSCL macro-scripts may be defined and implemented as in-presence scripts, involving students present in a same classroom, using different devices under the control of a teacher, or as on-line scripts, involving distant students, addressing the proposed tasks and communicating via a computer-based system only, under the control of an on-line tutor.

Examples of CSCL macro-scripts (from now "scripts", for short) are presented in Figures 2.1 and 2.2.

The setting involves three or four students who are going to work on a piece of text (e.g., a chapter). First, the teacher introduces different reading strategies: questioning oneself on the text that is being read, clarifying some issues, summarizing the text and predicting what will happen next. Then, the students must read the text. Afterwards, one of them, acting as a teacher, lists a certain number of questions to be considered. The students discuss these questions, and possibly raise some others. Afterwards, the student in the teacher role proposes an abstract; the group discusses it and modifies it until agreeing on it. Finally, the students make some predictions related to what will happen in the following stages of the text. [The scenario then continues with another piece of text and another student acting as the teacher]

Figure 2.1 A reciprocal-teaching script, adapted from (Palincsar and Brown, 1984 *apud* Kollar et al., 2006)

The scenario involves a group of four students. At the end of the process, they must produce a common document answering a set of questions. The four students are first given a common text introducing general principles related to energy saving. Then, two of the students are given a text focusing on insulation, and must produce a text listing different possible techniques. The two other students are given a text focusing on heating and, here again, must produce a text listing different possible techniques. The students are then put into pairs with one student that worked on heating and another who worked on insulation. They are given the document listing the insulation techniques, another document listing a set of questions, and together they must write a document answering these questions. (NB: The expectation is that, while writing, each will explain some things to the other). Similarly, the two other students are given the document listing the heating techniques and the same set of questions, and together they must write a document answering these questions. Finally, the four students are grouped. They must compare their two lists of answers and prepare a final team answer.

Figure 2.2 A jigsaw script, adapted from (Hernández-Leo et al., 2006)

2.2 Review of activities

Representing and managing learning scenarios involves different activities. These activities may involve different players and may be conceptualized and

articulated in different ways. There is no consensus on a precise life-cycle. However, analyzing different proposals (Dillenbourg and Tchounikine, 2007; Rodríguez-Triana et al., 2014; Tchounikine, 2008; Villasclaras-Fernández et al., 2009; Weinberger et al., 2009), the following general activities may be listed.

Design is the identification of the scenario's basic principles, i.e., how the featured tasks are supposed to lead to the intended learning goals, and important conditions that must be met. Depending on the context or institution, the design may be developed by instructional designers and/or teachers.

Editing is the adaptation and/or instantiation of the scenario to the subject being taught, the teaching context and the real students. For instance, using the jigsaw scenario to have students address energy-saving issues requires instantiation and editing actions such as deciding that expert groups will focus on "insulation" and "heating", deciding how many expert groups will be created and dividing the students, deciding how the students in the expert groups will be mixed to create the jigsaw groups and defining the resources provided to students during each phase (e.g., online documents during the expert phase and a quiz during the jigsaw phase). When editing scenarios, teachers must keep in mind both the scenario design rationale (e.g., creating jigsaw groups) and the local and contextual issues (e.g., number and characteristics of students or difficulty of the actual tasks). This may lead to adaptations (e.g., add intermediate activity). Edition is achieved by teachers. It is the core activity considered in this work. Teachers' basic practice is generally to reuse, adapt and instantiate existing scenarios designed by instructional designers.

Operationalization refers to deployment of the design on the technical enactment framework provided to students to perform the scenario tasks. Enactment platforms include educational modeling language players (e.g., Coppercore (2012) or LAMS (2013)), semi-specific architectures related to types of scenario (e.g., CeLS (Ronen et al., 2006) for collaborative scenarios) and specific architectures (e.g., jigsaw-specific architecture implementing specific principles such as grouping mechanisms (Dillenbourg and Hong, 2008)). In actual practice, general frameworks (e.g., wikis) and Learning Management Systems (LMSs; e.g., Moodle (2012)) are frequently used. Operationalization is not always considered: Learning Scenarios Editors (LSEs) may be used as a way to prepare sessions only.

Orchestration (Dillenbourg et al., 2013) refers to management of the session. While students enact the scenario, teachers may conduct the session (e.g., clarify instructions or provide support) and can also have to manage events requiring run-time adaptations of the scenario (e.g., add a task, change a resource or move a student from one group to another).

2.3 Learning scenarios languages and platforms

The next subsections present a brief description of some general and specific approaches to the representation and operationalization of learning scenarios and, in particular, CSCL scripts. This list is not exhaustive. Here, the objective is just to differentiate different types of approaches and mention some representative systems in the current state-of-the-art in LSEs. Other interesting instructional languages/platforms will be mentioned, in context, in the next chapters.

2.3.1 General modeling-based perspective

A first approach is to consider scenarios from a general Educational Modeling Language (EML) perspective, i.e., taking as a basis a general modeling language.

The prototypical example is IMS-LD (2012), the Learning Design specification released in 2003 by the Instructional Management Systems Global Learning Consortium (IMS GLC, 2013). Its objective is to help learning designers to model and share a large spectrum of learning scenarios (not only specific to CSCL) from the issues presented in existing proposals as, for instance: pedagogical flexibility, reproducibility, interoperability, compatibility, reusability and formalization (Lejeune, 2004).

With this purpose, IMS-LD is scaffolded by: (1) the independence of pedagogical approaches. (2) A set of general notions trying to cover a wide variety of learning scenarios (from the “activity”, “role”, “environment”, “learning objects”, “learning objectives”, “prerequisites” and “service” notions). (3) A conceptual separation between scenarios and their respective resources and services. (4) A computational language as a support to represent learning activities (XML based), cross-validated by three compliance levels as a way to facilitate, when needed, the development of scenarios more sophisticated: level A, describing the scenario itself; level B, adding properties (about users) and conditions (to improve the control flow and the representation of complex structures of decision) to level A; and level C, adding notifications (when events take place) to level B.

The first IMS-LD tool to implement all three levels of the IMS Learning Design specification was developed at the Open University of the Netherlands: CopperCore (2012). Coppercore is an open-source initiative offering an engine, that may be incorporated to other pedagogical projects using the IMS Learning Design specification; and a player, which final result (a tree-based structure) can be visualized on browser technologies.

Another open-source architecture implementing the three levels of the IMS Learning Design specification is the ReCourse LD Editor (ReCourse, 2013), conceived

from the TENCompetence European project (ended November 2009), which aimed to support lifelong learning development in Europe (TENCompetence, 2013).

ReCourse extends the CopperCore engine and improves the open-source RELOAD project (an IMS-LD form-based editor; RELOAD, 2012) in such a way to solve some of their issues as, for instance, the limitation in the number of services provided (as well as their execution in real-time), and the XML tree-format or form-based structures that do not totally hide to their users the complexity of the IMS-LD specification.

There are other IMS-LD compliant proposals allowing developing their IMS-LD learning scenarios through graphical environments. Examples are the MOT+LD, OpenGLM and TELOS platforms, where the first and the last have been developed at the LICEF Research Center, from the MOT concept map editor; and the second at the University of Vienna, from the Graphical Learning Modeller platform (GLM).

MOT+LD is a MOT specialization for the modeling of IMS-LD level A. Whether levels B and C are required, pedagogical scenarios designed from MOT+LD could be complemented and delivered via other IMS-LD compliant environments, as ReCourse or RELOAD, for instance (Paquette et al., 2008).

The OpenGLM is a learning design application authoring the levels A and B of the IMS-LD specification, and providing to its users an opened repository of resources (tools/materials) and teacher methods (as the jigsaw template presented in Figure 2.2, for instance). In the same way to the MOT+LD platform, whether the level C is required, it should be necessary another IMS-LD complaint player to complement and delivery scenarios designed from OpenGLM (Derntl, 2011).

Finally, the TELOS ontology-driven platform is a MOT+LD specialization extended by the MOT+OWL architecture and the MISA LD instructional engineering method for learning design, which reaches all three levels of the IMS-LD specification (Paquette, 2010). The scenarios produced by this platform can be operationalized on the platform itself, or delivered on other IMS-LD complaint players.

2.3.2 CSCL modeling-based perspective

With respect to CSCL scripts, a general conceptual framework has emerged (Kobbe et al., 2007). This framework has been elaborated as a consensus by scientists from several fields (educational, cognitive and computer sciences) as result of their experiences based on the analysis of effective scripts, and has proven to allow for modeling a variety of scripts of different natures. For these reasons, we reused the notions which it highlighted in our work.

In this framework, scripts are modeled as a combination of components (participants, groups, activities, roles and resources) and mechanisms (group formation, task distribution and sequencing).

Components denote the structural elements of a script. Participants represent the students (and teachers or tutors, if any) involved in the script. Depending on needs, participants may be represented by information such as name, age, gender, nationality, profile, knowledge, skills or preferences. Participants may be involved in groups (that can evolve during the script execution and can themselves be broken down into sub-groups). Activities denote what must be done during a script (the tasks and subtasks), and can be represented by instructions from a low to a high level of abstraction. Students can be prompted to interpret one or several roles, and can change role during the script unfolding. Roles can also be associated with a group. Finally, resources denote the virtual or physical objects used, modified or produced by the groups/participants during the script execution.

Mechanisms describe the dynamic of a script, and how components are related to each other. Group formation specifies the way groups of participants are constructed. Grouping can be conducted by listing participants by name, using rules related to static data (e.g., grouping by age or skills) or to dynamic data (e.g., students who have finished a given activity). Task distribution specifies how components (e.g., activities, roles or resources) must be distributed among participants or groups. Finally, sequencing specifies how the script's phases or tasks will be distributed over time. This can correspond to a simple linear ordering (phases or tasks are to be taken one after the other) or a complex dynamic structure using some sequencing pattern as traversion, rotation, fading or a combination of these.

MoCoLADe (Harrer et al., 2007) (standing for Model for Collaborative Learning Activity Design) is an example of visual CSCL script editor reusing this schema (another work using this same framework is XSS (Stegmann et al., 2009)). In this platform, collaborative scenarios are represented by statechart-based structures. The movement of tokens through such structures enables simulations, e.g., verifying if edges conditions (e.g., group size), or time limits, are satisfied or not.

The MoCoLADe representation, when exported to an IMS-LD format, can be interfaced with CopperCore (or other IMS-LD compliant tools, as the RELOAD editor, for instance) in a way to provide to teachers different services as operationalization (Harrer et al., 2007) and monitoring (CopperCore is capable to inform at run-time the actual state of a script enactment (Malzahn et al., 2008)).

MoCoLADe can be interfaced with other computational platforms thanks to its machine representation, i.e., the internal representation that can be directly interpreted by a computer and run, or transformed into such a runnable representation. For instance, MoCoLADe designs can be operationalized on the CeLS learning platform (see Section 2.3.4).

As an example of another CSCL-specific model, Haake and his colleague define atomic scripts by an 8-tupla ("roles", "actions", "states", "transitions", "start", "end", "input document", "output document") and composed scripts as atomic script combinations (Haake and Pfister, 2007).

2.3.3 Pattern-based approaches

Another possible approach to scenarios' representation is to offer teachers high-level scenarios' patterns that they can adapt and instantiate. The use of patterns is gaining interest in Technology Enhanced Learning (TEL) in recent years (Goodyear and Retalis, 2010).

WebCollage (Villasclaras-Fernández et al., 2013) is a web environment based on Collage (Hernández-Leo et al., 2006), a collaborative learning graphical authoring platform conforms to the IMS-LD level A specification (both developed at the GSIC team, University of Valladolid). Collage scaffolds teachers through a methodological approach from the creation (or reuse) of collaborative learning scenarios via the utilization of pre-defined educational templates (or patterns), considered as being of good practice in the CSCL community (Hernández-Leo et al., 2006, 2010). Such patterns can be used individually, or combined in a hierarchical way, to compose new scenarios according to teachers' pedagogical intentions. A classic example of CSCL pattern is the jigsaw script, presented in Figure 2.2.

As it takes place in the MoCoLADe environment, scripts modeled in WebCollage can be operationalized with CopperCore. Another possibility is the use of the GLUE!-PS middleware platform (also developed by the GSIC group; Prieto et al., 2011), which provides ready-to-use adapters to semi-automatically deploy learning designs expressed in several design languages, including WebCollage, to different enactment frameworks, including Moodle (GLUE!-PS maps the notions of each source learning design language into the notions of each target enactment platform).

2.3.4 Operationalization-based approaches

Another possible approach to scenarios representation is to offer an instructional language directly associated with a specific enactment environment. LAMS (LAMS, 2013) and CeLS (Ronen et al., 2006) are general and CSCL-specific examples, respectively.

LAMS (standing for Learning Activity Management System), is an open-source platform inspired from the IMS-LD ideas, but not compliant with its specification. It has been developed through an effort of collaboration between the LAMS Foundation, the LAMS International and the Macquarie University (LAMS, 2013).

LAMS enables teachers to perform on a single intuitive and friendly web-based authoring environment the design, sharing, delivering and real-time monitoring of collaborative learning scenarios from pre-defined activities (e.g., assessment, chat, forum, spreadsheet, voting or wiki), and sequence managing tools (e.g., stop point (a synchronization element), grouping or branching).

LAMS can be used with a stand-alone proposal or in conjunction with a LMS as, for instance, Moodle (Moodle, 2012), Sakai (Sakai, 2013), Blackboard (Blackboard, 2013) or .LRN (.LRN, 2013). In this case, both platforms become accessible through a single sign-on, and LMS courses, and respective users, being automatically imported to LAMS.

CeLS (standing for Collaborative e-Learning Structures) is another web-based authoring platform, developed at the Holon Institute of Technology, and aiming to enable teachers to model (from scratch), reuse (and adapt, if necessary), operationalize and monitoring (without interference) structured asynchronous CSCL scenarios through a single flexible learning environment (Ronen et al., 2006).

The CeLS script structure is conceived from the arrangement of stages composed by CeLS objects (presentation, input, interaction and communication) and properties (indicating grouping configuration by object and/or stage). Also, it is possible to assign to each stage some conditions as, for instance, “start”, “advance” (when a stage finishes) or “end”. Different students may visualize different flow of activities. In this way, the pedagogical scaffolding of a CeLS scenario is obtained from such a combination (properties, conditions, order and/or composition of each one of its stages).

CeLS also presents three important characteristics: (1) an output produced by a stage can be reused *a posteriori* by another one in the sequence; (2) a CeLS scenario may be modified at run-time (teachers can add/change objects on its stages, or add new stages on it), and; (3) CeLS can be used as an independent tool (as described just here), in association with a LMS (e.g., Moodle; Ronen et al., 2006), or articulated with other LSEs (Kohen-Vacs et al., 2011), as MoCoLADe, for instance (Harrer et al., 2009).

2.4 Flexibility in CSCL scripts

Although CSCL scripts have proven effective in promoting productive interaction and student learning (Wecker et al., 2010), script design treads a fine line between useful guidance and control of student activities. If the scaffolding it provides is too weak, the script will not produce the expected interactions. Yet, if it is too strong or irrelevant, it can hinder interaction (Dillenbourg, 2002). Therefore, an important requirement for managing CSCL scripts is flexibility.

What students must be prompted to do and how they should be supported must not be decided once for all before the session only (Dillenbourg and Tchounikine, 2007). CSCL scripts must be flexible in order to manage unexpected events at run-time, taking into account the students’ actual activity (e.g., a subgroup fails a task or two students unproductively conflict), or to seize some teaching opportunities (e.g., given

the group's dynamics, adding some additional activities or re-orientating the students' current process to engage the group in exploring interesting issues).

Students may also ask for some flexibility, for example by asking to change a group's composition, or still to divide or to postpone some task. In such a case, the teacher must reflect on the request and its consequences with respect to the rationale of the initial script in order to accept the request (i.e., adapt the script as requested by students, or in a related way) or refuse it (typically, because this conflict with the educational purpose of the original design decisions).

As an example, let us consider the jigsaw script presented in Figure 2.2. The rationale of the script is that in order to answer the questions, the students will have to share the knowledge they developed during the first stage, and thus get involved in explanations, questioning or argumentation. The fact that students who have read different texts are paired during the jigsaw phase is an *intrinsic constraint* (Dillenbourg and Tchounikine, 2007), i.e., a constraint bound to the script design-rationale that should be respected in order to remain consistent with the script's principle and conditions that have been identified as enhancing learning (e.g., "make students with different knowledge interact"). In direct contrast, the way a particular student who worked on heating is paired with another one who worked on insulation is an *extrinsic constraint*: it is contingent and can be changed without affecting the scripts' rationale if, for instance, some students experience difficulties working together. Such dissociation between intrinsic and extrinsic constraints is to be seen as a basis for guiding teachers' decision-making process.

Considering scripts as control devices, the fact that scripts should be adaptable (the requirement for script flexibility) has different implications according to the operationalization approach adopted.

One approach is to consider the script on one hand and the enactment framework on the other. In this case, the script is an artifact that helps teachers to think about the session, elaborate a plan and adapt the plan according to what happens during the session, and that helps students in structuring their process. However, it is technically disconnected from the enactment framework. This is typically the case when the students are supposed to use tools they can choose themselves, or a predefined technological framework that is not adaptable (or that one will not attempt to adapt), such as LMSs and wikis (and, of course, if they do not use any technological framework). In such a case, a formal representation of the script is a support for thinking. The requirement for scripts flexibility is thus that, as a support for teachers to plan and manage the session, and as the origin of the instructions and hints students will be given, the script as it has been edited and set up by the teachers should be adaptable.

Another approach is to automatize or partially automatize the relationship between the scripts' representation and the provided enactment framework and, in particular, provide teachers representation schemes that allow generating or adapting the enactment framework and, at run-time, adapting the script's representation (and,

following, the enactment framework) according to the script's effective unfolding (Tchounikine, 2008). In such a case, a formal representation of the script is a means for its operationalization. The requirement for script flexibility is that teachers must be provided a representation scheme which makes it possible to modify the enactment framework through the representation of the script. This raises technical issues such as adapting the interfaces students are given but, also, managing coherency issues, such as those related to the dataflow.

2.5 Review of matters of concern and objectives

Representation schemes can be analyzed in different ways such as their communication and creativity features (Botturi et al., 2006) or their "informedness" and computable dimensions (Harrer and Hoppe, 2008). Analyzing existing propositions in a transversal way, the following concerns and objectives may be highlighted:

- *Conceptual support*. This is a basic common objective. A script is a complex artifact, within which different and potentially conflicting constraints are to be integrated. An explicit representation is a means for instructional designers or teachers to reflect on the script being designed, edited or shared. Design of representation means (approaches, languages, editors) may thus consider offering a particular type of users relevant representation tools to work and reflect on the script independently from any enactment framework considerations (IMS-LD is an example). Providing a language/editor featuring relevant notions has a value in itself (Harrer and Hoppe, 2008).
- *Script flexibility*. While editing the script, teachers must be allowed to easily add or remove activities, change resource allocation or move a student from one group to another (Dillenbourg and Tchounikine, 2007). This flexibility may also be required at run-time.
- *Instantiation support*. A script representation may be considered as a basis for automated editing support. Important aspects of scripts are group formation and task distribution (Kobbe et al., 2007). Distributing students and tasks is easy when limited to few items, but becomes tricky if large numbers are involved.
- *Respect of constraints*. While editing or adapting a script (before and/or during the session), teachers have to consider both the script's intrinsic and extrinsic constraints (Dillenbourg and Tchounikine, 2007) through constraint-checking mechanisms.
- *Representation of complex scheduling structures*. Most scripts are simple constructions and can be represented as a list of steps to be taken one after the other (Haake and Pfister, 2007). However, some scripts require

more complex scheduling structures such as branching, parallelism or loops (Kobbe et al., 2007; Roschelle et al., 2009).

- *Simulation*. Simulate a script before it is operationalized may aid teachers to understand and reflect about how such a script could be unfolded in actual situations (and then, could support them in decision-making processes). Some authors have argued that simulating the script on a workflow engine before introducing it to students provides useful input to analyze and tune the workflow and dataflow (Harrer et al., 2007).
- *Operationalization*. The enactment framework provided to students may be generated or configured from the script description.
- *Monitoring*. At run-time, during the script's unfolding, the script's representation may be a basis for teachers to monitor students' processes and adapt the script, if necessary, for a new context/session or on-the-fly (Malzahn et al., 2008; Soller et al., 2005).

3 Problematic, adopted approach and methodology

3.1 Considered issue

3.2 Research question

3.3 Adopted approach

3.4 Historical dimensions and methodology

Résumé en français

Différents utilisateurs et différentes pratiques demandent différentes possibilités. A notre avis, ce n'est pas une bonne stratégie de chercher à créer des moyens de représentation qui présentent, simultanément, des caractéristiques d'intuitivité, de flexibilité, de complétude (i.e., les moyens qui permettent la modélisation de tous les aspects d'un scénario) et de calculabilité (i.e., les moyens qui permettent que la représentation soit directement interprétée par un ordinateur ou qu'elle soit automatiquement utilisée pour configurer une plateforme de mise en œuvre).

La complétude présente des avantages, tels que, par exemple, la création d'une *lingua franca* qui facilite la création de répertoires, l'interopération et le partage d'expérience. Cependant, la complétude présente aussi de la complexité et des contraintes. La simplicité et la flexibilité sont plus en ligne avec la vision de scénarios comme des structures complexes dont l'édition peut demander des démarches d'élaboration. Par rapport à la calculabilité, la définition d'une représentation informatisée demande la prise en compte des contraintes technologiques et, en conséquence, les caractéristiques de flexibilité sont, par définition, limitées par ces questions.

Si le développement de plateformes spécifiques ou de langages de mise en œuvre est pertinent, ce type de mise en œuvre est, dans les pratiques, marginale. Actuellement, la vulgarisation des Technologies de l'Information et de la Communication va plutôt dans le sens du développement de solutions à partir de l'utilisation des outils standard ou d'adaptations locales de plateformes génériques, avec des utilisateurs choisissant leurs technologies en fonction de leurs contextes et leurs perspectives.

Dans cette optique, nous avons étudié une approche originale dont le principe est le suivant : au lieu de tenter concilier des exigences contradictoires au sein d'une même représentation (avec le risque de générer des interfaces peu intuitives ou très complexes), nous proposons une représentation répondant à l'exigence qui constitue le goulet d'étranglement (comment offrir aux enseignants une représentation facile à utiliser) et à partir de laquelle d'autres services (par exemple, le support à l'instanciation d'un scénario à partir de mécanismes automatisés) peuvent être développés et intégrés.

3.1 Considered issue

Although the introduction of EMLs is recognized as an important contribution to TEL, mainly due to their attempt to describe learning scenarios through (semi-)formal graphical syntax (generally from UML diagrams and/or XML representations) (Botturi et al, 2008; Larnaca, 2012; Tchounikine et. al, 2009), these languages have not been widely adopted by either practitioners or institutions (Derntl et al., 2011).

One of the reasons is that EMLs often address (or are used as if they were relevant to address) a plurality of objectives. A first consideration is that EMLs may be used for different purposes, including elaborating the scenario (instructional design); adapting existing scenarios to particular settings; configuring the enactment framework presented to students; and reflecting on how students enact the scenario. Although these actions all require a representation of the scenario, the requested characteristics are partly different.

As a consequence, design choices related to one objective may directly impact, or be poorly adapted, to others, and hinder usage. Another consideration is that EMLs were originally thought of as modeling languages to be used by professional modelers (instructional designers). In many cases, however, they are used by teachers, whose concerns and skills are different.

Instructional designers may benefit from specific training, and are interested in means to specify scenarios or index them in repositories. Teachers, on the other hand, generally do not have any specific modeling training, and are mainly interested in means to adapt scenarios and deliver them effectively in real classrooms (Weinberger et al., 2009). Complexity of languages/platforms and their use in “basic” settings is an issue to be considered.

3.2 Research question

Taking into account this evidence, the research question considered in this work is: can one design and implement representation means that (1) are sufficiently simple and flexible to allow users (teachers) with basic ICT (Information and Communication Technology) skills and no particular methodology training to edit scripts and adapt them to their view and context, and; (2) may be extended to match other needs and implement advanced features offered by other classical languages/platforms such as simulation, checking constraints, supporting complex grouping mechanisms

and scheduling structures, configuring the students enactment framework or monitoring the script's unfolding?

The rationale for considering this research question, and considering it in this way, can be summarized as follows:

- Although usually not in charge of the script design, teachers, as the persons in charge of reflecting on and managing the classroom (the students, the overall activities, the institutional context) and the script enactment, may engage in substantial adaptations before the session (while instantiating the script) and, possibly, at run-time (while reflecting on adaptations).
- Editing a script is a possibly complex task, requiring iterative refinements.
- Offering easy-to-use and flexible means is an important feature to allow teachers to adapt the script to their context (the setting, the domain, the students, etc.) but, also, to their perspective and practices, and is likely to facilitate appropriation and usage.
- Most current works related to operationalization consider the link with the enactment framework as a first class concern, which leads to consider easiness and flexibility within the space of what is left open given the comprehensiveness and computability specifications.

3.3 Adopted approach

Different users and different usages require different means. In our opinion, it is not a good strategy to create representation means that simultaneously allow for comprehensiveness, ease of use, flexibility and computability. Languages and editors are tradeoffs. Complexity and inflexibility are generally not design decisions, but a consequence of constraints related to some other objectives.

Furthermore, constraints and complexity are increased by considering comprehensiveness (i.e., means to allow modeling all aspects of a script as, for instance, allowing the representation of all complex grouping mechanisms) or computability (e.g., allowing the representation to be directly interpreted by a computer to "play" the script and indicate the activity flow, or to be used to automatically configure an enactment framework).

With respect to completeness, this property is due to if targeting a kind of standard (e.g., in the way IMS-LD attempts to be a standard for Instructional Design in general). Completeness has advantages, such as dealing with industrialization issues or creating a *lingua franca* that facilitates repository creation, interoperation and experience-sharing. However, completeness comes with complexity and constraints.

Simplicity and flexibility are more in line with viewing scripts as complex structures whose edition may require an elaboration process, i.e., drawing a general structure, refining some aspects of it, testing an option, and reversing decisions until the designer (here, a teacher) considers the built artifact satisfactory. If necessary, simple representations can be complemented with a more comprehensive or detailed language/platform: consider how many Instructional Design frameworks or technical platform meta-models are mapped from or onto IMS-LD (cf. Section 2.3).

With respect to computability, defining a computable representation requires taking into account technology-related constraints and, as a consequence, flexibility features are by definition limited by these issues. This is a hard constraint. The development of script engines (Dillenbourg and Tchounikine, 2007) integrating features such as automatic configuration of the enactment framework from the script representation and automatic management of the script's unfolding is an interesting research question, which can certainly advance the field of CSCL.

However, not misunderstanding the interest of implementing CSCL settings via operationalization languages or specific platforms, this is far from being the major implementation approach, and may not be the pattern that is going to generalize. Currently, the spread of ICT technologies rather goes into the direction of the implementation of CSCL settings as mash-ups of off-the-shelf tools (e.g., freely downloadable and interoperable communication tools or resource-sharing tools) or local adaptations of generic platforms such as LMSs, with users mobilizing technologies according to their perspectives and contexts (Jones et al., 2006; Tchounikine, 2011).

Within this perspective, we have studied an innovative approach which rationale is as follows: instead of attempting to conciliate conflicting requirements within a single representation (with the risk of over-complex or poorly-intuitive interfaces), we propose to adopt a representation that fulfills the bottleneck requirement (offering teachers an easy-to-use representation) and from which other services identified as of interest (e.g., supporting instantiation and adaptation of the scenario by automated mechanisms) may be implemented and incorporated.

This approach is based on a technique that consists in using a pivotal-model that allows different representations and means for different objectives: an easy-to-use end-user representation of the scenario in the form of a table is put into structural correspondence with a machine representation as a tree. Modeling the scenario as a tree allows implementing the targeted services via model analyzes and transformations.

3.4 Historical dimensions and methodology

Initially, we developed a pivotal-model, named T^2, from a structural correspondence between a visual language (described by a table as a device of easy

manipulation for teachers) and a machine-readable representation (described by a tree data structure, which manipulations may be implemented by reusable algorithms of standard practice).

Next we developed, as a proof of concept, a CSCL-focused editor, named ediT2. This editor offers a table interface as a way to manipulate learning scenarios through natively-simple mouse-manipulations as in office suites (e.g., inserting a new activity/task in a script is achieved by the insertion of a new row in the table).

We then performed some studies and tests to analyze if the editor was usable and how teachers perceived its features.

Thanks to the pivotal-model approach, we enhanced ediT2 through the development of advanced services identified as of interest in the domain: instantiation support, respect of constraints, representation of complex scheduling structures, simulation, operationalization and monitoring.

Finally, due to some observations collected in the experiments and explorative studies performed, we extended ediT2 again (however still in correspondence with the T^2 model) to enable users to edit the notions (and respective attributes) needed to model and operationalize, when required, their learning designs.

Overall, the contribution of the work is an approach that allows designing easy-to-use and adaptable LSEs. Four objectives / evaluation criteria were considered: (1) pedagogical expressiveness, i.e., can table-based editors represent a wide range of scenarios?; (2) usability, i.e., do teachers find the editor easy to use and intuitive?; (3) computational expressiveness, i.e., does the approach allow implementation of complex services?; and (4) computational flexibility, i.e., is the editor easy to adapt to local needs?

The general requirements and services to be offered were identified by considering the state-of-the-art of LSEs. The pedagogical expressiveness of the table representation was tested by studying how the editor allowed representing different types of scenarios. Easiness-of-use was tested via usability tests. Computational expressiveness and computational flexibility were demonstrated by proofs-of-concept developments.

4 The T^2 model and the ediT2 editor

4.1 General considerations

4.2 The table/tree structure

4.3 Proof of concept – the ediT2 editor

4.4 Semantics

4.5 Offered flexibility

4.6 Constraints
4.6.1 The allocation of items into table cells
4.6.2 The script-structure
4.6.3 The 1-n relationship

4.7 Adding control rules

4.8 T^2 tree manipulation algorithms

This chapter builds on and extends some ideas and concepts presented in our papers published in the ijCSCL (2012) and in the Computers and Education (2015) journals.

Résumé en français

Le modèle que nous proposons dans cette recherche concerne une approche intentionnellement simple de l'édition et de l'adaptation de scénarios d'apprentissage. Tout d'abord, nous prenons comme point de départ et ligne directrice un langage/éditeur basé sur une table en tant que dispositif de structuration basique et commun. Ensuite, nous fondons la sémantique du modèle sur la structure de table plutôt que sur un méta-modèle spécifique. Cette approche n'est pas présentée comme une proposition de remplacement d'autres approches plus classiques basées sur des graphes et avec des soutiens sémantiques, mais comme une alternative caractérisée par la simplicité, la flexibilité et la capacité d'extension, à utiliser si et lorsque pertinente, ou en complément.

Nous gérons cette spécification en mettant l'interface (la table) en relation structurelle avec un modèle informatique interprétable (un arbre). Ce principe permet la manipulation d'un scénario pédagogique à partir de manipulations naturellement simples (manipulations de la souris sur le contenu et la structure de la table, comme dans des outils bureautiques), au lieu d'élaborer un langage riche et complexe hypothétiquement rendu facile à utiliser par les utilisateurs grâce à des interfaces « intelligentes ». Une telle table est composée de colonnes et de lignes (lesquelles peuvent être fractionnées en sous-lignes), quand on l'envisage comme interface pour l'utilisateur, et comme une structure n-aire composée de niveaux et de branches, quand on l'envisage de façon formelle comme un arbre. Ce modèle (appelé T^2) introduit trois contraintes structurelles :

- Allocation des items dans des cellules de la table : un item défini comme une notion N peut seulement être déposé dans la colonne correspondante de la table (la sémantique des notions n'est réalisée que par les labels des colonnes de la table et, par conséquence, liée à la perspective de l'utilisateur) ;
- Structure du script : la structure de table implique que tous les composants d'un script aient la même représentation ;
- Relation 1-n : un composant d'un script doit respecter la relation 1-n ($n \geq 1$), afin d'éviter des constructions ambigües.

Du point de vue de l'utilisateur, l'avantage de la structure de la table est d'éviter des constructions syntactiques complexes. En considérant la perspective informatisée, l'avantage est similaire. Des manipulations sur un script correspondent à des manipulations sur des branches et des nœuds. Les manipulations des utilisateurs sont interprétées par rapport à la structure de la table et rapportées dans la représentation en arbre (ou inhibées en cas d'incompatibilité avec le modèle). Du point de vue informatique, la table et l'arbre respectent une correspondance structurelle, c'est-à-dire, un élément du modèle correspond à un élément de la table et vice-versa.

Dans ce chapitre, nous présentons le modèle T^2 ainsi que la conception et le développement d'un éditeur de scénarios collaboratifs spécialement conçu à partir de ce modèle: l'éditeur ediT2.

4.1 General considerations

Our model approach (named T^2) is an intentionally simple model created to allow easy edition and adaptation of scripts, in such a way to outline two of the eight matters of concern presented in Section 2.5 (conceptual support and script flexibility), in addition to the representation flexibility capability, which concept will be introduced afterwards, in Section 4.5. The other six of them (instantiation support, respect of constraints, representation of complex constructions, simulation, operationalization and monitoring) will be discussed in Chapters 6, 7 and 9, when we will present the possibility of the extension of such a model through the inclusion of advanced services.

Not misunderstanding the interest of the approaches listed in Section 2.3, our work explores an alternative built on different premises. First, we take as an entry point and governing design decision offering a language/editor based on a table, as a basic and very common structuring device. Second, we consider basing the model semantics on the table structure rather than on a proper meta-model. This approach is not proposed to replace more classical graph-based and more semantically-supporting approaches but as an alternative, featuring simplicity, flexibility and extension capabilities, to be used if and when pertinent, or in complement.

We manage these specifications by putting the interface (the table) into relation with a machine-readable model (the tree). This principle allows addressing the manipulation of the script via natively-simple manipulations (mouse-manipulations of a table structure and content, as in office suites), instead of elaborating a rich and complex language hypothetically rendered easy to use by users thanks to smart interfaces.

4.2 The table/tree structure

The T^2 model proposes to consider a CSCL script as (1) a *table* composed of *columns* and *rows* (which can be broken down into *sub-rows*) if one addresses it via its user-oriented visualization, and (2) an *n-ary tree* composed of *levels* and *branches* if one addresses it as a formal structure. The model is named T^2 to denote this double Table and Tree structure.

From an end-user perspective, the advantage of the table structure is to avoid complex syntactical constructions, the constraints of which must be understood and properly used. From a machine perspective, the advantage is similar. Natively, script manipulations correspond to manipulations of branches and nodes. They are

interpreted with respect to the table structure, and reported in the tree representation (or inhibited if inconsistent with the model). From a Computer Science perspective, the table and the tree respect a structural correspondence, i.e., one element of the model corresponds to one element of the table interface and vice versa.

Figure 4.1 presents the jigsaw script described in Figure 2.2 as a table, and Figure 4.2 the corresponding tree representation, which provides a general perspective on the scripts' complexity and structure. The number of table rows and sub-rows is denoted by the tree's width, and the number of table notions by the tree's depth. A fictive root is added to the tree to tie together the first level nodes.

Activity	Group	Participant	Resource
Read the general text	Class	P1,P2,P3,P4	General text(in)
Identify techniques	G1	P1,P2	Insulation text(in), Insulation list(out)
	G2	P3,P4	Heater text(in), Heater list(out)
Crossing groups	G3	P1,P4	Insulation questions(in), Insulation list(out)
	G4	P2,P3	Heater questions(in), Heater list(out)
Regrouping	Class	P1,P2,P3,P4	Answers(out)

Figure 4.1 A jigsaw script (Figure 2.2) as a table

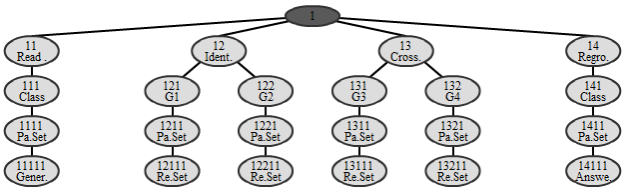


Figure 4.2 The jigsaw script (Figure 4.1) as a tree

In Figure 4.2 each child in the first tree level corresponds to each cell in the first table column (Activity column). "Read the general text" and "Regrouping" activities have a single child (branching factor = 1), whereas "Identify techniques" and "Crossing groups" activities have two children (branching factor = 2). In this case, none of the Participant and Resource cells are further broken down, which means that their corresponding sub-trees are linear (branching factor = 1).

In this representation, we will call *script-structure* the ordered list of notions used as columns (e.g., in Figure 4.1, Activity-Group-Participant-Resource); *pivotal notion* the notion used as the first column and that defines the modeling commitment (e.g., in Figure 4.1 the script is modeled as a set of Activities); *script-component* a table row, referring to it by its pivotal notion (e.g., the "Read the general text" script-component); and *items* the particular values attached to the cells or, within the tree perspective, to the nodes.

In the basic implementation, each item is represented by a label (Name) plus an optional textual description (Description). This makes possible, for instance, to

represent a resource using an identification name along with a textual description defining an instruction, a configuration data or a link to a document/external tool (e.g., in Figure 4.1, the “Class” group could have “Class” and “All class students must be involved in this activity” as its name and description fields, respectively).

As an abstract model, T^2 can be used as a basis for different objectives. For example:

I. Usage as a general conceptual model. The table/tree structure can be used as a convenient and simple model to describe CSCL scripts. Such a usage does not require any specific technical implementation. The table representing a script can be edited with basic tools (e.g., a text editor offering table-related features) and, of course, with a pencil, an eraser and a piece of paper. The model can also be used to represent (part of) scripts implemented within specific tools or platforms, as a simple intermediation means. In such cases, however, coherence issues and computability are not supported.

II. Usage as a basis to design script editors. A T^2 editor is natively a table-like interface. The implementation may be addressed in a variety of ways, and to different extents, for instance:

    a. The editor provides the means to edit and adapt a table, i.e., proposes features to create and modify columns and rows, to drag-and-drop items from one cell to another, and to split and merge rows. However, this is of little interest since such features are proposed by many existing tools such as tables in text editors. The advantage can be in the interoperation of this editor with other components of the educational setting (see here below).

    b. The editor provides the means to edit and adapt a table in a way that respects the structural constraints of the model, i.e., (1) actions that would lead to incoherent constructions (in the sense of the model) are made impossible and (2) the implications of modifications that can be derived from the model are automatically generated and propagated in order to keep the script (table) coherent. Such an editor provides support for editing scripts and maintaining their coherence as instances of the T^2 model.

    c. The editor provides the means to edit and adapt a table in a way that (1) respects the structural constraints of the model, and (2) offers advanced features such as checking specific constraints (for example, script specific considerations such as the intrinsic constraints of a jigsaw script), exporting the representation into a format readable by another software component (for instance, a simulation framework or an enactment framework), or monitoring the script’s unfolding (e.g., in a teacher-oriented approach, the script representation could be completed at run-time in accordance with the students’ advancement).

The specific interest of the T^2 model is that it is natively based on a table structure, a representation which basic users are highly familiar with, and which can be easily adapted using direct manipulations (merge, split, drag-and-drop, copy-and-paste or duplicate). This allows for a rather straightforward specification for usages such as an editor or a monitoring device. In such cases, a T^2 editor may be constructed as a standalone tool, or as a software component embedded or interoperated with some other components or ICT means.

4.3 Proof of concept – the ediT2 editor

In order to test whether the model was implementable and to conduct usability studies (see Chapter 5), we reused the notions proposed in (Kobbe et al., 2007) to the specification of CSCL scripts (Section 2.3.2) to design and implement a Learning Scenario Editor (LSE), named ediT2, corresponding to the II.b possible usages of the T^2 model (Section 4.2). In Chapters 6, 7 and 9 the extension of such editor will be discussed in order to implement the ideas corresponding to the item II.c (the architecture and technologies employed in the development of this study are described in Appendix A).

ediT2 has a general menu allowing the following operations:

- "new" to create a new script/template;
- "open" to open a script/template previously created;
- "save" to save the script/template under design. The file saved (.zip) contains two elements: the script/template file itself (.t2ml, an XML-based file created in this work to support the T^2 model implementation and to facilitate its interoperability with other platforms; see Appendix B, for details); and a trace file (.csv) that registers all table-tree manipulations during the script/template conception/adaptation, for future analyses (see Appendix C, for details);
- "view" to visualize the tree corresponding to the script/template (table) under manipulation;
- "print" to print the script/template under design as a .pdf file.

Figure 4.3 presents the general interface of ediT2, which is composed of two zones. The first zone, on the left side, is an *ad hoc* feature that allows creating the items (activities, groups, participants, resources and roles) that will be referred to in the script. The second zone, on the right side, is the table interface. On the top of each

column are the notions that have currently been selected (in the example presented in Figure 4.3: Activity, Group, Participant and Resource (the Role notion not being used)).

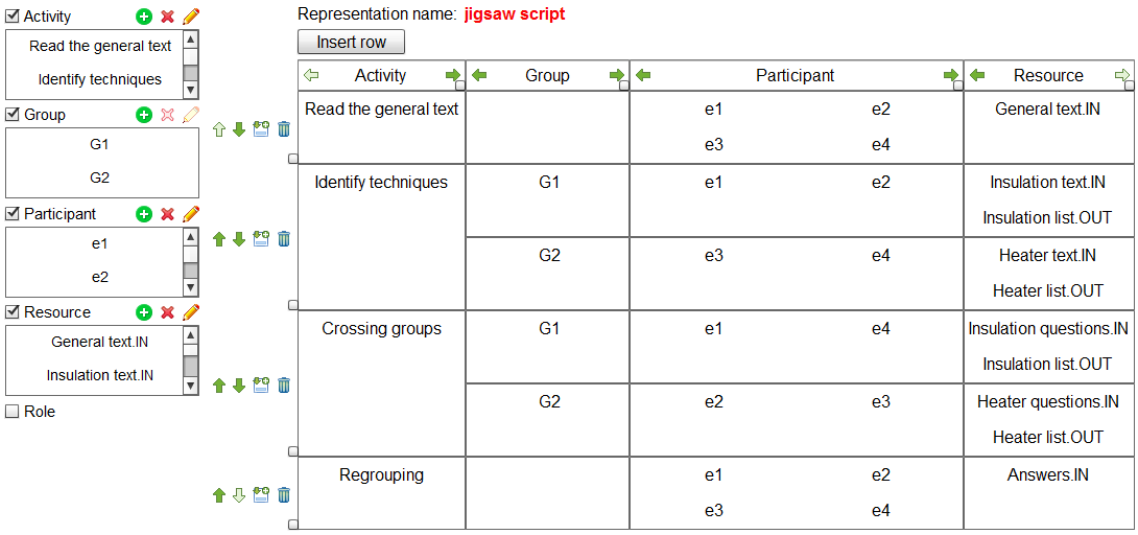


Figure 4.3 The ediT2 general interface (jigsaw script (Figure 2.2) from the first usability test, Chapter 5)

In this figure, we have re-written the representation created by one of the teachers who participated in the tests of the system (see Chapter 5, for details) for the jigsaw script presented in Figure 2.2, and translated the items (names of activities, etc.) from French to English. One may notice that for the “Regrouping” activity (last row) the teacher tagged the “Answers” resource as “.IN” when it is likely to be, rather, an “.OUT” resource, to represent, in fact, an artefact to be produced (and not to be received) by the students.

Deciding to use a component notion (i.e., creating a column) requires ticking the corresponding box in the left side of the interface. This creates the corresponding column in the table (e.g., the “Activity” column). De-selecting such a box removes the respective column in the table. For each selected notion, three buttons allow creating the items that will be referred to in the script (e.g., the activity “Read the general text”), removing an item or editing it. In an effective system, part of these items may pre-exist (typically, students and resources lists) and be imported.

Script-components are created by clicking on the “Insert row” button. Script-components are edited (i.e., associating items to cells) by dragging-and-dropping an item from a left box into a cell corresponding to a same box notion. Sub-rows are managed by right-clicking on the corresponding cell(s) and selecting the “split” or “merge” choice from the contextual menu.

Different facilities are offered. As a first example, the editor offers to duplicate a row (with or without its items). In many scripts we found script-components whose structures were identical or highly similar (see for instance the second and third rows in Figure 4.3). Duplicating a row (or adding a new branch identical to its left-brother branch, within a tree perspective) is very convenient, in particular when its internal structure is complex (sub-rows, merged cells, etc.). As another example, items can be copied and pasted from one cell to another.

Every column is associated with a left and right arrow that moves it (the column is moved as an entirety). Similarly, every script-component is associated with two arrows that move it up or down (and with a bin icon, which removes it from the table). Finally, columns' and script-components' dimensions can be modified (reduced and expanded; e.g., the "Participant" column is expanded in Figure 4.3).

All manipulations offered to the user are first interpreted in terms of the model. While the user creates his/her table, the system (1) builds the corresponding abstract tree and uses it as a reference to accept or reject users' actions, and then if applicable, (2) modifies the table, which includes the propagation actions if any.

Figure 4.4 presents a translated representation produced by another teacher who also participated of the ediT2 tests (Chapter 5), in this case, for the reciprocal-teaching script (Figure 2.1).

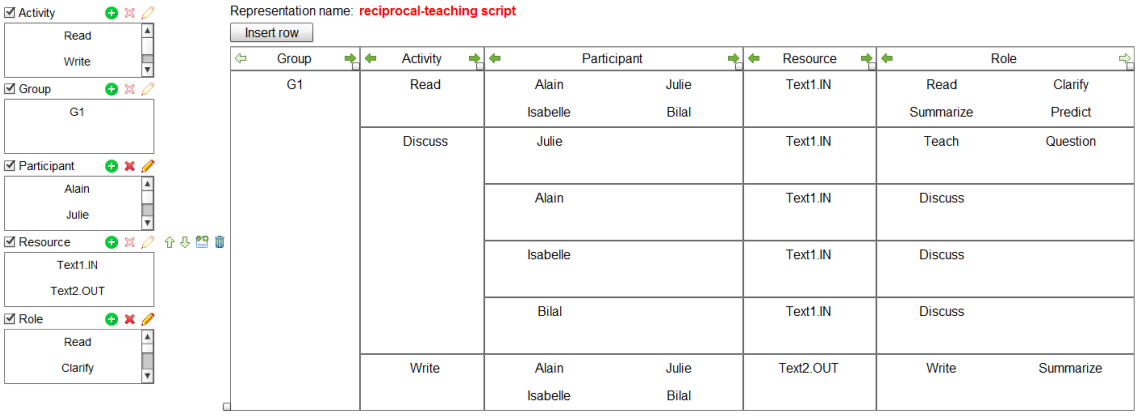


Figure 4.4 Using ediT2 to represent the reciprocal-teaching script (from the first usability test, Chapter 5)

The first steps for a straightforward process to obtain the same representation displayed in Figure 4.4 could be as follows (other solutions are possible, since some representations can be obtained via different series of actions):

1. Select the components to be used (left part of the interface, select "Activity", "Group", etc.). This creates the corresponding (empty) columns in the table.

2. Use the right and left arrows to put the components in the requested order (header line of the table). In the example, the adopted order is Group, Activity, Participant, Resource, Role.
3. If not imported from elsewhere, create the necessary items (left part of the interface). In the example, three activities have been defined (“Read”, “Write” and “Discuss”), one group (“G1”) and four participants (“Alain”, “Julie”, “Isabelle” and “Bilal”). Items can be created at any time and not necessarily at first as done here.
4. Create a first row by hitting the “Insert row” button. A first empty row corresponding to the five columns (Group, Activity, Participant, Resource, Role) is created.
5. Drag-and-drop “G1” in the first row / first column cell (Group column).
6. Right click in the first row / second column (activity cell) to split it, and create three sub-rows.
7. Fill the remaining cells of the first row by dragging-and-dropping the items (first sub-row: activity “Read”; participants “Alain”, “Julie”, “Isabelle” and “Bilal”; resource “Texte1.IN”; roles “Read”, “Clarify”, “Summarize” and “Predict”).
8. Manage the second sub-row (activity “Discuss”) by dragging-and-dropping “Discuss” in the Activity cell, right-clicking on the Participant column, splitting it into four cells, and filling each of these new sub-rows (first sub-row: participant “Julie”; resource “Texte1.IN”; roles “Teach” and “Question”, etc.)

4.4 Semantics

Natively, column headings are just type labels. An *activity* column states that the values that can be put into this column are edited as of the type *activity* (i.e., in the ediT2 interface, are defined as activities in the activity box in the left side of the interface). The notions’ semantics is only carried out by the label and, thus, related to the user’s perspective. How an activity relates to groups or roles is natively represented in the system by a generic *is-associated-with* relationship denoted by the row structure (we will refer to this as a *structural semantics*).

Table 4.1 A basic interpretation of the *is-associated-with* generic relationship

	Activity	Group	Participant	Role	Resource
Activity	N/A	The activity is achieved by group(s)	The activity is achieved by participant(s)	The activity is achieved by playing role(s)	The activity is achieved by considering/producing resource(s)
Group	The group is to consider activity(ies)	N/A	The group is composed of participant(s)	The group is to play role(s)	The group is presented with / is to produce resource(s)
Participant	The participant is to consider activity(ies)	The participant is associated in group(s)	N/A	The participant is to play role(s)	The participant is presented with / is to produce resource(s)
Role	The role involves considering activity(ies)	The role is played by group(s)	The role is played by participant(s)	N/A	The role is to be played using / producing resource(s)
Resource	The resource is to be considered / produced in activity(ies)	The resource is presented to / produced by group(s)	The resource is presented to / produced by participant(s)	The resource is associated with role(s)	N/A

Table 4.2 Basic actions to adapt a script (examples from the script presented in Figure 4.4)

	Adaptation of the script	Example	Nature	Table perspective	Tree perspective	Actions to be processed
Script flexibility	Add/Remove an item within the script	Associate a new role to Isabelle within the activity "Discuss"	Change the definition of a script-component	Add/Remove an item in/ from a cell	Add/Remove an item in/from a node	Create the item. Drag-and-drop it into the cell (last column, 4th row)
	Re-allocate an item within the script	Associate the roles "Teach" and "Question" to Bilal rather than to Julie	Change the definition of a script-component	Drag-and-drop an item from one cell to another	Displace an item from one node to another	Cut and paste the roles from Julie's row to the cell in Bilal's row
	Add/Remove/Duplicate a script-component	Remove the G1 script-component	Change the script-component list	Add/Remove/Duplicate a row (and its sub-rows, if any)	Add/Remove/Duplicate a branch (and its sub-branches, if any)	Click on the bin icon corresponding to row G1
	Change the order of script-components	Create a step to be addressed before the one presently associated to G1	Change the script-component list	Create a row (or sub-row) and move it up/ down	Create a branch and move it left/right	Insert a row. Move it up using the up arrow. Define its structure and content

	Introduce a new level of break-down into (part of) a script-component	In the "Read" activity, associate different resources to Alain and Julie on one side, and Isabelle and Bilal on the other	Change the internal structure of a script-component	Split a cell into several cells	Split a leaf/node in several leaves/nodes	Right click on the first row participant cell and split it into two sub-rows. Distribute participants and resources as needed
	Regroup different levels of break-down of (part of) a script-component	Regroup the "Discuss" and "Write" activities	Change the internal structure of a script-component	Merge cells into a single cell	Merge leaves/nodes	Select the two "Discuss" and "Write" activities. Right click and select "merge" (to be followed by actions to reorganize the participants if necessary)
Representation Flexibility	Use a new (or not using anymore a) component notion to describe the script	Withdraw the group notion	Change the script-structure	Add/Remove a column	Add/Remove a level	Unselect the "Group" component in the left side of the interface (the column is automatically removed)
	Change the order in which component notions are used	Highlight the relationship between participants and roles	Change the script-structure	Moving a column left/ right	Moving a level up/ down	Displace the role column towards the left

Table 4.1 presents a basic interpretation of the generic is-associated-with relationship. As an example, within this interpretation, the "Identify techniques" script-component represented in Figure 4.3 may be read as follows: Activity "Identify techniques" *is achieved by groups* G1 and G2; G1 *is composed of* e1 and e2; each of these participants *is presented with* resource "Insulation text" and *is to produce resource* "Insulation list". G2 *is composed of* e3 and e4; each of these participants *is presented with* resource "Heater text" and *is to produce* resource "Heater list".

This structural approach is significantly different from representing relationships and associated constraints semantics by an explicit meta-model (see for instance (Miao et al., 2005) meta-model of CSCL scripts). If taking a meta-modeling perspective, the implicit meta-model underlying the T^2 model is basic: a list of notions N_i, N_i related to N_{i+1} by a 0..* relationship and, in our case, $i \in [1..4]$ and N_i taking values in {"Activity, "Group", "Participant", "Resource", "Role"}. Such a perspective, however, does not capture much of the work's rationale. Another way to phrase it, more in line with the approach, is to say that a table is an easy-to-use way to offer end-users with (limited) meta-modeling means, in this case selecting the notions they want to use and ordering them as they prefer.

4.5 Offered flexibility

As mentioned previously, we consider that editing and setting-up a script is not a straightforward process. Rather, teachers engage in an elaboration process, i.e., iteratively refining the script under design until a satisfactory structure has been obtained. Teachers may come to change their modeling perspective during this process, i.e., while skipping from a broad idea to a precise description. Finally if, at run-time, the script requires some important changes, teachers must here again reflect on and re-design the script (Dillenbourg et al., 2013), which may lead them to here again adapt the way they consider the script. Flexibility is thus an important concern.

With respect to flexibility, different types of adapting actions can be discerned. The first aspect is related to the script-components and, basically, the possibility to easily manage the table structure, e.g., add rows or displace items. The second aspect, which we have proposed to call representation flexibility, is related to the script-structure.

Teachers, as individuals, develop a personal professional experience. Using a script is just one episode that they harmonize with their practices, not the other way around. Languages and editors introducing notions provide teachers some support. Yet, by imposing a particular way of conceptualizing and representing scripts, they also may poorly correspond to individual teachers' perspectives or needs, and may pose an appropriation problem.

Considering the fact that a script is a complex artifact to be elaborated and adapted, and that teachers may have different perspectives corresponding to their

modeling perspectives, their interpretations of the scripts and the way they prepare and manage their sessions, introduces another aspect of flexibility, that we call *representation flexibility* (Sobreira and Tchounikine, 2012): teachers should be allowed to adapt the way the provided representation scheme can be used to represent the script.

If one considers that the used representation scheme is a conceptualization tool, this corresponds to some extent to adapting the representation scheme to the teachers' conceptual perspective on the script. Just as CSCL scripts must guide and support students whilst not over-constraining their activity (Dillenbourg, 2002), teachers should be provided representation means that, while elaborating the script, guide and support them whilst not over-constraining them, i.e., imposing complex notions or syntactical constructions that make it difficult or painful (e.g., time-consuming) to change design decisions.

One of tables' interests is to natively offer a basic representation flexibility feature by ordering or re-ordering the columns. This allows to change the way the notions provided to represent a script are used, e.g., moving from a conceptualization of the script as "a set of activities to be realized by participants" to a conceptualization as "a set of participants playing roles associated to resources".

Such different script-structures correspond to different ways of conceptualizing a setting, but do not necessarily lead to semantic differences. Operationalized on an enactment framework (for instance Moodle, see Section 7.1), different CSCL scripts representations (e.g., Activity-Group-Resource or Participant-Resource-Activity) may lead the same deployment (courses associated with students and resources). However, allowing teachers to adapt the representation structure (as long as representations are equivalent) allows them to use the most convenient one according to the setting, their practices and/or perspectives.

Table 4.2 takes as example the script presented in Figure 4.4 to summarize how these different actions allow for the targeted flexibility, i.e., adapting the script (script flexibility) and the script-structure (representation flexibility).

4.6 Constraints

The model introduces three native structural constraints, presented in the next subsections.

4.6.1 The allocation of items into table cells

An ediT2 natively-implemented basic constraint is that an item defined as a notion *N* (being this item originated from the *N* editing box of the editor, or some table cell of its corresponding *N* column, e.g., *N* = "participant") can only be dropped into the corresponding column of the table.

4.6.2 The script-structure

Given a script-structure, the table perspective causes all script-components to have the same representation: a script is a homogeneous construction. The fact that (for a given script) all script-components must be conceptualized in the same way is an important design decision, and the rationale and arguments supporting this decision are:

1. The representation's homogeneity allows for managing the basic coherence of scripts in structural terms, which is easy for both humans and machines, unlike the defining of syntactical rules that would allow for accepting different forms at the cost of complexity. Instead of providing end-users syntactical constructions with constraints that must be understood and properly used, the table structurally denotes the constraint.
2. From a practical perspective, most of the disadvantages of the homogeneity imposed by a table representation can be easily overcome. For instance, the table-interface can be partially filled, i.e., if a notion is not useful for a given script-component, the corresponding cell can be left blank; this remains coherent and easily understandable.

As highlighted in Chapter 2, we believe that simplicity is a core argument. When Computer Scientists often tend to allow all possibilities at the cost of complexity, we believe that this is not necessarily very positive for the effective use of software in education. To our opinion, this homogeneity constraint should be seen and addressed as an issue if and only if it appears to be an effective practical issue, and an obstacle to the fact the model is perceived as useful but too constraining.

4.6.3 The 1-n relationship

In order to avoid ambiguous constructions, the T^2 model introduces as a constraint the fact that a script-component must follow a 1-n with $n \geq 1$ relationship (or, in other words: a script is a tree perspective and not a graph, i.e., a cell has only one parent).

If all script-components respect a 1-1 relationship (i.e., a table composed of full-rows, with no split cells), the order of notions denotes the way the script is conceptually addressed (e.g., as a set of activities or as a set of roles), but the fact that columns are displaced does not further change the script semantics as the 1-1 relationships remain identical. In direct contrast, if the script presents 1-n with $n > 1$ relationships, a change of perspective resulting in a displacement of columns may deeply affect the tree structure.

In order to explain the implications of this constraint without going into formal constructions, we will use the example introduced in Figure 4.1. There, the first and fourth script-components follow a 1-1 relationship, whereas the second and

third ones a 1-2 relationship. Let us consider that one wants to change the structure of the “Identify techniques” script-component and state that it will be achieved by a single group. Visually, it means that the cells containing G1 and G2 are merged into a single cell. Modified in this way, the script remains coherent with a 1-n relationship. The script now specifies that “Identify techniques” will be achieved by a single group from which P1 and P2 will use “Insulation text” to produce “Insulation list”, and P3 and P4 will use “Heater text” to produce “Heater list”.

Let us now consider in the same figure that one wants to change the structure of the “Crossing groups” script-component and regroup its participants (Table 4.3). Visually, it means that the cells containing (P1, P4) and (P2, P3) are merged. Modified in this way, the script becomes incoherent with the 1-n relationship: this cell, if created, would have two different antecedents (the cells containing G3 and G4).

Table 4.3 Rationale for the 1-n with n ≥ 1 relationship

interpretable construction

Activity	Group	Participant	Resource
Crossing groups	G3	P1,P4	Insulation questions(in), Insulation list(out)
	G4	P2,P3	Heater questions(in), Heater list(out)

ambiguous construction (it is no longer a tree)

Crossing groups	G3	P1,P4,P2,P3	Insulation questions(in), Insulation list(out)
	G4		Heater questions(in), Heater list(out)

In the sense of the tree-model, a node would have two fathers, and is therefore not a tree anymore. Such a structure is discarded by the model and made structurally impossible. The reason is that, if accepted, the script’s semantics become ambiguous (the Group-Participant relationship of this script-component becomes unclear: there are two groups but one set of participants).

This constraint allows keeping the table equivalent to a tree and thus interpretable within the adopted structural semantics. Some syntactical sugar could be though to let users specify specific intentions. An interface could thus be offered to not impose this constraint but, rather, allows cells to have different antecedents as long as the construction would be interpretable (keeping the machine representation as a tree to benefit from tree manipulation algorithms). However, as it is of little interest and would go against the simplicity principle, we have opted to keep the representation simple and orthogonal.

As is obvious when considering the table structure, the “Crossing groups” participants can easily be grouped by merging G3 and G4 in a single cell and then merging the participants. Now, “Crossing groups” is associated with a group containing all students, who are presented with two different input resources (“Insulation questions” and “Heater questions”) to produce two different output resources (“Insulation list” and “Heater list”).

It may be noticed that if G3 and G4 are merged but the participants (P1 P4) and (P2 P3) are not merged, the script has different semantics (referring here to the basic semantics as defined in Table 4.1). There is now one group containing all

students within which pairs of participants are provided with different resources, and are expected to produce different outputs. Another adaptation (coherent with the 1-n structure), carrying again different semantics, would be to associate a specific resource to each of the participants by creating one sub-row per participant (i.e., creating a 1-4 relationship).

As a different example, let us consider that one comes back to the initial structure as shown in Figure 4.1 and that one wants to change the structure of the "Identify techniques" script-component by regrouping its Resources. Visually, it means that the cells containing "Insulation text", "Insulation list", "Heater text" and "Heater list" are merged. Modified in this way, the script becomes incoherent with the 1-n relationship. However, such a change may easily be represented by keeping the current table structure (the two cells are kept) and duplicating the four items (the four resources) in each one of the two cells.

Another interesting feature is that some of the modifications required to adapt a script while keeping coherent with the model can be automatically derived from the model and thus automatically achieved by propagating actions. For instance, let us consider that one wants to change the structure of the "Identify techniques" activity by splitting the G1 group into two different sub-groups G1-1 and G1-2. In this case, the script becomes incoherent (the {P1, P2} participants cell has two fathers). However, coherence can easily be maintained by propagating the implications of the change, in this case duplicating the associated participants' and resources' cells.

As another example, if the Group and Activity columns are swapped, it is necessary to adapt the table (if not, "Identify techniques" and "Crossing groups" would have several parents, G1/G2 and G3/G4, respectively). In this case, two new activity cells must be created (with duplication of values) to represent the G1-"Identify techniques" and G2-"Identify techniques" relationships and two others must be created to represent the G3-"Crossing groups" and G4-"Crossing groups" relationships. This will lead to a script based on the Group-Activity-Participant-Resource script-structure, and composed of six script-components (all with a 1-1 relationship). In general, such propagation actions are to be processed recursively through the tree's branches.

As in the column displacement operation, splitting cell may also require the propagation of modifications to maintain the representation coherent with the model. Table 4.4 presents more two examples of propagation action, where the first one is yet about displacement of columns and the second one, about split cell operation. It may be noticed that, when propagating changes in the tree structure, the sub-rows and items may be managed in different ways. For instance, when splitting the {P1, P2, P3, P4} participant cell into two (Table 4.4, row 2), the resulting participant cells may be reinitialized to blank, associated to the participant(s) preexisting to the manipulation or spread over the two new cells. This may be configured and, anyway, the result may be easily adapted by further merge/split or drag-and-drop actions.

Taking the "Discuss" activity (Figure 4.4) as a last example, three of its students (Alain, Isabelle and Bilal) have the same "Discuss" role, and in order to highlight this, one could want to merge these three cells, what would be impossible

by the 1-n relationship. However, different strategies could overcome this issue. One is to displace the role column and adopt a Group-Activity-Role-Participant-Resource script-structure. Indeed, if one analyzes the representation produced by the teacher, the Role notion seems more structuring than the Resource or Participant ones. Once this move is achieved, the three "Discuss" cells can be merged.

Table 4.4 Examples of manipulations of script representations and propagations

<table>
<tr><th>Script representation</th><th>Manipulation</th><th>Script representation modified (when applied)</th></tr>
<tr>
<td>
<table>
<tr><th>Act</th><th>Part</th><th>Role</th></tr>
<tr><td rowspan="3">A1</td><td>P1</td><td>RL1</td></tr>
<tr><td rowspan="2">P2</td><td>RL2</td></tr>
<tr><td>RL3</td></tr>
</table>
</td>
<td>Displacement of the Role notion towards the left (or of the Participant notion towards the right)</td>
<td>
<table>
<tr><th>Act</th><th>Role</th><th>Part</th></tr>
<tr><td rowspan="3">A1</td><td>RL1</td><td>P1</td></tr>
<tr><td>RL2</td><td>P2</td></tr>
<tr><td>RL3</td><td>P2</td></tr>
</table>
</td>
</tr>
<tr>
<td>
<table>
<tr><th>Act</th><th>Part</th><th>Role</th><th>Res</th></tr>
<tr><td rowspan="6">A1</td><td>P1</td><td>RL1</td><td>R1</td></tr>
<tr><td rowspan="5">P1 P2
P3 P4</td><td rowspan="5">RL1
RL2</td><td>R2</td></tr>
<tr><td>R3</td></tr>
<tr><td>R4</td></tr>
<tr><td>R5</td></tr>
</table>
</td>
<td>Split the cell containing P1, P2, P3 and P4 Participants in two</td>
<td>
<table>
<tr><th>Act</th><th>Part</th><th>Role</th><th>Res</th></tr>
<tr><td rowspan="6">A1</td><td>P1</td><td>RL1</td><td>R1</td></tr>
<tr><td rowspan="2">P1 P2
P3 P4</td><td rowspan="2">RL1
RL2</td><td>R2</td></tr>
<tr><td>R3</td></tr>
<tr><td rowspan="2"></td><td rowspan="2"></td><td>R4</td></tr>
<tr><td>R5</td></tr>
</table>
</td>
</tr>
</table>

Another option is to merge the three Participant cells, and then the three Resource cells ("Text1.IN" in all cases), and finally the three Role cells. Given the overall representation, however, this second option might convey a different perspective for the teacher (in the "Read" row he groups the participants and the roles, thus one may hypothesize that splitting each student's role in the "Discuss" activity is an explicit decision). Finally, as the teacher did, one may simply duplicate the value in the cells.

As one can see in these examples, the model implements semantics that are to a large extent structurally represented by the table interface. Using the table-interface natively produces coherent scripts. In some cases, the 1-n relationship imposes duplications but, from the point of view of the editor interface, these are managed by copy-and-paste and drag-and-drop manipulations and do not seem to significantly disturb users. Moreover, some of the manipulations requested to maintain coherence are automated by propagation actions, which here again removes some burden.

4.7 Adding control rules

The basic structural semantics may be enhanced by adding an analysis of the users' actions with respect to other constraints. Technically, this may be implemented through constraints as control rules, and firing these rules when end-

users act on the table. Control rules may be used to impose a specific semantic and/or provide modeling support.

For instance, we found many different script-structures in our usability study (Chapter 5). As an example, we found some teachers denoting what work participants had to achieve with the Activity notion, some others with the Role notion, and some others using both (one complementing the other). In some contexts, these differences in perspectives and usages could be an issue, for example, if aiming for representations to be indexed into a repository (to be afterwards reused for others) or, on another dimension, correspondence with the format of a given enactment framework. In such a case, the model can be enhanced by control rules imposing a particular pivotal notion or a partial ordering of notions or constraints related to relationships between items (e.g., constraints related to the branching factors or to some notions' cardinality). For instance, it may be imposed that Activity is the pivotal notion; that different activities in the table must be named differently (one cannot drop the same activity item into different Activity-cells), and; that an Activity-cell contains only one item (in other words: a line depicts an Activity and thus Activities should be different and single).

Other reasons to use additional constraints may be to provide teachers modeling support (linking constraints to some methodological instructions to avoid, for example, odd constructions) – this topic will be emphasized in Chapter 6.

4.8 T^2 tree manipulation algorithms

A basic advantage of a tree representation is that manipulation algorithms are simple for most of them and generic. As example, we present here after two (simplified) algorithms involved in T^2 tree manipulations: "split a cell" (Figure 4.5) and "move left/right a column" (Figure 4.6). In such algorithms, "cell" and "column" correspond to table interface elements, and, correspondingly, "node" and "level" to tree model components.

Algorithm: Split a node N in level l_i
(to be applied recursively over the tree structure containing the N root)

1. Identify the branching factor between N and its descendant(s) in level l_{i+1}
2. Apply a strategy to identify the number s of nodes that N will be split in
{Different strategies are possible. One option is to allow the user to split N in any number of nodes, but this may give rise to incoherent constructions that must be corrected afterwards. The implemented option is to restrict possibilities according to the set of divisors in level l_{i+1}}
3. Create s – 1 new nodes
4. Apply a strategy to fill the nodes in level l_i
{As examples of strategies: leave N with its original content and initialize new nodes to blank; initialize all nodes to blank; replicate the N content to all s – 1 new nodes}
5. Update the connections from level l_{i-1} to level l_i and from level l_i to level l_{i+1}
{Connections must be updated consistently with the strategy adopted when identifying s}

Figure 4.5 "Split a cell" algorithm

Algorithm: Move up/down level l_i

(considering "Move right" operation; the "Move left" algorithm is equivalent to that one)

1. Move down level l_i, corresponding to the column under manipulation
2. Correspondingly, move up level l_{i+1}, corresponding to the right column taking into account the column under manipulation

{If nodes in l_i and l_{i+1} maintain a 1-n father-children relationship (n > 1), apply under the nodes in level l_i the algorithm presented in Figure 4.5}

3. Update the relationship between levels l_i and l_{i+1} and, when applied, between levels l_{i-1} and l_{i+1}, and l_i and l_{i+2}

Figure 4.6 "Move left/right a column" algorithm

5 Validation of ease of table-tree representation, perceived flexibility and pedagogical expressiveness

5.1 Initial considerations

5.2 Is the model/system easy to use?

5.2.1 First usability test
5.2.2 Second usability test
5.2.3 Analysis of the technical competences required to use the editor with respect to teachers' ICT skills
5.2.4 Discussion

5.3 Do teachers avail themselves of the flexibility provided?

5.4 Does a table notation allow representation of a wide range of scenarios?

This chapter builds on and extends some ideas and concepts presented in our paper published in the Computers and Education journal (2014).

Dans ce chapitre nous présentons les résultats obtenus dans quelques expérimentations et études exploratoires menées pour analyser comment les enseignants utilisent ediT2. Dans cette perspective, nous avons examiné les questionnements suivants : « le modèle/système est-il facile à utiliser ? » ; « les enseignants profitent-ils de la flexibilité fournie par la plateforme ? » ; et « la notation en table permet-elle la représentation d'une large gamme de scénarios ? ».

Afin de répondre au premier de ces questionnements, nous avons mené deux expérimentations d'utilisabilité. Dans la première, ont participé cinq enseignants français d'écoles primaires et secondaires, un spécialiste de modélisation de scénarios pédagogiques et un spécialiste de modélisation de données. Dans la seconde expérimentation ont participé dix-huit professeurs universitaires de différents domaines de l'Université de Valladolid, Espagne. Elle a permis de comparer l'utilisation de deux outils dont l'ediT2.

Afin de répondre au deuxième questionnement, nous avons considéré les données de la première expérimentation d'utilisabilité pour étudier si les enseignants utilisent la flexibilité de l'éditeur, à partir de quatre indicateurs : les squelettes des représentations produites (les arbres produits) ; la structure et le nombre des éléments des différents arbres modélisés ; les actions effectuées par les enseignants pour définir/modifier la structure de chacun des scénarios représentés ; et le nombre d'actions effectuées par les utilisateurs qui ont participé à l'expérimentation.

Par rapport au troisième questionnement, nous avons apporté différents éléments de réponse. Tout d'abord, nous avons examiné les deux expérimentations d'utilisabilité et, dans les deux, aucun des enseignants n'a rapporté de soucis avec l'expressivité pédagogique de l'éditeur. Un autre ensemble d'actions d'évaluation a été exploratoire. Tout d'abord, nous avons récupéré 25 scénarios collaboratifs de la littérature, qui ont tous pu être représentés par des tables. Ensuite, nous avons examiné certains scénarios élaborés par la communauté LAMS et, là encore, ils ont pu être représentés par des tables. Enfin, une dernière action d'évaluation a été menée avec la version générique d'ediT2 (qui sera présentée dans le chapitre 8) pour représenter des scénarios collaboratifs modélisés par un autre éditeur de scénarios d'apprentissage (SceDer), qui a été évalué comme en étant capable de modéliser des scénarios pédagogiques de référence. Ces actions ont montré que certains scénarios nécessitant des planifications complexes ou des principes de dynamisme devraient être représentés à partir d'une version améliorée de l'éditeur. Nous concluons que la table présente une certaine expressivité, mais qu'elle ne doit pas être considérée comme un remplacement de moyens plus complets.

5.1 Initial considerations

In this chapter, we present the results obtained from some experiments and exploratory studies conducted to analyze how teachers used ediT2. We consider the questions “is the model/system easy to use?” (Section 5.2), “do teachers avail themselves from the flexibility provided?” (Section 5.3), and “does a table notation allow representation of a wide range of scenarios?” (Section 5.4). Complements will be presented in Chapter 8.

5.2 Is the model/system easy to use?

In order to support our claim that the model/system is easy to use, we propose two elements: the results of usability tests and an analysis of the technical skills required to use such an editor with respect to teachers’ ICT skills (Subsections 5.2.1-2 and 5.2.3, respectively). In Subsection 5.2.4 we discuss these elements.

5.2.1 First usability test

This experiment involved five primary and secondary French school teachers (a sample of 5 is usually considered as sufficient for such an editor usability test). Also, we involved one learning-scenario modeling specialist (an instructional designer) and one modeling specialist (a Computer Science university professor) as a way to get some possibly different input.

The experiment and analyses were conducted to identify teachers’ representations of two CSCL scripts adapted from the literature (the ones presented in Figure 2.1 and Figure 2.2). We introduced two scenarios to limit the intrinsic bias that consists in introducing a computer-based tool and, within the same session, analyze how users use it (although the editor builds on its resemblance with spreadsheets or table editors, any tool requires some time to be adopted and adapted to one's needs).

The task of the teachers corresponded to the considered prototypical use-case: given a script represented in a narrative (and more or less abstract) way, engage in an edition activity and adapt it in a way that corresponded to how they would implement it in their classrooms (instead of targeting a “best solution” that could be compared with that of “experts” in view of quality analysis). Here, they were asked to stop the process when happy with their representation, highlighting that there were no “good” or “bad” answers, and that the amount of time spent representing the script was not important.

We did not consider if the different representations produced by the different teachers were or were not semantically equivalent to each other (or to the canonical patterns). The study was conducted to analyze trends reflecting the editor's (and the underlying model's) usability and, more precisely, whether teachers succeeded in using the editor to model the script the way they wanted (the editor was introduced free of any methodological training).

The protocol of this experiment is summarized in Table 5.1.

Table 5.1 The French usability test protocol

Phase	Content
1	The teacher was presented with a demonstration of the ediT2.
2	The teacher was presented with the narrative of a first script (the reciprocal-teaching script as presented in Figure 2.1) and asked to create a representation with the editor. The teachers were prompted as follows: "You plan to implement the following script in your classroom. Use the editor provided to create the synthetic representation you find most suited to plan the different steps and, while the script unfolds, annotate the plan, if necessary, to monitor what is happening or adapt the script".
3	A first questionnaire and debriefing were conducted to collect the teacher's first impressions, and respond to any questions related to the editor. It determined whether users were at ease with the editor features and opened up a first general discussion. Sample question: "What problems did you experience, if any?"
4	The teacher was presented with the narrative of a second script (the jigsaw script as presented in Figure 2.2) and asked to create a representation with the editor (prompted as in Phase 2).
5	The teacher was presented with two events related to his/her jigsaw script representation and asked to explain how she/he would react and adapt the script representation. The first event involves one student stating that she/he does not want to work with his/her assigned partner in one of the pairs defined by the teacher. The second event involves one student finishing long before the others, while working in parallel.
6	The teacher was presented with a final questionnaire presenting four parts. First, questions on the teachers' perspective on scripts in general, and on the notions provided by the editor to represent scripts (sample question: "I have difficulties thinking with the notions provided"). Second, questions on the editor. The objective of these two first questions was to make sure teachers dissociated the editor usability (which is what we were interested in) from their personal perspectives or potential difficulties with the notion of script, the two scripts used as case studies and/or the notions available. Third, questions on the way teachers engaged in the process and used the system (sample questions: "I found that the capacity to adapt the representation [...] allowed me to reflect on the script, to refine my vision"; "When I represented the first script, I got right into it and adapted things little by little"). Finally, teachers were asked to highlight any comments or suggestions (open discussion).

The answers to the questions on the editor (see Table 5.2) confirm that the editor and underlying model are intuitive and easy to use: all five teachers agree or strongly agree they found the tool easy to use for the second script (and four of them agree or strongly agree since the first script).

We may notice that the instructional designer found the tool easy to use for the first script and less for the second (neither agrees nor disagrees). The debriefing discussion revealed that this answer indicated difficulties with the model's

completeness (she wanted to represent the schedule in a more explicit way than the basic frame allows). The only other severe criticism related to the editor's expressiveness came from the modeling specialist (a Computer Scientist) who, unsurprisingly, raised the issue that such a modeling tool did not support the modeler by imposing precise rules. She was not at ease with the fact that she could represent the scripts in different ways or, in other words, with the editor design rationale.

Table 5.2 Questions related to the editor and number of answers
Ti correspond to teachers, ID to Instructional Designer and MS to Modeling Specialist

Questionnaire results	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
I think the tool looks like a spreadsheet or an array editor as can be found in classic office suites.			T3	T4 T5 MS ID	T1 T2
Once the demo was over, I had the feeling I had understood how to use the tool.				T1 T4 MS	T2 T3 T5 ID
Looking back, I had correctly understood how to use the tool after the demo.			T4	T1 T3 MS	T2 T5 ID
When I represented the first scenario I found the tool easy to use.		T3		T1 T4 ID MS	T2 T5
When I represented the second scenario I found the tool easy to use.			ID	T1 MS	T2 T3 T4 T5
I have the feeling that if I had to use the tool a third time it would be easy.				T1 ID MS	T2 T3 T4 T5

In direct contrast, the teachers' criticisms were suggestions of extensions such as an additional column to mention activity length or how the script unfolded for adaptation in future sessions, i.e., suggestions related to the way they would effectively use such a tool on a personal level (this functionality was added after the experiment (see Section 7.3)).

Although no general conclusion is to be drawn given the limited number of participants, the difference between the input originating from basic teachers and from modeling specialists may be relevant and confirm the interest of considering simplicity and flexibility rather than comprehensiveness when targeting such a public.

5.2.2 Second usability test

A second usability test was organized in collaboration with the GSIC group of the University of Valladolid (Spain). The experiment aimed at comparing the use of different tools from which ediT2 (see Prieto et al., 2014, for details).

Questionnaires included questions related to usability, which is the part of the experiment we report here.

Eighteen university lecturers from different disciplines at the University of Valladolid were asked to represent CSCL scripts in a 12 hours blended experiment split in two classroom sections (4 hours each) intercalated by an individual in-distance activity (4 hours). These activities were introduced and finalized by pre- and post-workshop sections, respectively, as described in Table 5.3.

Table 5.3 The Spanish usability test protocol

Phase	Content
1	Pre-workshop section. Teachers were presented to an example of CSCL script and conducted to answer an initial profiling questionnaire.
2	First classroom section. Teachers were introduced to general collaborative learning approaches and the CSCL tools used in the experiment (ediT2 and WebCollage). In dyads, teachers modeled a hypothetical CSCL script using a learning authoring tool (50% using ediT2, 50% using WebCollage). Then, the ones who modeled such script using ediT2 were to represent it using WebCollage, and vice-versa). After, teachers individually answered some questions concerning tools features and usage.
3	Online section. Teachers individually, at their homes, modeled a CSCL script, that could exist in their realities in classrooms, with one of the two editors manipulated in the last phase (tool freely chosen by each teacher).
4	Second classroom section. After receiving the feedback concerning the last phase, teachers individually revised the questionnaires produced in Phase 2. In dyads, they discussed about the editors to finally agree, in groups composed by 2-3 previous dyads, about the possible issues to the adoption of such authoring tools. Then, they were introduced to the operationalization architecture used in the experiment and after, individually deployed their respective scripts produced in the last phase, into Moodle. Finally, they were conducted to answer an evaluation questionnaire, concerning professional questions.
5	Post-workshop section. Three weeks later, teachers were conduct to answer a final questionnaire, reconsidering the subjects discussed in Phase 2 (tools features and usage), and taking into account the questioning about the possibility of adoption of such tools in future situations.

In Phase 2, teachers were asked to the questions: “[Do you think that] the editor is easy to use?”, using a Likert scale 1-6 (6=totally agree). The average response was 4.9 (std. dev.=0.6, min. value=3) – the questionnaire in Phase 5 gave similar results.

5.2.3 Analysis of the technical competences required to use the editor with respect to teachers’ ICT skills

Another more general way to investigate the usability question is to consider the technical competences required to use the model as implemented by the editor, and analyze them with respect to teachers’ ICT skills.

The technical skills required to use the editor are those of a table editor in a word processing office tool: add/remove lines or columns, split or merge cells,

displace an element, and copy-and-paste. All these actions correspond to mouse manipulations (left-click, right-click, drag-and-drop).

Teachers' ICT skills are assessed in a few countries only (Bakia et al., 2011), and we lack general data. However, in one investigation into the ICT knowledge and skill levels among Western Australian government school teachers (Trimmer, 2006), word processing was part of the basic suite of ICT applications used by more than 95% of teachers.

The ICT skill item map that was constructed from the analysis of teachers' skills led to a three-score division: competence scores between 0 and 39.9 (22% of teachers who typically have basic skills such as word processing and Internet), scores between 39.9 and 60.6 (53% of teachers with more advanced skills) over 60.6 (25% of teachers with even more advanced skills). The "creating tables" skill is in the middle of stage 1. In other words, the technical skills required to use the editor are considered as basic skills, accessible to most teachers.

The ediT2 editor works like a table editor rather than a spreadsheet (spreadsheet specificity is to allow formulas and programming interaction, which requires more advanced skills; although one may use spreadsheets to manage a basic table, the overall interfaces is much more complex and impressive). However, we may notice that the same study mentions that although 35% of teachers have never tried any of the spreadsheet tasks, 56% have inserted and deleted rows and columns (this skill being placed in the middle of stage 2).

Different studies have also shown that tables were simple and effective representation devices that help users shape a representation of their problems (Mangano et al., 2011; Nardi and Zarmer, 1993).

We also conducted an Internet search to collect examples of learning pre-structure sheets offered to teachers in the French context, and we found out these sheets were almost always tables, in which their rows indicated the session phases and their columns, the scenario / teaching sequence representation notions.

These elements suggest that although teachers' ICT skills are context-dependent, a model/editor requiring the technical skills of an office table-editor will in all likelihood be usable by a large set of teachers.

5.2.4 Discussion

Reconsidering the first usability test in such a way to answer the questioning of this section ("is the model/system easy to use?"), we found some positive indicators in the qualitative analysis of the teachers' responses to the questionnaire and the open discussion (preliminary results of the second test will be presented in Chapter 9): all teachers stated that the proposed notions made sense to them (although some of them suggested using some others, or some more), and easily reflected on the table representation they had built when asked to react to the proposed run-time events (fifth phase of the study).

The following are some examples of feedback: “[for the second script] I knew where I was going (…) I had “tools to think” (…) it was a pleasure to use the tool (...) it’s a genuine organizational resource (…) [the proposed notions] “work well”; “Step by step, I took ownership of the setting thanks to the array (….) it’s a skeleton (…) one is challenged by the playful aspect of the exercise”; “The tool helped me to tidy up [the script], even in my head”; “I would like to see a column to indicate the results, how students reacted, etc.”

With respect to the questionnaire statement “I found that the capacity to adapt the representation […] allowed me to reflect on the script, to refine my vision”, 3 teachers totally agreed, 1 agreed and 1 disagreed. With respect to the questionnaire statement “I think this tool could allow me to conduct my session, to reflect on what is happening during the session and how I could adapt things according to the effective unfolding of the script”, 4 teachers totally agreed and 1 disagreed. In both cases, the disagreeing teacher is the same. She mentions that she had the script in her head very clearly from just reading the text, thus the tool just helped her to organize her thoughts; and, with respect to run-time management, “It’s a help before the session, but while conducting the session I would be anchored in the setting and would not come back to it (…) but I would afterwards, to reflect on the effective enactment”. Analyzing these conceptual dimensions would require a specific qualitative study involving a larger panel and a longer time-span.

5.3 Do teachers avail themselves of the flexibility provided?

To answer the question “do teachers avail themselves of the flexibility provided?” we will consider the first usability experiment. The data for the second experiment is not available as teachers’ representations were transformed for operationalization (see Chapter 7). We narrow the question to whether teachers use the flexibility provided when asked to edit a script with the provided editor (which is the topic we consider here). Whether teachers use the editor and its flexibility within their current practices is a distinct question, for longer-term study.

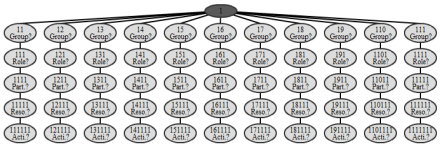
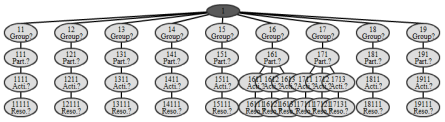
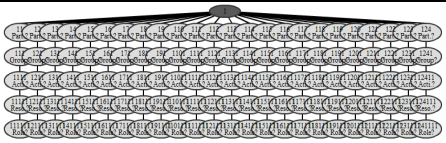
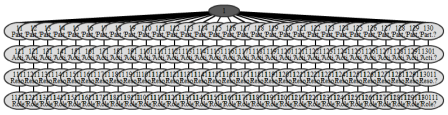
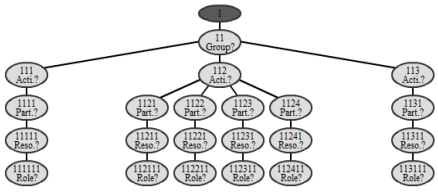
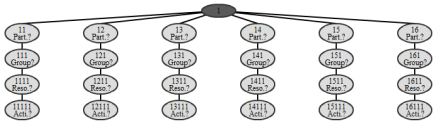
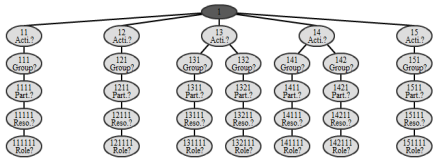
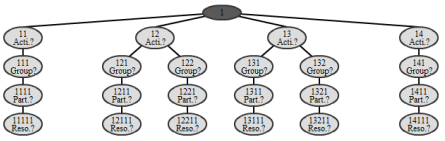
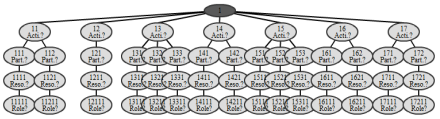
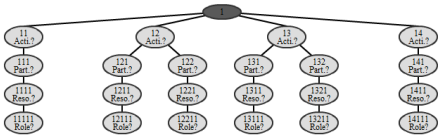
Four indicators gathered from the first usability test show that teachers take advantage of the editor flexibility.

First, with respect to representation flexibility, the seven participants involved in the experiment individually represented two different scripts, as described in Section 5.2.1. The analysis of these 14 representations’ skeletons produced shows that 12 different script-structures were used (9 different script structures for the 10 representations by teachers), with most of the involved teachers (4 out of 5) using different script-structures for the first and second script. Table 5.4 presents the scripts’ skeletons and script-structures produced by the 5 teachers.

We find this variety of script-structures particularly interesting because, in some sense, the script structure is a modeling engagement that corresponds to the

way teachers appropriate the script. This illustrates the variety in modeling preferences and ways teachers addressed these scripts, and the fact they adapt the script-structure to their perspective and to the script.

Table 5.4 Scripts' skeletons

Teacher	Representation skeletons	
	Script#1	Script#2
#1	 Group-Role-Participant-Resource-Activity	 Group-Participant-Activity-Resource
#2	 Participant-Group-Activity-Resource-Role	 Participant-Activity-Resource-Role
#3	 Group-Activity-Participant-Resource-Role	 Participant-Group-Resource-Activity
#4	 Activity-Group-Participant-Resource-Role	 Activity-Group-Participant-Resource
#5	 Activity-Participant-Resource-Role	 Activity-Participant-Resource-Role

We may notice that although some representations may be equivalent with respect to the formal model, the fact that notions are used in a given order seems to be of importance for teachers. As an example of this, in Table 5.4 we can observe that although the script#2 was represented by the teachers #4 and #5 through two different script-structures, the latter is, in fact, an extension of the former, enhanced by the Role notion.

With respect to the fact some script-structures may be considered as equivalent, however, we can mention the following interesting episode. While modeling, teacher#5 attempted to merge two cells when this was not possible given the 1-n constraint. She realized that she could easily and neatly solve the problem by changing the column order, but she explicitly decided not to do so, preferring to

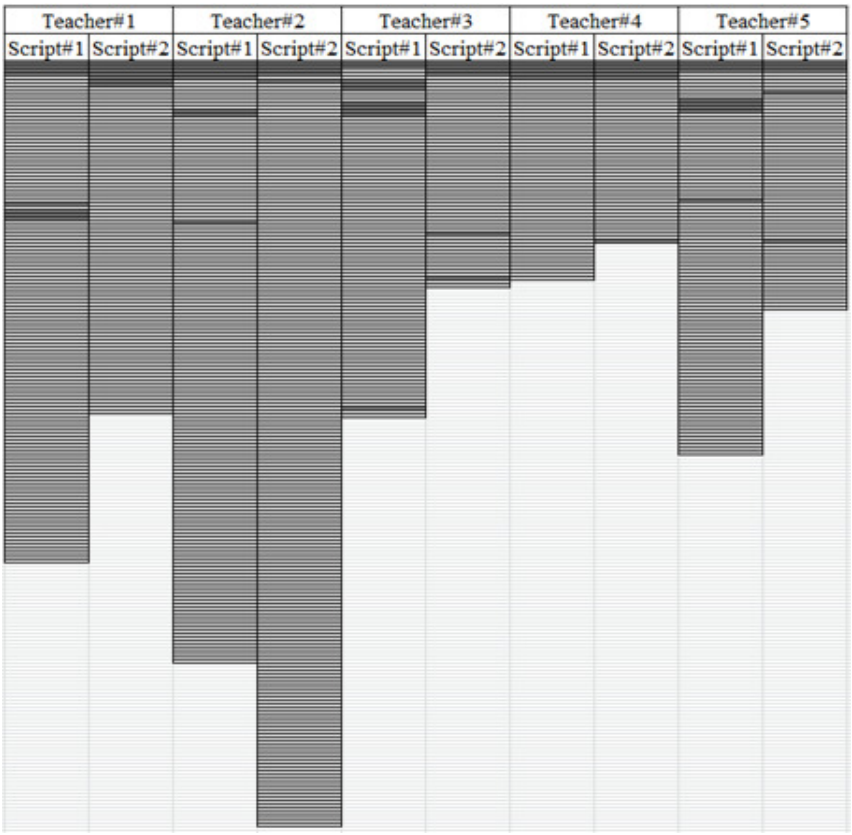
keep two separate cells with duplicate values (which was a less neat way of solving the problem), stating that "I prefer to view the script this way" (i.e., with this script-structure). This suggests that the precise script-structure is an important topic for teachers.

Second, the structure and number of items of the different trees also illustrate that teachers edit the script in different ways, although all of them developed an understanding of the script in line with the script principles (Table 5.5). In this table, the "–" symbol following the name of a notion indicates that it was not used in the script representation.

Table 5.5 Number of different involved items

Script	Number of items				
	Teacher#1	Teacher#2	Teacher#3	Teacher#4	Teacher#5
#1	Activities:9 Groups:1 Participants:5 Resources:9 Roles:1	Activities:17 Groups:2 Participants:5 Resources:9 Roles:3	Activities:3 Groups:1 Participants:4 Resources:2 Roles:8	Activities:5 Groups:2 Participants:5 Resources:2 Roles:3	Activities:7 Groups: – Participants:5 Resources:6 Roles:6
#2	Activities:4 Groups:5 Participants:4 Resources:10 Roles: –	Activities:17 Groups: – Participants:5 Resources:12 Roles:1	Activities:7 Groups:5 Participants:4 Resources:6 Roles: –	Activities:4 Groups:2 Participants:4 Resources:8 Roles: –	Activities:4 Groups: – Participants:4 Resources:10 Roles:3

Table 5.6 Modifications of script-structures



Third, Table 5.6 highlights the actions performed by the teachers to define/modify the script-structure of each one of the scenarios represented by them. Every line in grey corresponds to a teacher's action (chronological order). The darker lines correspond to actions related to the definition or adaptation of the script-structure, i.e., inserting, moving or removing columns. All this information was gathered from the teachers' traces when manipulating the editor (see Appendix C, for details).

We find particularly interesting the fact that teachers use the editor flexibility to define the script-structure they feel at ease during their first steps (and in some cases, remove columns they did not use, if any, at the end of the process), but also modify the script-structure while the modeling is advanced and on-going. Teachers do change their modeling perspective if, at some time, it appears less convenient or less adapted than expected. This suggests that, at some moments, what is under edition (analysis, correction, refinement) is not only the internal structure of the script (e.g., the list of activities or the composition of groups) but, also, the modeling perspective itself.

Table 5.7 Users' actions
Ti correspond to teachers, ID to Instructional Designer and MS to Modeling Specialist

	Participants' number of actions													
	T1		T2		T3		T4		T5		ID		MS	
	S1	S2	S1	S2	S1	S2	S1	S2	S1	S2	S1	S2	S1	S2
Total number of actions	135	95	162	206	96	61	59	49	106	67	72	118	52	66
Normalized number of actions*	135	96	162	206	96	61	64	67	106	67	84	185	52	68
Minimum number of actions**	68	75	142	189	41	52	51	44	89	60	70	129	45	53

* The number of actions is normalized. This means that when teachers use the duplicate feature (which creates in one step what may require several atomic actions) the corresponding number of basic actions is counted. The data is cleaned. For instance, in the context of the first script, some teachers first checked how the editor reacted by inserting and immediately removing a column. These actions were not taken into account. Similarly, some teachers pre-created a series of empty rows and, when over with the representation, deleted the ones that were not used. These actions were not counted.

** The minimum number of basic actions denotes the actions that would be needed to represent the final structure as produced by the teachers within a straightforward process.

Finally, as contextual information, Table 5.7 mentions the number of actions performed by the users that participated of this experiment, and indicates the minimal number of actions that would have been necessary to obtain the same representations in a straightforward way. This data is not to be over-interpreted. The difference corresponds both to the users' explicit changes of some previously decided design options and to events that are much more difficult to interpret properly. For instance, some users punctually use the table as an untidy draft, whereas at some other step, their process is much more structured. Moreover,

teachers do not necessarily attempt to be efficient, acknowledging they use more actions than necessary but not bothering as they know they can change the representation sufficiently easily to obtain their desired end result with little effort (and this is very interesting with respect to the system's usability). And, of course, there are a few simple mistakes such as dropping an item in a different cell than the one that was targeted. However, a significant part of these actions correspond to adjustments.

As highlighted by the debriefing session and verbalizations during the process, teachers do not build an accurate and precise representation in their heads and then transcript it with the editor, but rather refine the script while editing it. As examples of verbalizations: "No, I prefer splitting this activity in …"; "Actually, I think I might differentiate the roles …"; "It might be interesting to …".

5.4 Does a table notation allow representation of a wide range of scenarios?

As raised in Chapter 2, scenario editors are usually designed to make it possible to represent a range of scenarios that can be expressed with a given EML and its objectives or constraints as, for instance, standardization and industrialization for IMS-LD, or operationalization within a given framework such as LAMS. Here, the expressiveness scope is not defined by an EML but by a structural aspect: scripts that can be represented with a flat table representation.

As the overall approach is to allow additional services (see Chapters 6, 7 and 9), expressiveness may be more or less enhanced. However, to evaluate the approach's pedagogical expressiveness we stuck to the basic representation and considered this question in different ways, and in relation to matters under study.

A first evaluation action was conducted in the context of the two experiments reported in Section 5.2. Let us recall that the French teachers were asked to represent scripts as they would like to conduct their sessions in the classroom, while the Spanish academics have been interested in terms of scripts' operationalization with Moodle. In both experiments, none of the teachers raised any issue related with pedagogical expressiveness and the fact that they could represent the scenario as they wished given their concerns. The fact that the offered notions were satisfactory is not surprising as it has been proven they could be used to represent a variety of scenarios (Kobbe et al., 2007). What this study confirmed is that the table was expressive enough for teachers to express their designs.

Another set of evaluation actions was exploratory.

First, we collected 25 CSCL scripts from the literature (see Appendix D, for details). All of them could be represented as tables. This representation was more or less complete and more or less straightforward. For scripts that included loops, these repetitions could not be directly designed as such, but could be represented by duplicating some of their rows and/or items. For some scenarios, the

representation was not complete (i.e., not sufficient for direct operationalization) because the scenario included specific constraints requiring additional means. For instance, one scenario is based on the fact that the enactment system requires students to agree on an answer before moving to the next step (Roschelle et. al, 2009; see Appendix D (Section D.9), for details). This may be represented by using a dynamic principle (see Section 6.2), or an additional workflow representation (see Section 6.4).

Second, we considered the scenarios (referred to as "teaching strategy templates") elaborated by the LAMS community, such as role-play, problem-based learning or predict-observe-explain (LAMS, 2013). Here again, most structures are simple sequences of tasks, and branching constructions may be represented by using sub-rows. For instance, the role-play scenario general structure is a linear list of steps (e.g., considering documents concerning a certain subject), branching that leads different students to consider different activities (e.g., organizing a vote and making the ones that vote for and against consider different tasks) and then, all students are regrouped to perform final activities. Such a structure may be represented as a table in the same form as the jigsaw in Fig. 4.3. As for the CSCL example previously described, LAMS's representation of this type of scenario is based on features related to the operationalization approach: dynamic distribution of students based on its Voting tool and scheduling implementation (students meet when they have all gone through their branch). These features are native for LAMS, which addresses both representation and operationalization means. For a representation that is decoupled from the enactment framework, they must be implemented as enhancements (see Sections 6.4 and 7.1, for details).

A final evaluation action was conducted with the generic version of ediT2 (discussed in Chapter 8), but we will report its conclusions here. We considered the SceDer Authoring tool (Section 8.3.3; Niramitranon, 2009), which has been evaluated as capable of enacting 9 between 13 high-priority educational scenarios (Niramitranon et al., 2010). We showed that the table/tree model offers the same expressiveness by implementing a SceDer adaptation of ediT2.

These elements suggest that basic tables can be used to represent a wide range of scripts. Moreover, some scenarios requiring complex sequencing or dynamic principles may be represented via enhancements (see next chapter). Nevertheless, it may only be possible to sketch some scenarios requiring complex relationships or layered stratification, or represent them in an indirect way (e.g., if having to duplicate too many rows). A table presents some expressiveness, but is not to be thought of as a replacement for more comprehensive means.

6 Enhancing the editor/model through the development of advanced functionalities

6.1 General considerations

6.2 Instantiation support

6.3 Respect of constraints

6.4 Representation of complex scheduling structures

6.5 Simulation

This chapter builds on and extends some ideas and concepts presented in the CSCL International Conference (2013), and in our papers published in the ijCSCL (2012) and in the Computers and Education (2015) journals.

Le modèle table/arbre et l'éditeur ediT2 peuvent être utilisés sans aucun service additionnel, c'est-à-dire, simplement comme un moyen de base pour représenter et partager des scénarios pédagogiques. Cependant, dans cette proposition, l'expressivité informatisée est limitée et différents services proposés par d'autres plateformes pédagogiques ne sont pas offerts. Conformément à notre approche générale, nous avons examiné comment étendre les fonctionnalités d'édiT2, sans rater ses avantages fondamentaux de simplicité et de flexibilité, afin d'envisager les services avancés suivants : support à l'instanciation ; respect de contraintes ; représentation de structures complexes de planification ; et simulation (les services de mise en œuvre et de monitoring seront présentés respectivement, dans les chapitres 7 et 9).

Dans le modèle T^2 de base, la formation de groupes et la répartition de tâches sont représentées par extension, c'est-à-dire, à partir de la description des items associés à chaque nœud/cellule de l'arbre/table. Cette représentation est suffisante dans de nombreux cas, mais elle peut poser des problèmes dans d'autres (la création de ces listes peut devenir très complexe ou intraitable lorsque les scénarios impliquent de nombreux étudiants). D'un point de vue général, surmonter cette limitation ne demande pas la modification des bases du modèle T^2, mais plutôt la modification de sa mise en œuvre. Pour répondre à cette problématique, nous avons développé une extension d'ediT2 afin de supporter l'instanciation de scénarios d'apprentissage à partir de patrons pédagogiques prédéfinis.

Un scénario créé par un mécanisme d'instanciation automatique respecte naturellement les contraintes de son patron. Cependant, cela peut ne pas être le cas si un script est construit de zéro, ou créé par instanciation et ensuite édité par l'enseignant. Offrir des patrons prédéfinis, un support à l'instanciation et des mécanismes de vérification de contraintes permet des cas d'utilisation comme, par exemple : générer un scénario à partir d'un patron ; utiliser l'interface de la table pour éditer le scénario généré ; et recevoir des messages d'avertissement si l'édition en question modifie le scénario de telle façon qu'il ne respecte plus les contraintes prédéfinies par le patron.

Une autre limitation intrinsèque du modèle T^2 est la représentation de planifications complexes : la table ne permet que la représentation d'un simple ordre linéaire descendant de ses lignes. La modélisation de mécanismes complexes de séquencement et, plus généralement, d'une représentation visuelle et intuitive de dimensions dynamiques de scénarios, demandent des langages de modélisation fondés sur graphes. L'approche que nous développons pour répondre à cette problématique vise à combiner les deux représentations en inter-opérant ediT2 avec un workflow.

L'utilisation de technologies de workflow offre différentes perspectives intéressantes dans le domaine de la représentation de scripts collaboratifs dont la possibilité de simuler certaines propriétés des scénarios. A titre d'exemple, nous

avons adapté ediT2 afin de tester différentes techniques de formation de groupes. Cette fonctionnalité peut être utile si, dans un scénario avec un nombre considérable d'étudiants, il peut être difficile d'identifier la façon dont les étudiants devraient être groupés. Avec la simulation, de nouveaux groupements peuvent être composés jusqu'à ce qu'un réglage souhaitable soit trouvé.

6.1 General considerations

The table/tree model and ediT2 implementation may be used free of any other services, as a basic way to design and share pedagogical scenarios. This is an interesting feature in itself (Harrer et al., 2007). However, within this basic implementation, computational expressiveness is limited and different services offered by state-of-the-art systems are not offered.

In line with our general approach, we have considered how to extend the ediT2 functionalities without losing its native simplicity and flexibility advantages. This is based on the manipulation of the tree model. How model-driven approaches to ELMs allow implementing different specific services on the basis of model manipulations has been mentioned as an interesting potentiality of such an approach (Laforcade and Choquet, 2006).

In Section 2.5 we have listed different matters of concerns. The first two of them have been addressed in Chapter 4 (conceptual support and script flexibility) – in this same chapter the representation flexibility concept has been introduced and also discussed. In the current chapter, we address instantiation support, respect of constraints, representation of complex scheduling structures and simulation. Operationalization and monitoring modules will be addressed in Chapters 7 and 9, respectively.

6.2 Instantiation support

Within the basic T^2 model, group formation and task distribution are represented by-extension, i.e., listing the items associated with nodes. While this may be sufficient in many cases, it may be an issue in others. We may dissociate two cases, instantiation before the session and dynamic mechanisms.

Instantiation before the session may be an issue if the script involves many students. Creating these lists may become time consuming and/or over complex. For instance, managing reciprocal-teaching and jigsaw scripts' mechanisms (Figure 2.1 and Figure 2.2, respectively) for 4 or 6 students is easy, but may become intractable for 20 or 200 students.

Dynamic mechanisms are required when the script involves principles such as "G2 is made up of the five students who finished activity A1 first", or "G1 is made up of students whose answers to the quiz Q1 were correct". This requires the editor to be interoperated with an enactment framework and retrieve data related to the script enactment.

From a general perspective, overcoming the "by-extension description" limitation does not need to modify the bases of the T^2 model but, rather, its implementation.

For instance, the Universanté script (Dillenbourg and Jermann, 2007) requires, in some places, to refer to groups studying similar clinical cases and, in others, to groups made up of students from the same/different countries. Representing such scripts may be addressed by representing the necessary information in the participant data-structure (e.g., country) and modify the group definition box to allow defining a group as the intersection, union or crossing of other groups, i.e., associating nodes with algebraic constructions and introducing configuration interfaces in the groups (etc.) definition boxes. This is standard engineering work. Representing dynamic mechanisms such as "a country-theme set is made up of students such that the "from a same country" and the "worked at a same theme" conditions hold" is similar.

As another example, the ArgueGraph script (Dillenbourg and Hong, 2008; see Appendix D (Section D.10), for details) is based on proposing a task that identifies students' opinions on an issue and then, forming groups with students of conflicting opinions. Such scenarios may be implemented by mixing a direct-manipulation (by-hand; definition of groups) for the first phase, and a dynamic mechanism for the second one.

As a way to show how the T^2 model allows by-intension descriptions and configurations, we have considered instantiation before the session issues. More precisely, we have developed an extension of the ediT2 editor to implement a pattern-based approach close to Collage, i.e., grouping students and distributing resources according to a script pattern (Hernández-Leo et al., 2006, 2010).

We considered the classical reciprocal-teaching and jigsaw script patterns. Let us describe the latter. The configuration data is defined by (1) the list of students, (2) the list of topics/themes (e.g., two topics: "insulation" and "heating"), (3) the list of resources tagged with their related topic (i.e., an indication which documentary resources address "insulation" and which ones address "heating"), and (4) a set of parameters (participants, topics and resources can be defined in the editor or imported from an enactment platform, if any).

These parameters include (i) the required number of students per group (which indirectly defines the number of groups), and how to manage odd cases, and (ii) the way resources should be distributed. The top of Figure 6.1 presents a possible example of interface to inform these parameters. The bottom of this same figure presents another example, when taking into account the reciprocal-teaching script.

Given the script principles and the configuration data, a specific algorithm (Figure 6.2) generates a solution, i.e., a tree, by associating expert groups with topics (e.g., for 5 groups and 2 topics, 3 of them will work with one topic, and the other 2 with the other one) and, then, from the expert groups, distributing the resources to the participants for the initial phase and creating the jigsaw groups. If

the configuration accepts no solution, the algorithm raises the impossibility and the reason(s) for this.

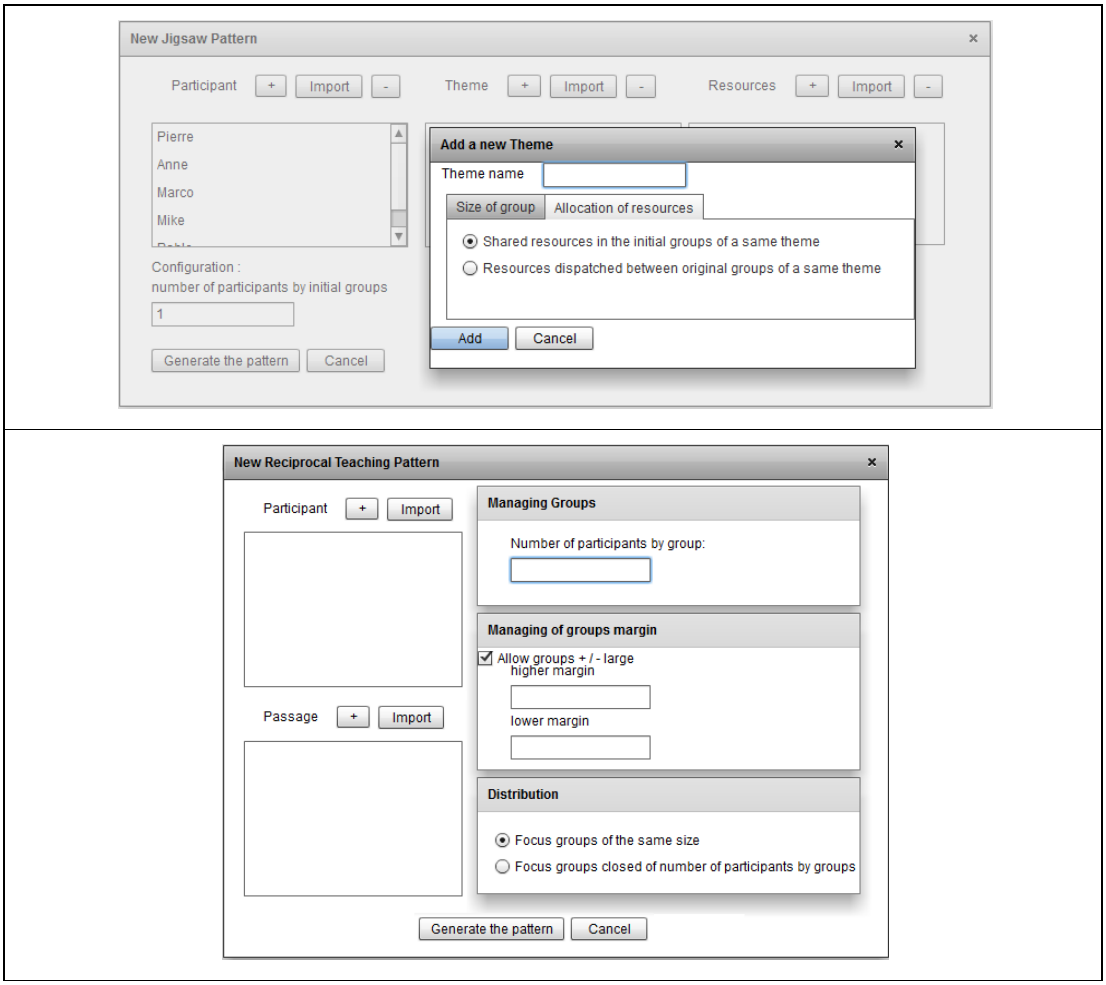


Figure 6.1 Possible interfaces to inform jigsaw (top) and reciprocal-teaching (bottom) scripts' data configuration

Algorithm: Generating groups and distributing resources (for jigsaw scripts)

1. Create the sub-tree nodes denoting the expert groups
{When the numbers of groups and students do not fit, two strategies are possible: increasing the number of students per group or creating an additional small group}
2. Associate expert groups with subjects
3. Distribute resource(s) among expert groups
{Different strategies are possible depending on the number of resources in particular}
4. Create the sub-tree nodes denoting the individual phase
5. Distribute resource(s) among the individuals
{Different strategies are possible, e.g., students receive all resources related to their subject of study or a subset of the resources to be used by their respective groups in the expert phase}
6. Create the sub-tree nodes denoting the jigsaw groups
7. Distribute resource(s) among the jigsaw groups
{Different strategies are possible, e.g., each student brings the resource(s) used by him/her in the individual phase}

Figure 6.2 Instantiation mechanisms for jigsaw scripts

If we take the case of 22 students, a target number of 4 students per group and the option of extended groups (i.e., some groups will be composed by more than 4 participants – 3 groups with 4 students and 2 other with 5 students), a solution is a tree with 78 nodes (a table with 32 sub-rows; Figure 6.3). Indeed, such a description would be difficult to manage by-hand.

Activity	Group	Participant		Resource
Individual Task		Pierre		R1
				R2
				R3
		Anne		R1
				R2
				R3
		Marco		R1
Expert Task	Expert Group #1	Pierre	Anne	R1
		Marco	Mike	R2
		Pablo		R3
	Expert Group #2	Pericles	Hannah	R5
		Isabelle	Joelma	R6
		Dominique		R7
	Expert Group #3	Marie	Maria	R9
		Neuza	Leonardo	R10
Jigsaw Task	Jigsaw Group #1	Pierre	Pericles	R1
		Marie	Paul	R2
				R3
	Jigsaw Group #2	Fabrice	Anne	R21
		Hannah	Maria	R22
				R23
	Jigsaw Group #3	Pietro	Laurie	R13
		Marco	Isabelle	R14

Figure 6.3 Automatic generation of a jigsaw script (the tree (left) and parts of the table presentation (right))

However, once the tree is automatically generated, the table representation is presented and can be used to manage slight modifications using the editor native features, before or during the session (e.g., breaking down an activity into two activities, adding a resource or moving a student from one group to another). In this case, a variety of options are open such as allowing the teacher to apply any changes or checking whether the changes applied are contrary to the jigsaw intrinsic constraints via control rules (this is discussed in next section).

6.3 Respect of constraints

Instantiation and constraint-checking support are of interest when facing complexity. Specific study has shown that supporting teachers in preparing the group distribution before the class and adapting the groups to unexpected situations

was useful when there are many constraints to control and a large number of students (Pérez-Sanagustín et al., 2009).

A script created by an automatic instantiation natively respects the pattern constraints. However, this may not be the case if a script is defined by hand, or created by instantiation and then edited by the teacher. Offering scenario patterns, instantiation support and constraint-checking mechanisms facilitate use-cases such as: generating a scenario from a pattern and the requested data; using the table interface to edit the scenario generated as necessary; and receiving a warning if such editing goes against the pattern's intrinsic constraints.

Table 6.1 presents some of the constraints associated to the jigsaw pattern and how they may be checked using the tree representation.

Table 6.1 Examples of pattern constraint-checking mechanisms (to the jigsaw pattern)

#	Jigsaw constraints	Implementation principle within the tree representation
1	(Expert/Jigsaw) participants are uniformly distributed on groups	Compare the cardinality of participants nodes (taking into account the Expert/Jigsaw groups)
2	All script phases (individual, expert and jigsaw phases) are performed by all participants	For each of the individual, expert and jigsaw tree branches, get the items of the "Participants" level and check the lists for correspondence
3	The number of expert/jigsaw groups is the same	Check the number of nodes in the group level for expert/jigsaw activities
4	In the individual phase, all participants receive resource(s) related to one subject only	For all three of these constraints the principle is similar: in the phase branch, check the subjects of the resource nodes (the checked criteria varying accordingly)
5	In the expert phase, resources associated to a group are related to the same subject	
6	Each jigsaw group has, at least, one resource of each subject	
7	Each jigsaw group is made up of students who did not work together in the expert phase	Check the jigsaw and expert subgroups participant lists for intersection (pair by pair)

- Constraint#1 (The number of participants present at Expert and Jigsaw groups should be uniformly distributed): Not satisfied
  - The grouping [Joelma,Leonardo,Brigitte,Nicolas,Pablo,Dominique] should have 4 or 5 students, but, however, it has 6 participants
- Constraint#2 (All phases must be performed by all participants): Satisfied
- Constraint#3 (The number of Expert and Jigsaw groupings should be the same): Satisfied
- Constraint#4 (In the Individual phase all participants receive, individually, resource(s) related to only one topic): Satisfied
- Constraint#5 (In the Expert phase all resources used in each expert group must be related to a same topic): Satisfied
- Constraint#6 (In the Jigsaw phase each jigsaw group must have, at least, one resource of each topic): Satisfied
- Constraint#7 (Each group in the Jigsaw phase must be composed by students who did not previously work together in the Expert phase): Not satisfied
  - Joelma has already worked together with Dominique
  - Dominique has already worked together with Joelma

Figure 6.4 Firing out the pattern constraints verification module (to the jigsaw pattern with "22 students shared in extensible groups of 4 students" as configuration parameters)

An example is presented in Figure 6.4, based on the jigsaw script constructed from the data set presented in the last section (22 students shared in extensible groups of 4 students). Using the ediT2 table, we moved the student Joelma from a group to another (both groups in the jigsaw phase) and after, we run out the constraints verification module to analyze if the script still satisfied the pattern constraints.

As we can observe in Figure 6.4, two between seven constraints were not satisfied for the new Joelma's jigsaw group: the first issue informs that its cardinality is not allowed (Constraint#1); and the second one, that two of its members has already worked together in the expert phase (Constraint#7).

Offering such constraints-checking mechanisms requires (1) identifying script patterns (proposals already exist, e.g., Hernández-Leo et al., 2010), and; (2) defining and implementing the corresponding algorithms, which is technically of little difficulty. This may be an important price to pay for individual uses, but makes sense in relation to pattern repositories development.

It may be noticed that, in the case of original scripts, as there is no predefined model, support such as automatically distributing students according to the script's principles or checking intrinsic constraints is not possible. An option, however, is to predefine and implement general principles (e.g., "all students must be involved in a collective activity" or considerations such as gender issues) and offer teachers to check the ones that appear pertinent for the script they designed.

6.4 Representation of complex scheduling structures

An intrinsic limitation of the ediT2 model is the representation of complex sequencing: tables allow representing sequences through simple top-down linear ordering of script-components (phases or tasks are to be taken one after the other as listed in the script).

If used for reflecting on the script only (i.e., human interpretation) and considering simple cases, lack of sequencing expressiveness is not necessarily an issue. For instance, in the jigsaw script presented in Figure 4.3, the fact that the jigsaw activity ("Crossing groups") takes place after the focus activity ("Identify techniques") is implicit but obvious (they are in sequence, and the resources produced in the context of the latter are inputs for the former). Similarly, the fact that the focus groups ({e1, e2}; {e3, e4}) – and, later on, the jigsaw groups ({e1, e4}; {e2, e3}) – may work in parallel is again implicit but rather intuitive.

This, however, would not work for a complex scheduling as, for example, in the script presented in (Roschelle et al., 2009; see Appendix D (Section D.9), for details), which involves repetitions and conditions. Although it is possible to find

more or less explicit and neat ways to represent repetitions or rotations from ediT2 (e.g., duplicating some of rows and/or items), this is more difficult for conditions.

Enhancing the table editor with *ad hoc* means (e.g., using constructions based on numbering rows) is possible, but this goes against the approach rationale of exploiting the intuitiveness of table structures, and, also, would remain limited.

The modeling of complex sequencing mechanisms and, more generally, visual intuitive representation of the dynamic dimensions of scripts require languages building on the top of graph-based representation (Botturi et al., 2008) and workflows. As examples: COW (Vantroys and Peter, 2003), Flex-eL (Sadiq et al., 2002), (Haake and Pfister, 2007), MoCoLADe (Harrer et al., 2007), LeadFlow4LD (Palomino-Ramírez et al., 2008)).

The utilization of a workflow-like editor is not a basic ICT skill, and requires some training. A work related to IMS-LD highlighted that using representations inspired from data or process modeling, such as XML-like trees or process charts may be an issue for adoption (Neumann et al., 2010). Nevertheless, such representations may be required and, in such cases, many languages exist (see above).

Another approach is to mix both representations, i.e., extending the ediT2 editor with means for representing complex sequencing via a workflow representation.

Using a complementary workflow-based representation makes different services possible.

First, to control student enactment of the script, e.g., to prompt them with activities or make resources such as documents or tools accessible based on the script scheduling. This goes in the direction of developing workflow-based scripting languages similar to the one presented in (Haake and Pfister, 2007).

Second, to configure an enactment framework. As an example, it has been shown how such a statechart representation could be used as a basis to configure the CeLS platform (Harrer et al., 2009; Ronen et al., 2006).

Third, to assist teachers in monitoring their learning sessions, offering an instrument of reflection when the script enactment does not correspond to the design specification. Malzahn et al. (2008) have explored the elaboration of a player to provide to teachers means of supervision of CSCL scripts modeled via MoCoLADe, and monitored from CopperCore.

Fourth, to implement pre-session simulations of the script, as a support for teachers to test options (e.g., the scheduling of the activities or the composition of the groups) while editing complex scripts (Harrer et al., 2007; Sobreira and Tchounikine, 2013).

To study how such an ediT2 / workflow engine interoperation may be addressed, we have designed and implemented an ediT2 extension module that, focusing on the Activity-Group-Participant-Resource script-structure, transforms a T^2 table-script into a jBPM statechart skeleton. jBPM (2012) is an open-source workflow engine aiming to manage business processes through building blocks such

as *split*/*join* nodes for branching and synchronization, or *for-each* nodes to represent repetitions.

The transformation process principles is presented in Figure 6.5 (other options are possible) from the jigsaw example presented in Figure 4.3. The Figure 6.6 shows the script skeleton generated as it appears when opened within the jBPM platform, an Eclipse plug-in (Eclipse, 2012).

Algorithm: From ediT2 to a jBPM representation

1. A "Start" node and an "End" node are automatically created. The "Start" node is linked to a "Setting objects" operation which contains the code automatically generated to set the different objects' values, e.g., the list of students per groups.
2. Each activity is transformed into a set of three operations: the first one corresponds to the activity itself (e.g., "Read the general text"). It is connected via a split connector to a "Problem" operation (which denotes the state to be reached if the activity fails) and a "Next" operation (which denotes the state to be reached if the activity succeeds).
3. When an activity is associated with *n* different groups (or sets of participants), it is automatically broken down into *n* operations which correspond to the *n* instances of the corresponding activity. For example, the "Identify techniques" activity is associated with two groups G1 and G2; this is translated into two instances of the "Identify techniques" operation, one corresponding to G1 and the other to G2.
4. An initial graphical position is set for the different operations and their associated connectors, ordering the operations by the table top-down structure and juxtaposing multiple instances of activities, if any.

Figure 6.5 Interoperating ediT2 and jBPM

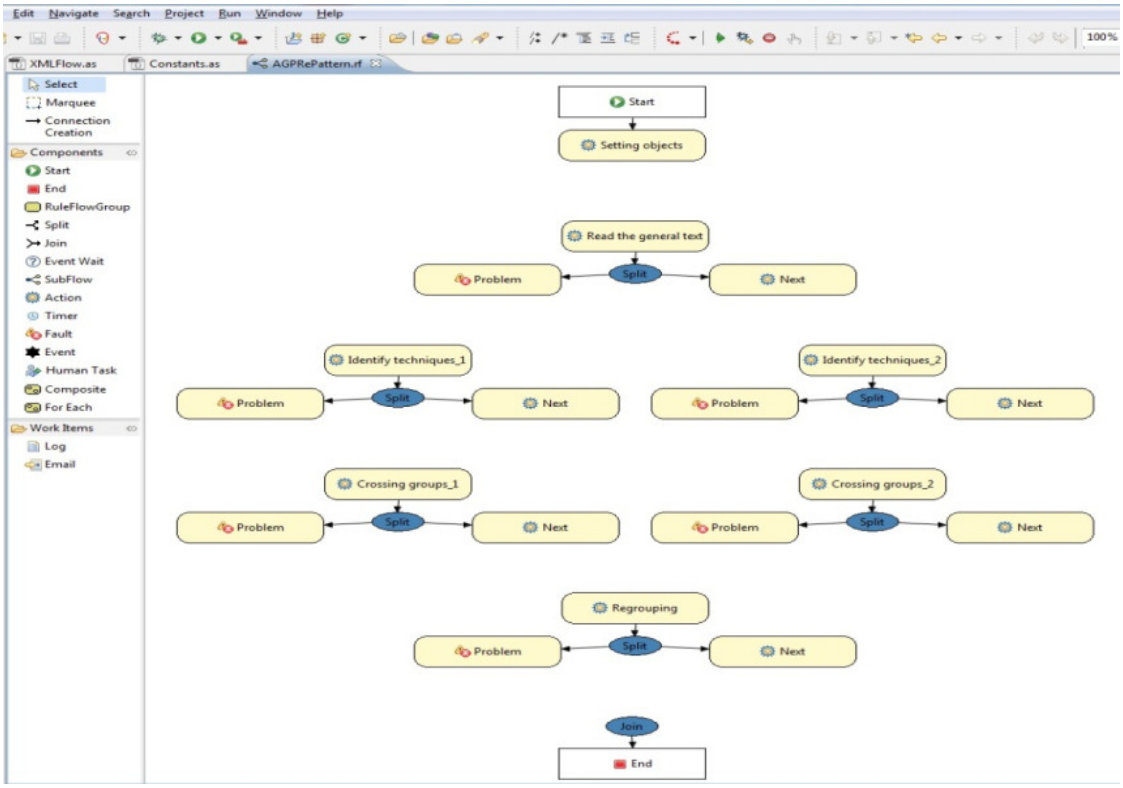


Figure 6.6 A generated jBPM skeleton (from the jigsaw script presented in Figure 4.3)

From a user point of view, the overall process is as follows:

1. The script is sketched using the table representation.
2. The user hits the “ediT2-to-rf” button (“rf” is the jBPM’ files extension) in the ediT2 interface. This generates an XML representation of the script mapped onto the jBPM’ concepts, i.e., the workflow skeleton.
3. The user opens the resulting XML file with the jBPM Flow editor and completes the model by drawing the connections denoting the flow. In this example, the actions to be proceeded are (all these actions correspond to simple mouse manipulations):
    a. Indicate that the first activity is “Read the general text”: draw a connection from the “Setting objects” operation to the “Read the general text” operation.
    b. Represent the scheduling mechanism (parallelism) of the “Identify-techniques” activity. First, create a split node, and connect the “(Read the general text)Next” operation to it. Then, draw two output connections towards the “Identify techniques_1” and “Identify techniques_2” operations (which correspond to the activity “Identify techniques” for G1 and G2, respectively). Set the split connector to “AND” to state that when the flow reaches the split connector, the two following operations must be launched simultaneously (in parallel).
    c. Represent the crossing mechanism. First, create a join node, and connect the “Next” operations of the two “Identify techniques” instances to it. Second, create a split node, connect the join node just created to it, and draw output connections towards the two “Crossing Groups” instances.
    d. Represent the regrouping mechanism. In a similar way as above, join the “Next” operations of the two “Crossing Groups” sub-activities.
    e. Indicate that the “Regrouping” activity is the final one: draw a connection from the “(Regrouping)Next” operation to the End node (in this case, the join attached to the End node is useless and can be deleted).

This example highlights the fact that the T^2 model can easily be completed and transformed into a statechart. Part of these transformations may be automatized by the generator that could, for example, suggest that the first script-component is automatically connected to the “Setting objects” operation, or deduce part of the script scheduling from the resources ending with .IN or .OUT tags. The upper part of the workflow as completed by this process can be seen in Figure 6.7.

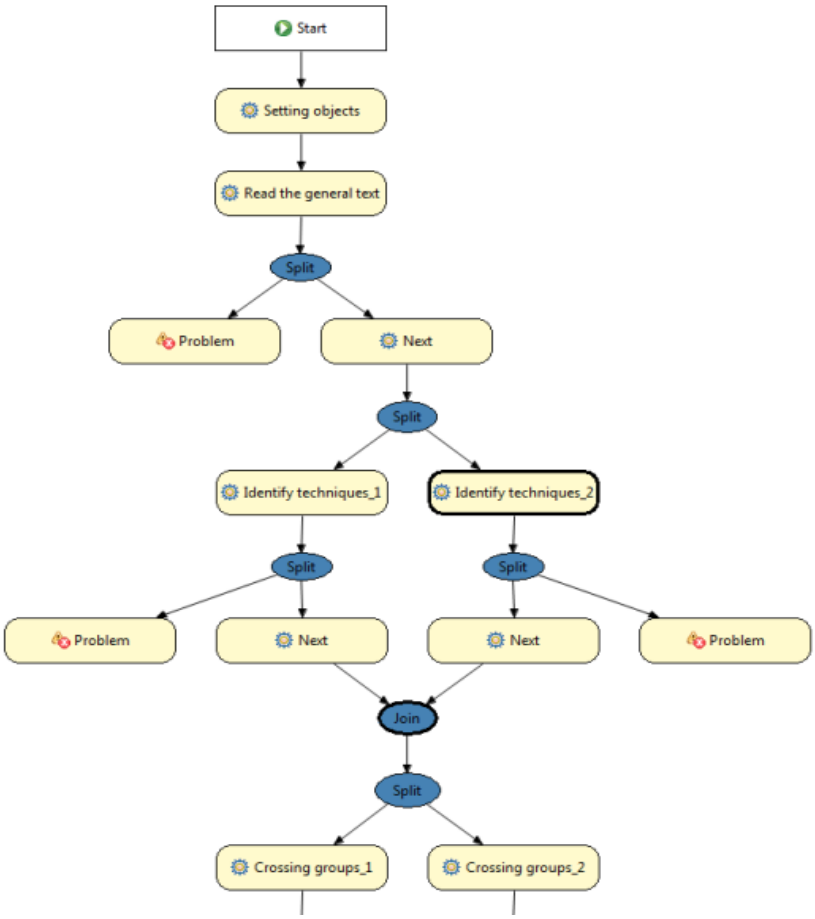


Figure 6.7 (Part of the) Completed workflow

6.5 Simulation

As discussed in last section, the use of workflow technologies may offer interesting perspectives in the CSCL scripts representation domain. Among the opportunities discussed, one interesting possibility presented is their utilization for simulating some script features.

As an example of application, we have customized the transformation process just discussed to generate operations nodes in a way that allows for testing different group formations (various authors have raised the fact that teachers could benefit from simulations to refine parameters such as grouping (Weinberger et al., 2009)).

Such a feature may be useful if, given a large number of students, it can be difficult to identify in which way students should be paired. A possible approach is to model students' profiles, form groups, and simulate the script enactment (i.e., make the computer compute the fact that activities are achieved or not according to

students' profiles). This makes sense if and only if students' profiles can pertinently be defined. Here, we will just use this as an example to simulate the script.

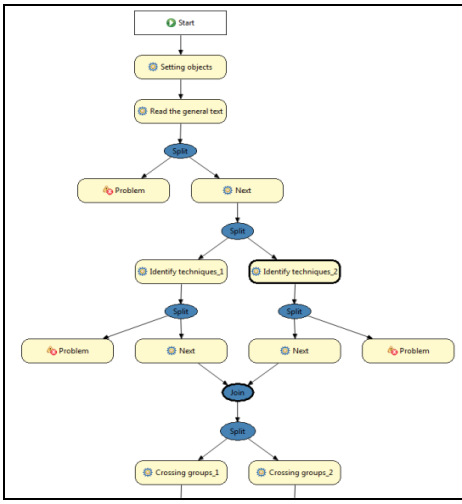
The adopted trivial modeling is as follows. Each student is associated with an activity-skill value (a value between 0 and 1 per activity) and a peer-collaboration value (a value between 0 and 1 per peer). In a similar way, each activity instance is associated with two threshold values related to skill and collaboration, respectively. An activity is considered as achieved if the involved group workforce and group collaboration values are above the corresponding thresholds.

Given a group formation and an (instance) activity, the simulation calculates the group workforce and collaboration according to the following formulas (here *n* represents the number of students participating in the (instance) activity):

$$GroupWorkforce = \sum_{i=1}^{n} activitySkill_{Student_i} \qquad GroupCollaboration = \sum_{i=1}^{n} \frac{\sum_{j=1(j\neq i)}^{n} peerCollaboration_{Student_i Student_j}}{n-1}$$

The code of the different operations is automatically generated from the ediT2 model in the jBPM XML representation. Each operation is associated with the corresponding activity's list of students as defined in the ediT2 interface. Therefore, once the workflow has been completed, the simulation can be launched.

The process goes from one node to another following the different simple, join and split connections. Activities are defined as failed or succeeded according to the students' associated values, the activity's associated threshold and the used formulas (all these features being easily modifiable). If an activity fails, the flow reaches the corresponding "Problem" node and stops the simulation; if it succeeds, it reaches the "Next" node. "Problem" and "Next" nodes are associated with some code to print messages on the console.



Activity: Read the general text
Participants: e1, e2, e3 and e4.
Resource: General text.IN.
"Read the general text" performed with success!

Activity: Identify techniques_1
Participants: e1 and e2.
Resources: Insulation text.IN and Insulation list.OUT.
"Identify techniques_1" performed with success!

Activity: Identify techniques_2
Participants: e3 and e4.
Resources: Heater text.IN and Heater list.OUT.
"Identify techniques_2" failed because the skill summation (0.7) is less than the workforce threshold (1.0).

Figure 6.8 (Part of the) Completed workflow (left) and simulation (right)

Figure 6.8 shows an example of execution. On the left side of the interface we can see the workflow being run within the jBPM debugger facility. The different operations are highlighted one after the other, whilst a message is printed on the console (right part of Figure 6.8). Here, the process is stopped on the join connector because the activity “Identify Techniques” for G2 has failed.

New groupings (new skill, collaboration or threshold values) can be edited and tested via the current simulation. The simulation can also be adapted by, for instance, modifying the connections or the criteria attached to the join or split nodes.

7 Operationalization

7.1 General considerations

7.2 Middleware-based strategy

7.3 *Ad hoc* strategy

7.4 Focus on technical representation transformations

7.5 Discussion

This chapter builds on and extends some ideas and concepts presented in our paper published in the Computers and Education journal (2015). Also, it represents part of the results achieved from a collaborative research developed with the GSIC group and, in particular, with the colleague Luis Pablo Prieto (University of Valladolid, Spain).

Dans notre approche de découplage général, le service d'opérationnalisation est adressé via la configuration de plateformes de mise en œuvre existantes, à partir de notre modèle pivot. Dans ce chapitre est présenté comment nous avons intégré notre outil (ediT2, avec lequel les enseignants représentent leurs scénarios) avec un system externe de gestion de contenu éducatif (Moodle, avec lequel les étudiants déroulent ces scénarios) à travers deux stratégies différentes : tout d'abord, à partir d'un intergiciel et ensuite, à partir d'une mise en correspondance directe (mapping *ad hoc*).

Le but d'un intergiciel est de faciliter l'interconnexion de différents systèmes en cachant ses détails internes. Afin d'étudier si cette approche était possible, nous avons utilisé l'intergiciel GLUE!-PS, dont l'objectif est de faire la correspondance semi-automatisée, à partir d'adaptateurs, entre différents outils pédagogiques et différentes plateformes virtuelles d'apprentissage. Utiliser GLUE!-PS pour inter-opérer ediT2 et Moodle n'a demandé que le développement d'un adaptateur pour traduire le modèle conceptuel de la représentation table/arbre dans le modèle de données de l'intergiciel.

La stratégie *ad hoc* est basée sur un développement informatique rapprochant les structures de la table et de la plateforme cible de mise en œuvre. La création d'un cours Moodle à partir de la représentation d'un scénario modélisé via ediT2 a été développée à travers des manipulations directes de la base de données Moodle. Les activités d'ediT2 (représentées dans la table par des lignes et sous-lignes) ont été directement transformées dans des cours Moodle.

Le principal avantage de la stratégie fondée sur un intergiciel est le fait qu'un seul mapping (entre ediT2 et l'intergiciel) est nécessaire. L'intergiciel permet ensuite le déploiement de scénarios construits via ediT2 sur différentes plateformes de mise en œuvre (dont Moodle). Comme inconvénients, nous pouvons mentionner que cette approche n'est pas souhaitable dans des situations de configuration dynamique et de flexibilisation en temps réel, car cela demande de re-effectuer la démarche complète. Par ailleurs, le mapping est générique et n'est possible que si les modèles conceptuels correspondent.

La stratégie *ad hoc* est plus flexible que la stratégie par intergiciel, permettant le développement de fonctionnalités spécifiques selon certains contextes. Cela permet par exemple l'interconnexion entre des aspects de représentation et de mise en œuvre pour améliorer le dynamisme des scénarios, ou la réduction de l'écart entre des outils de conception de scénarios pédagogiques et des plateformes de mise en œuvre. Un autre avantage de cette stratégie est le fait que l'on peut décider quand les activités doivent être déclenchées sur Moodle, ce qui permet de gérer les choses étape par étape (alors que toutes les activités sont créées d'un coup dès le départ dans l'approche intergiciel). Cependant, la stratégie *ad hoc* présente aussi quelques inconvénients : elle n'est pas évolutive,

et elle peut conduire à réinventer localement du code qui peut avoir déjà été produit dans l'approche intergiciel.

7.1 General considerations

Within our overall decoupling approach, operationalization is addressed via the configuration of existing enactment frameworks from the pivotal-model.

Examples of targetable platforms include CSCL-specific frameworks giving importance to the implementation of complex workflows (such as in the MoCoLADe-CeLS integration, discussed in Section 2.3.2) or general LMSs, offering general features such as means to manage a repository of resources (e.g. documents, quizzes or URLs), communication tools (e.g., chats, forums or wikis) and means to organize these elements (Drira et al., 2012).

The integration between learning scenarios platforms and LMSs can be done through two approaches. One is to embed a scenario player into the LMS. An example is the GRAIL project: an IMS-LD player (conform to the levels A and B of the IMS-LD specification) integrated into .LRN (De-la-Fuente-Valentín et al., 2007). The other approach is to keep two different environments: the learning authoring tool itself, where teachers design scripts; and the LMS, where students unfold them. An example is the CADMOS platform, conforming to the levels A and B of the IMS-LD specification, and which learning scenarios can be exported and operationalized via Moodle (Boloudakis, 2012).

In our work, we have investigated the second of these approaches from the mapping of ediT2 designs into Moodle through two strategies: using a middleware platform and a direct *ad hoc* mapping.

7.2 Middleware-based strategy

The purpose of middleware is to facilitate gluing together different systems by hiding internal details. The development of middleware facilitating the deployment of learning designs on different operationalization frameworks is a desirable evolution of learning technologies, well in line with the approach presented in this work.

In order to study if this approach is tractable, we have used the GLUE!-PS middleware (Prieto et al., 2011), which objective is to semi-automatically map, from ready-to-use adapters, different authoring learning tools into different virtual learning platforms (see Section 2.3.3).

Therefore, using GLUE!-PS to interoperate ediT2 and Moodle requires only implementing an adapter to translate the conceptual framework used in the table/tree representation into the GLUE!-PS data model concepts (the adapter responsible to map GLUE!-PS into Moodle is already ready). Such a mapping exists

for the Kobbe's et al. CSCL notions, and we therefore used this instantiation to successfully test this interoperability (experiments using this integration were performed at Valladolid, Spain; see Section 5.2.2, for details).

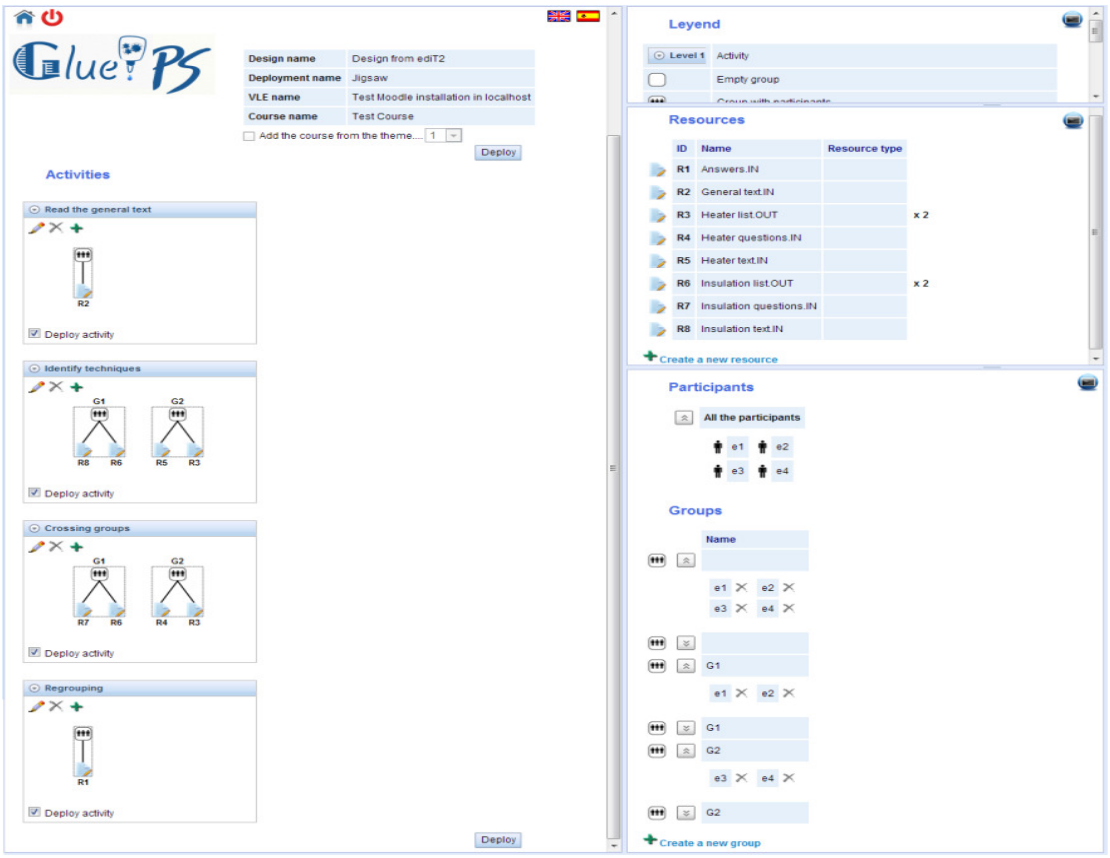


Figure 7.1 ediT2-GLUE!-PS mapping (from the jigsaw script presented in Figure 4.3)

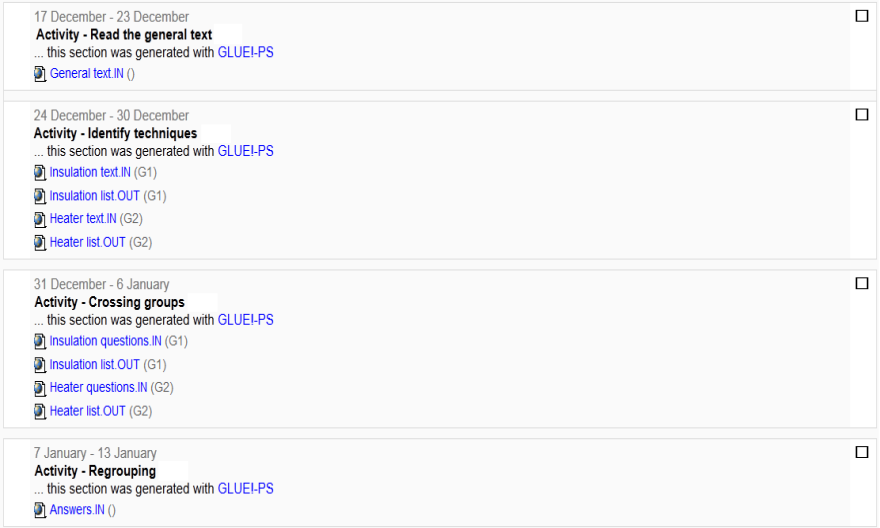


Figure 7.2 GLUE!-PS–Moodle mapping from the jigsaw script presented in Figure 4.3 (administrative (teacher) interface)

This approach allows a use-case such as:

- The students, related to a specific Moodle course, are imported from it thanks to the GLUE!-PS API (Application Programming Interface);
- The table editor is launched. The participants editing box is initialized with the labels of the students just imported. Each resource created by the teacher must mention the address web where it is hosted (in the "Description" field);
- The editor is used to represent/edit the script (the jigsaw script represented in Figure 4.3 is reused here as an possible example);
- The user presses the export to GLUE!-PS button, opens the GLUE!-PS editor, imports such file and selects Moodle as the target platform;
- The adaptor generates a GLUE!-PS representation of the scenario that can, if needed, be complemented or adapted by the user (Figure 7.1);
- The user presses the GLUE!-PS deployment button to that a second (ready-to-use) adaptor generates a backup file (.zip) to be imported in Moodle, through its course restoration process;
- Such file is imported and the course created (or adapted) is associated with the corresponding resources and participants;
- Finally, the students are grouped and offered resources according to the script as designed via ediT2 (Figure 7.2). This figure presents the Moodle interface when visualized by its administrator user (the teacher), who can access all activities to be accomplished to her/his students.

Group	Participant					Activity	Resource	
Class	s1	s2	s3	s4	s5	Introduction	Role play overview	Scenario
	s6	s7	s8	s9			Task structure	Role groups
Class	s1	s2	s3	s4	s5	Grouping for roles	Options	
	s6	s7	s8	s9				
Pros	s1	s4	s3	s2		Edition for Pro	Journal Pro	
						Q&A for Pro	Q&A Pro	
						Forum for Pro	Forum Pro	
Cons	s5	s9	s8	s7	s6	Edition for Con	Journal Con	
						Q&A for Con	Q&A Con	
						Forum for Con	Forum Con	
Class	s1	s2	s3	s4	s5	Conclusion	Forum - Everyone	Voting
	s6	s7	s8	s9			Journal	Final Q&A

Figure 7.3 A possible design of a simple role-play script from ediT2

Another example of the ediT2-Moodle operationalization was performed to model a simple role-play script (LAMS, 2013; see Section 5.4). A possible design from ediT2 is depicted in Figure 7.3 (for a hypothetical classroom with nine

students), and its deployment into Moodle, after manipulated from GLUE!-PS, is presented in Figure 7.4.

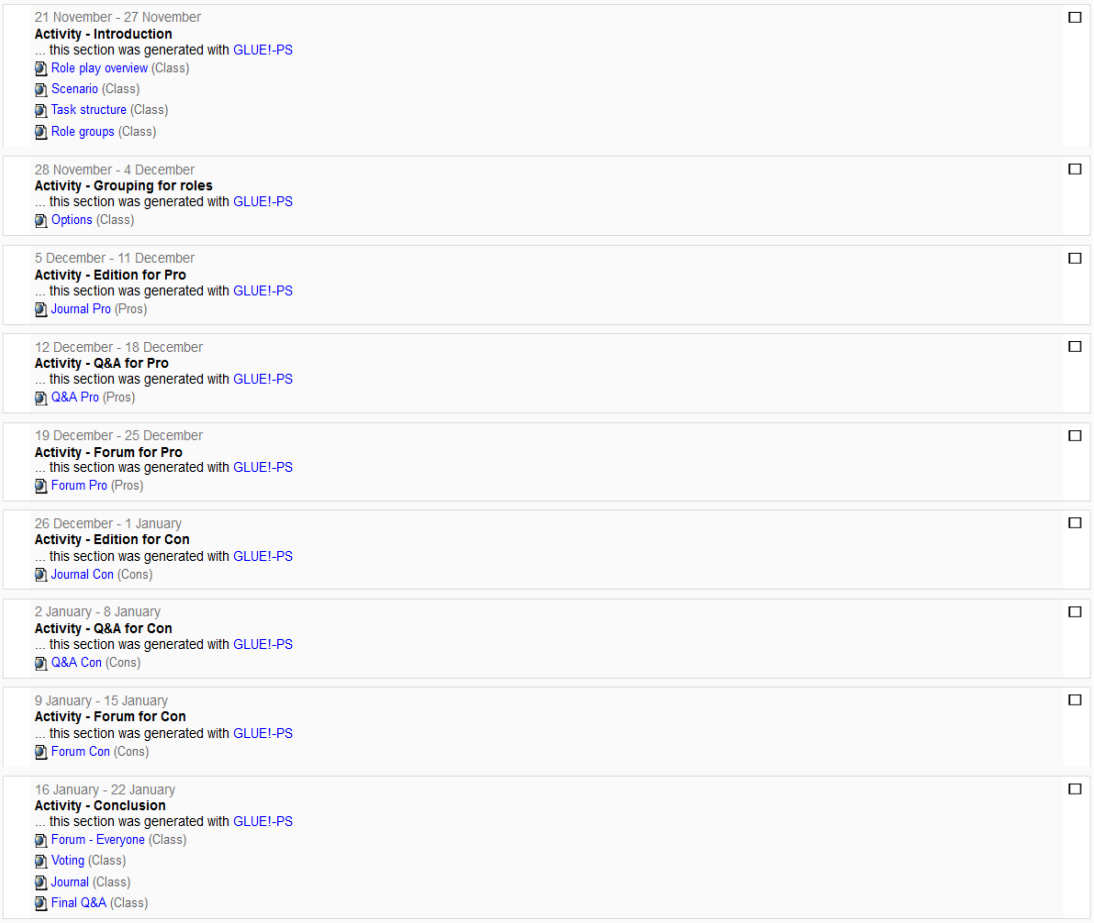


Figure 7.4 A simple role-play script deployed into Moodle via GLUE!-PS (from the script presented in Figure 7.3)

7.3 *Ad hoc* strategy

The *ad hoc* implementation strategy consists of implementing a specific piece of code that directly matches the table structure and the targeted framework (as developed to the jBPM integration described in Section 6.4).

Particularly for the interoperation discussed in this section, resources may be defined using the Moodle platform and after, imported to ediT2. The idea here is to take advantage of this LMS, which offers friendly editors to create on-line artifacts such as documentary-resources (e.g., files or quizzes) and collaborative means such as chats or forums.

The generation of a Moodle course from an ediT2 design was implemented by direct manipulations of the Moodle database. Activities as represented in ediT2

(i.e., a row or a sub-row) are transformed into a Moodle course (other options are possible). If there are n-ary relationships (i.e., the activity is spread over different groups), *n* Moodle courses are created, where *n* represents the number of leaves of the branch representing the script-component in the table.

We implemented a specific web-service since the Moodle API (at least for the version 2.2.3 we used) does not provide as external services the creation of resources or their association with courses. Such web service implements the functionalities to allow mapping the following Moodle resources: assignments, chats, folders, resources, wikis, IMS-CP objects and quizzes.

This deployment, in particular, considered simple text files as input entry, which were mapped into Moodle "resource" objects (other types of files could have been used). On the other hand, the artefacts produced by the students (output information) were uploaded into Moodle from the use of its "assignment" objects.

This makes possible a use-case such as:

- Using Moodle, the teacher defines the resources to be used to instantiate the script (if these resources do not exist yet);
- The editor is launched. The teacher clicks on "load resources" and references to the Moodle resources are uploaded into the editor's resource box. The same could be done for participants, as discussed in last section, but this functionality has not been developed in this strategy. Here, the technique used was different: we changed the attributes of ediT2 participants to meet the requirements of the integration (in this implementation, participants fields are defined by "user name", "first name", "surname", "email", "city" and "country", all required at Moodle);
- The teacher edits the script;
- Finally, from the ediT2 interface she/he conducts the session by deciding the activities that should be launched.

The Figure 7.5 presents an example of such an interoperation when considering only the first activity of the jigsaw script described in Figure 4.3 (the activity "Read the general text").

In this version, ediT2 was enhanced to associate each activity or sub-activity with means to take notes about it (possible observations about its items or about its unfolding); and a "play" button (the top of Figure 7.5). Hitting this button automatically creates a Moodle course implementing the activity, i.e., the activity is launched and the corresponding resources become available for the corresponding students, as defined by the script.

The middle of Figure 7.5 represents the initial Moodle interface for the student "e1" (when connected). Here, she/he is presented to the only one course which she/he is enrolled ("Read the general text"). Finally, she/he may access the course to know the Moodle activities to perform. Specifically in this example, "e1" must read the content of the document "General text" (the bottom of Figure 7.5).

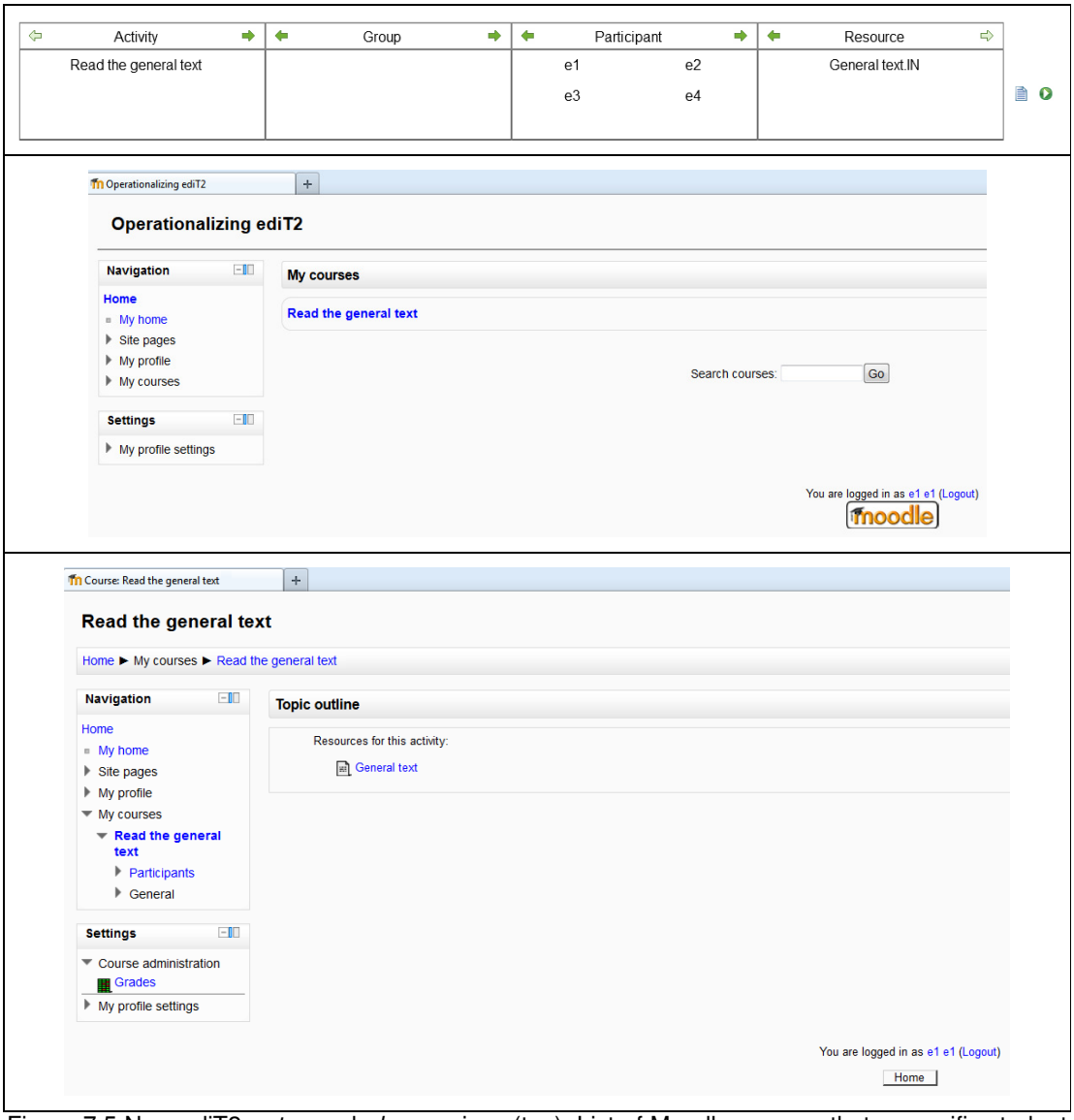


Figure 7.5 New ediT2 *note*s and *play* services (top). List of Moodle courses that a specific student ("e1") is enrolled (middle). List of resources of a course assigned to such student (bottom)

Another example of mapping between ediT2 and another delivery system will be discussed in details in the next chapter (Section 8.2).

7.4 Focus on technical representation transformations

From a general perspective, the interoperation between ediT2 and another system requires the development of a specific module to map ediT2 notions, and the script internal structure as defined by the tree and branches structure, into the model representation of the target platform.

This mapping is to be technically made easier if focusing on a given representation structure. However, this does not hold if offering representation flexibility, i.e., if users can adopt different script structures. This tension may be solved by developing one mapping, based on a canonical structure, and using an algorithm specially constructed to transform a script representation from one structure to another (see Figure 7.6) – this was the technique adopted in the experiment performed at the University of Valladolid, Spain (see Section 5.2.2, for details).

Algorithm: Changing the representation structure

- *Move a notion up/down*: Described in Figure 4.6
- *Insert a notion*: Create new empty leaves in a new level (l_i), and establish a unary relationship with the nodes in level l_{i-1}, when applied
- *Remove a notion*: Remove level l_i and update, when applied, the relationship between levels l_{i-1} and l_{i+1}

Figure 7.6 Transforming the representation from one script-structure to another

When transforming the table/tree from a structure S1 to a structure S2 involving the same set of notions, the process may lead to duplicate values. However, this is not an issue if S2 is only used as input for a piece of code implementing a mapping to another system (users do not access it). The same occurs when S2 introduces new notions (in this case, empty cells/nodes are created). If notions used in S1 are not used in S2 anymore, the representation is, of course, different. For example, this may occur if teachers use the "Role" notion to make some aspects of the script more precise but this notion has no equivalent in the enactment framework and instead is implemented as specific instructions to students.

The transformations that are made possible by the tree modeling may be used to normalize users' description and, for instance, change a user's description of a script into a canonical description. For instance, canonical representations may be defined with respect to criteria such as the tree balance or the number of nodes (e.g., reorganizing the tree to avoid duplications or odd constructions). From an end-user perspective, this would allow use-cases within which teachers describe the script and the system reorganizes the description and turns it into an equivalent though simple table. Although technically implementable, the advantage of such a service in teaching terms is questionable, and we have not explored this perspective in this work.

7.5 Discussion

These two implementation strategies have different characteristics, pros and cons.

The main advantage derived from the middleware approach is that only one mapping (from the script editor to the middleware) has to be implemented. Once this is done, the middleware enables deployment on different delivery platforms. A related benefit is that any technical modification of the enactment platform will only have implications for the corresponding GLUE!-PS-TargetPlatform adaptor.

As its drawbacks we could mention that: (1) for run-time flexibility and dynamic configuration, this approach is not suitable as adaptations require re-launching the overall process, and; (2) the mapping is generic (cannot be customized) and is only possible if the conceptual frameworks match. GLUE!-PS offers a fairly general conceptual framework and thus, unsurprisingly, it was easy to match it with a consensual framework such as the one proposed by Kobbe and his colleagues (this does not necessarily work when using a more specific conceptual framework).

With respect to the middleware strategy, the *ad hoc* implementation is more flexible. It allows the implementation of specific features needed to certain contexts as, for instance, directly linking representation and operationalization aspects to enhance the dynamism of scripts and/or to reduce the gap between learning design tools and enactment platforms, when these environments are represented by different environments (Neumann and Oberhuemer, 2009). This has been identified as a possible obstacle for teachers to follow the flow of information during the enactment step (Ronen et al., 2006). This type of issue have been diminished in proposals such as LAMS and CeLS, for instance, where these environments closely associate teacher authoring and learner implementation of activities on a specific enactment platform.

Ad hoc operationalization supports controlling the generation of activities on the enactment platform. Using the middleware GLUE!-PS approach, all activities are generated together from start; using an *ad hoc* implementation, one can decide when activities should be launched on Moodle, which means that synchronicity features can be implemented.

In the *ad hoc* strategy, Moodle courses are directly configured from the script description, avoiding the fact that teachers have to manage too many representations. Whereas in the middleware strategy the teacher needs three steps to achieve her/his script life-cycle ((1) design, with ediT2; (2) instantiation, with GLUE!-PS, and; (3) operationalization, with Moodle), in the *ad hoc* approach she/he needs only two of them ((1) and (3)). Such a reduction minimizes the possibility of information loss caused by human and/or technological factors intrinsically related to each mapping between the data models of the tools involved in each step of this process (Muñoz-Cristóbal et al., 2012).

As disadvantages about the *ad hoc* implementation, we could mention: first, this approach can lead to locally re-invent code that to some extent has already been produced by the middleware designers, which is always a pity. Second, the fact that evolutions of the LMS may need considerable update of the web service specially constructed to such finality, which is not a scalable strategy.

8 Editor generalization

8.1 Initial considerations

8.2 Developing a specific variant: the MediaWiki example

8.3 A generic editor

8.3.1 Making the editor generic

8.3.2 Example #1: using an ediT2 variant to represent preparation sheets

8.3.3 Example #2: using an ediT2 variant to represent SceDer scenarios

8.3.4 Example #3: introducing monitoring information in the scenario design

This chapter builds on and extends some ideas and concepts presented in our paper published in the Computers and Education journal (2015). Also, it represents part of the results achieved from a collaborative research developed with the GSIC group and, in particular, with the colleague María Jesús Rodríguez-Triana (University of Valladolid, Spain).

Dans les chapitres précédents, nous avons décrit l'instanciation du modèle T^2 pour la conception et le développement d'un éditeur de scénarios collaboratifs (l'éditeur ediT2). Cependant, les expérimentations et les études exploratoires concernant la façon dont les utilisateurs d'ediT2 conceptualisaient leurs scenarios nous ont fait comprendre que, souvent, ils recherchaient des adaptations locales et/ou des extensions des notions/attributs proposées et que, parfois, ces adaptations apparaissaient comme une nécessité pour l'interopération d'ediT2 avec d'autres plateformes de mise en œuvre.

Sur la base de ces remarques, nous avons considéré comment généraliser notre approche, selon deux stratégies différentes : d'une part, par le développement d'une variante d'ediT2 à travers l'adaptation directe de son code (ce qui nous a permis de confirmer que notre approche générale peut être facilement adaptée à des besoins locaux) et, d'autre part, en rendant l'éditeur générique (ce qui nous a permis de confirmer l'utilisabilité de l'éditeur, que la modélisation en table avait un intérêt général pour la représentation de scénarios, et la flexibilité informatique de notre approche).

Dans le chapitre précédent, Moodle a été utilisé comme plateforme de mise en œuvre des scénarios collaboratifs. Ici, nous allons considérer une autre possibilité technologique basée sur le Web, un wiki, pour étudier ces variations d'ediT2. La plateforme utilisée pour cette opérationnalisation a été MediaWiki, qui a été interfacée avec ediT2 grâce à un parseur spécialement construit pour analyser l'arbre d'un scénario et l'associer avec les propriétés MediaWiki correspondantes.

Pour la deuxième stratégie, nous avons modifié l'implémentation d'ediT2 pour modéliser ses notions à partir d'une liste d'attributs généraux et d'une liste d'attributs d'instance. Techniquement, ces adaptations sont simples car elles ne nécessitent que la modification de la structure des données des nœuds et de l'interface de l'éditeur (pour inclure des boîtes d'édition des instances) – les algorithmes de gestion générale restent inchangés. Avec cette version générique, nous avons pu créer trois variantes d'ediT2.

La première de ces adaptations a utilisé des notions et des attributs pris à partir de fiches de préparation de séances (« *Step* », « *Teacher's Activity* », « *Student's Activity* », « *Organization* » et « *Resource* ») qui ont été testées avec huit étudiants-enseignants à qui on a demandé de manipuler l'éditeur pour préparer des séances d'enseignement.

La deuxième est une variante créée pour permettre à des enseignants de conceptualiser des scénarios SceDer à partir de l'interface d'ediT2. La plateforme SceDer a été conçue pour la création de scénarios d'apprentissage collaboratif pour des séances un-à-un (chaque étudiant avec son ordinateur/portable) à partir de cinq notions conceptuelles (« *Provider* », « *What to do?* », « *Receiver* », « *Electronic Resource* » et « *Presentation Space* »). Avec ce travail, nous avons montré que le modèle table/arbre offre la même expressivité de SceDer.

Enfin, la troisième variante d'ediT2 créée grâce à sa version générique a été utilisée pour permettre à des enseignants d'informer des paramètres de monitoring lors de la conception de scénarios collaboratifs, à partir de la déclaration des notions/attributs suivants : la période du monitoring ; les activités à effectuer (l'échéance, les participants, le niveau social de l'interaction et le soutien technologique), et les ressources à utiliser (utilisation et le type d'usage).

8.1 Initial considerations

In the preceding chapters, we described the instantiation of the T^2 model to design and implement an authoring system using the notions presented in (Kobbe et al., 2007) to represent CSCL scripts (the ediT2 editor).

However, experiments and explorative studies concerning how ediT2 users conceptualized scenarios made us realize that, often, they looked forward to local adaptations and/or extensions of the proposed collaborative notions and/or attributes. For instance, in one of the experiments, a teacher asked for a proper "duration" notion. Such adaptations also appeared as a need for interoperation with other systems' frameworks, e.g., through the adaptation of the participant notion attributes in the *ad hoc* strategy for operationalization (Section 7.3).

Actually, we initially used the Kobbe et al. notions as a substratum, in order to target a large spectrum of CSCL scripts but, however, the T^2 model is independent of these notions, which can be adapted or extended.

Based on these remarks, we considered how to generalize the approach. Two strategies are possible. One is to go into the code and adapt the system. The other is to introduce some genericity by rendering editable the notions, and respective attributes, of the editor.

In this chapter, we present different research actions we conducted to study these aspects and reinforce different conclusions. First, we present the development of an editor variant by modifying the code. This allowed us to confirm that the overall approach allows easy adaptations to local needs. Then, we present how we rendered the editor generic, and different examples that allowed us to confirm usability, that the table representation had a general interest for representing scenarios, and the approach computational flexibility.

8.2 Developing a specific variant: the MediaWiki example

In Chapter 7 a LMS (Moodle) was used as a framework to operationalize CSCL scripts. Here, we will consider another web-based technological possibility, wiki. Wiki has produced interesting results in collaborative learning (e.g., Honegger and Notari, 2009; Larusson and Alterman, 2009), mainly due to the fact that it allows users without technical knowledge in Internet development to create and edit web pages, and interact on-line. The platform used to this operationalization is MediaWiki (2013), with strategy of implementation *ad hoc* (see Section 7.3, for details).

We considered the following type of scenario. The overall objective is to lead students to edit documents composed of a list of pages, each one corresponding to a topic. A topic page contains a description, a synthesis text (to be edited individually or collectively by the students), a list of sections, and some inputs in the form of URLs. Each section page contains a description, a text zone, and inputs, if any (other design choices are possible). Editing a scenario consists in listing the topics and sections that students should work on, defining the editors (i.e., the students in charge of editing the different pieces of text), and indicating the provided inputs.

Such scenarios may be represented by instantiating the T^2 model with the notions “topic”, “section”, “editor” (used twice to allow separate topic and section editors) and “input”. Figure 8.1 presents an illustrative example of wiki-based scenario using these notions in which students are invited to study the Brazil’s geography and history.

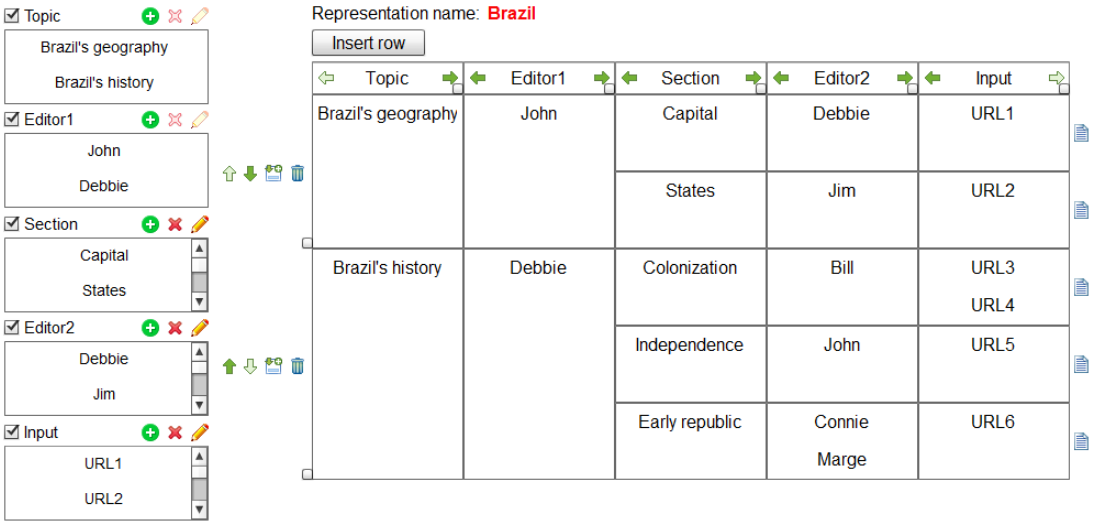


Figure 8.1 “Brazil” scenario example modeled from the ediT2-MediaWiki instantiation

This example allows illustrating the approach’s representation flexibility again. The “Topic-Editor-Section-Editor-Input” structure used in Figure 8.1 suggests that each topic has one or more editors and sections, and each section has one or more editors and inputs. However, one can also adopt other combinations. For instance, a “Topic-Section-Editor-Input” structure suggests that only sections are editable (no editors for topics); a “Topic-Editor-Input-Section-Editor” structure suggests that the provided input is common to the topic (rather than local to the sections). In fact, the table representation allows for a series of different meaningful constructions (e.g., considering sections or not, having different editors at the topic and section levels or not, associating inputs to topics or sections), where each of these corresponds to different wiki structures. In formal ways: ([Topic] [Editor]* [Input]* ([Section] [Editor])*) || ([Topic] [Editor]* ([Section] [Editor])* [Input]*).

To allow a straightforward operationalization of such scenarios on MediaWiki, a parser has been implemented. It analyzes the tree (i.e., the topics and

topics' sections) and calls the corresponding MediaWiki API feature (through its very convenient ready-to-use methods such as "edit/create a page", for instance). Figure 8.2 presents such an operationalization for the "Brazil" scenario example depicted in Figure 8.1.

Figure 8.2 The "Brazil" scenario (Figure 8.1) when deployed into MediaWiki

Managing the implementation by hand allows customizing the editor and the operationalization process. For instance, ediT2 was enhanced by a piece of code that uploads data (here, students' name) from the MediaWiki database. The operationalization respects the defined roles: the editors, as defined by the table, can only edit the wiki-sections "Summary" (in "Topic" pages) and "Text" (in "Section" pages) in which they were designed to. For instance, taking as example the "Brazil's history" Topic page, we can observe in Figure 8.3 that "Debbie" has the rights to edit this page, but not "John" (as expected, in accordance with the script representation in Figure 8.1).

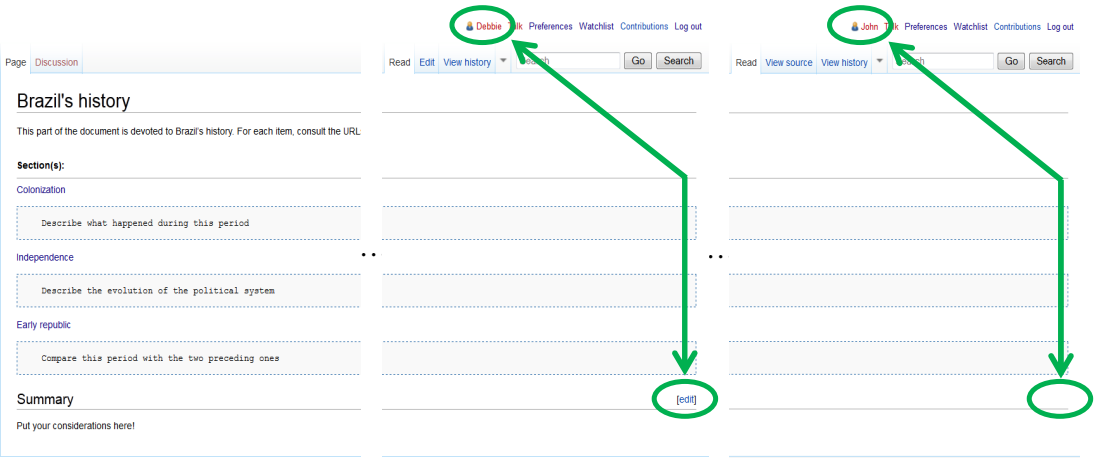


Figure 8.3 User rights in the edition of wiki pages ("Debie" and "John" accessing the "Brazil's history" Topic)

This example also allows illustrating instantiation support. We developed specifically for this ediT2 version an instantiation algorithm concerning group formation and task distribution mechanisms for wiki scenarios. An example of pattern is: given a list of *n* students, each one should be editor for one topic, this topic being composed of *m* sections individually edited by one other student. The instantiation process is simple for a limited set of students and sections (e.g., if *n*=3 and *m*=2), but may become a nightmare in effective settings if *n*=21 and *m*=4.

Figure 8.4 presents a possible implementation of such algorithm (top), and the generated tree when *n*=5 and *m*=3 (bottom). Many variants of this pattern may be considered such as working from a list of topics and sections, introducing a parameter to state the number of editors per section (which was introduced above as one and only one), or managing the distribution of inputs.

Algorithm: Generating a topics/sections structure
LS (list of students); *nst* (number of sections per topic)

Let *n* be the number of students

1. Create L1 = list of topics (a topic by editor; L1 length = *n*)
2. Create L2 = list of topics' editors (L2 = *LS*)
3. Create L3 = list of *n* sub-lists of *nst* sections each
4. Create L4 = list of *n* sub-lists of *nst* editors (considering *LS* as a circular list, distribute student as editors over L3 sections; when the candidate is the topic's editor, she/he is moved to *LS* queue)

Going through the L1, L2, L3 and L4 lists (recursively), construct the tree branches, i.e., associate a topic (L1), a topic editor (L2), the sections (L3) and the sections editors (L4)

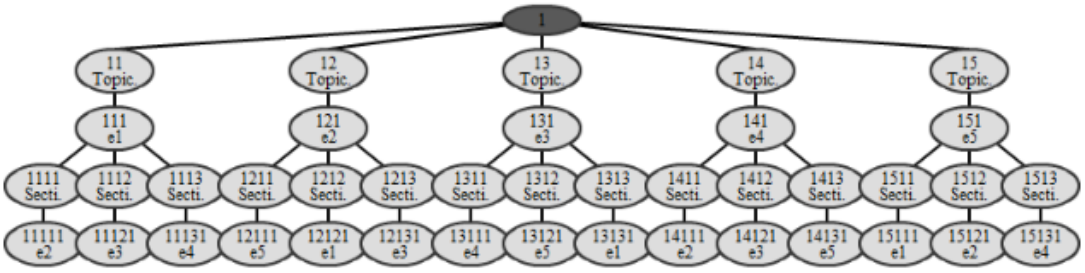


Figure 8.4 Instantiation mechanism for a wiki-based pattern (top) and generated tree, when *n*=5 and *m*=3 (bottom)

8.3 A generic editor

8.3.1 Making the editor generic

We modified ediT2 implementation to model notions as a list of general attributes, from which a name; and a list of instance attributes. These attributes can be a text field or a choice list. The list and structure of the notions are set when launching the editor. Technically, this required adapting the node's data-structure and the editor interface (creating instance-editing boxes).

Albeit there is no theoretical limit concerning the number of notions to be used, we continued to use five notions to understand that too many columns in the table would probably lead to reduce the simplicity of the ediT2 interface (this could easily be defined as a parameter, however).

Figure 8.5 presents the interface for defining notions' attributes. As an example, a "Task" notion is modeled from "Type" (Individual/Collective) and "Duration" (time in minutes) as its general attributes, and "Beginning time" and "Ending time" as its instance attributes (representing when student(s) allocated to perform such a task should start and finish it, respectively).

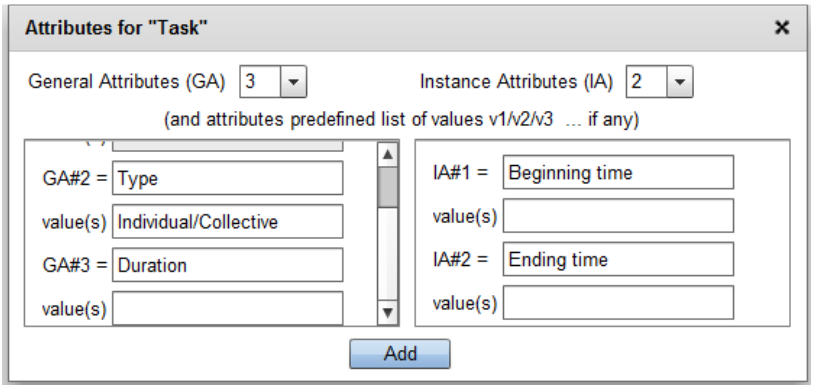


Figure 8.5 Defining the general and instance attributes for the "Task" notion

Figure 8.6 presents the functioning of this reengineered editor. When a "Task" item is dragged-and-dropped into a table cell, an editing box is opened to that its specific information (instance attributes) can be edited (instance attributes are the only editable fields in this operation). This editing box is also showed when the teacher double-clicks on an item in a cell.

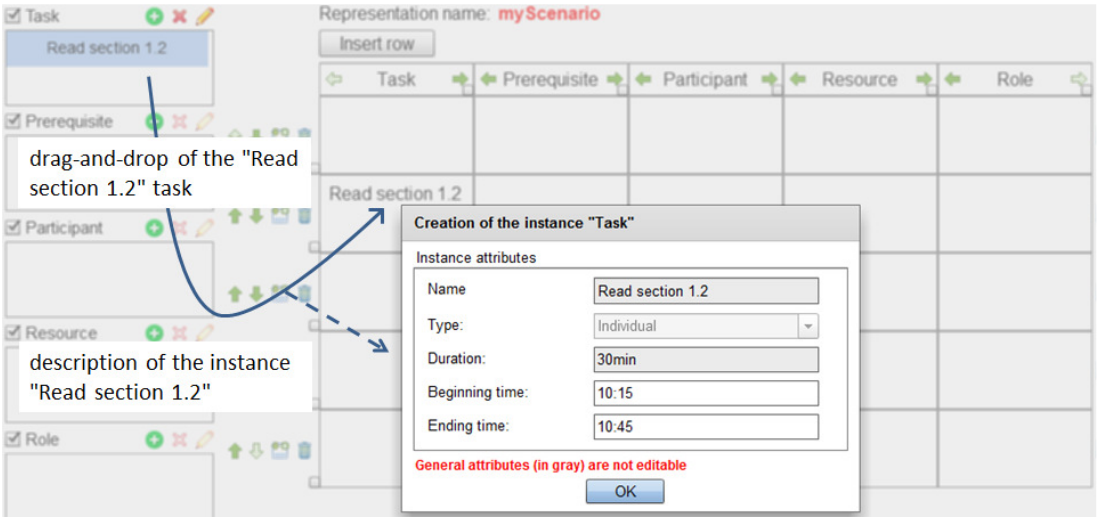


Figure 8.6 Learning scenario designed with general and instance lists of attributes

The way we rendered the editor generic is aligned with the overall approach: simplicity. The adopted data structure acts as a basic meta-model. An implementation with general and instance attributes is fairly general but does not support every construction. Other choices, however, can be made. Such adaptations are simple because they require modifying the nodes data structure only. The general management algorithms remain unchanged.

A natural step forward opened by the ediT2 generic version is to offer editor variants, i.e., several sets of predefined notions and respective attributes. Technically, variants can be offered as empty scenarios, i.e., with no rows yet.

A variant may be associated with predefined list of items (e.g., predefined list of activities); if/then specific constraints related to the used notions (e.g., "If the activity is [of] *Individual* [type], then do not allow several participants"); and/or controls specific to methodological considerations or particular concerns. For instance, one may offer a predefined list of activities associated with detailed information such as difficulty or requested level of engagement, and introduce controls helping teachers to take care of these aspects, e.g., to avoid a particular series of activities. This goes into the direction of the utilization of predefined pedagogical patterns (Hernández-Leo et al., 2006, 2010).

8.3.2 Example #1: using an ediT2 variant to represent preparation sheets

We used the generic version of ediT2 to create an instance with the notions and attributes taken from a standard classroom preparation sheet (Figure 8.7). It was presented to eight student teachers in pre-service education, i.e., spending part of the year as students and part of the year teaching. They were asked to manipulate the system as a possible means for preparing their classroom sessions.

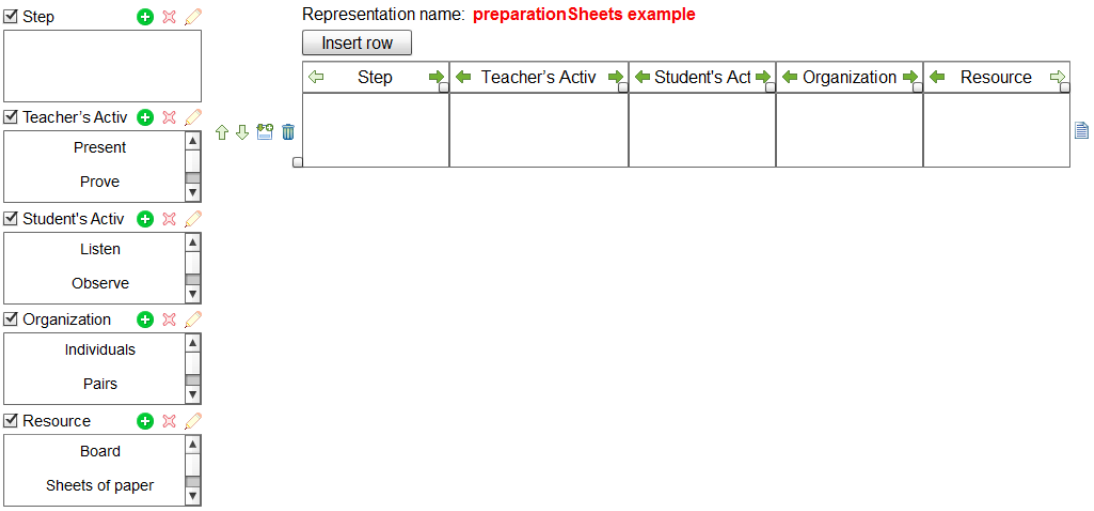


Figure 8.7 "Classroom preparation sheet" scenario

This mini-experiment allowed confirming the usability aspects. All student teachers agreed the editor was easy to use. With respect to usefulness, responses differed. Some of the pre-service students did not find any particular additional usefulness compared to using a basic word editor. Conversely, others perceived the table as a very pertinent way to deal with both a synthetic presentation (the table with the items, easy to reflect on and edit via the drag-and-drop function) and detailed presentation (the item's attributes and open text description, accessible by double-clicking on the item (cf. Figure 8.6)).

8.3.3 Example #2: using an ediT2 variant to represent SceDer scenarios

SceDer Authoring is a visual LSE designed to create learning scenarios for one-to-one classrooms (one student per computational setting; Niramitranon, 2009). It represents such scenarios from tables composed by simple rows (steps) with cells respecting a 1-1 relationship. Its framework introduces five conceptual notions ("Provider", "What to do?", "Receiver", "Electronic Resource" and "Presentation Space") and predefined values for them (e.g., "Teacher", "Students" or "Groups" for "Provider" and "Receiver" notions; "Question?", "Answer" or "Discussion" for "What to do?" notion).

The SceDer output is a COML Package which consists of an interchangeable COML Document (an XML file based on a language created specifically to this project), and resources allocated to the scenario's unfolding (e.g., image files). This package is used to delivery SceDer Authoring designs on the Group Scribbles collaborative learning tool (Group Scribbles, 2013).

As discussed in Section 5.4, the SceDer Authoring tool is capable of representing 13 high-priority educational scenarios, and this feature guided this research action from the creation of an ediT2 variant able to represent SceDer designs, as a way to reinforce the argument that our approach allows representing a wide range of scenarios.

With this purpose, we showed that the table/tree model offered the same SceDer Authoring expressiveness by implementing an adaptation that offers the same SceDer notions and predefined items, and exports ediT2 scenarios' representations to COML Document files. We represented some of the scenarios mentioned in the Niramitranon et al.'s work (e.g., the "Chem Drawing" scenario, Figure 8.8) and checked that the XML files produced complied with COML (Niramitranon et al., 2010).

The ediT2-SceDer editor was conceived from the ediT2 generic version (to the declaration of the SceDer notions and attributes required). Additionally, we developed a module responsible to map the scenario machine-representation (the tree) into the XML format complying with the COML Document. The Figure 8.9 depicts a possible modeling of the "Chem Drawing" scenario (Figure 8.8) in such editor.

In Figure 8.9, the "What to do?" item dropped in the ediT2 table (deep gray selected cell) is represented by its name ("Question?") and its list of instance

attributes (two attributes: "What to do" (with "Work out Nitrous oxide" as text typed); and "Description" (not yet typed)). In this example, the list of general attributes was not initially declared by the teacher and, consequently, it is not showed in this edition box.

In a chemical class, a teacher asks the whole class of students verbally "Please draw the molecular formula and electron dot of Nitrous oxide, Carbon dioxide, and Carbon monoxide". Each student answers the teacher by drawing a molecular form (N_2O, CO_2, CO) and electron dot (chemical representation form of electrons) on their personal computing devices and then submitting back to the teacher. The teacher groups answers both right and wrong, for each chemical substance. The teacher then divides students into three groups, passes each group the previous work (a group of right and wrong answers in the same formula), and asks them to choose or reproduce the correct chemical formula and electron dot of that substance. Group 1 works out Nitrous oxide. Group 2 works out Carbon dioxide and Group 3 works out Carbon monoxide. In each group, members discuss to choose or reproduce the joint answer and then send it back to the teacher. The teacher then reviews the work of each group and shows a prepared image of the correct molecular formula and electron dot to the whole class.

Figure 8.8 The "Chem Drawing" scenario (extract copied *in extenso* from Niramitranon et al., 2010)

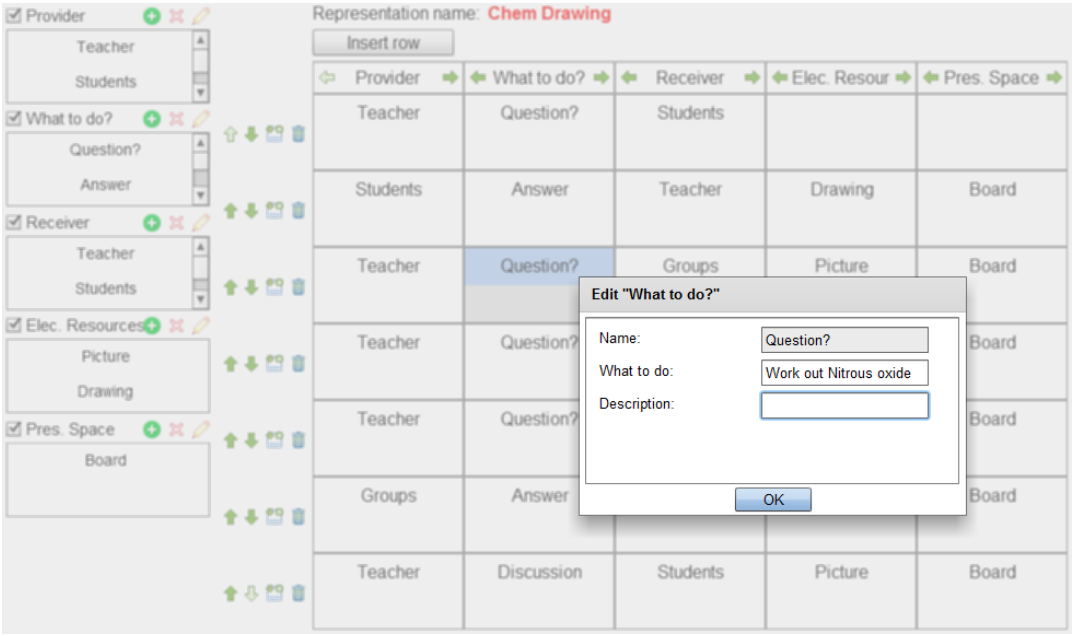


Figure 8.9 The "Chem Drawing" scenario (Figure 8.8) designed from the ediT2-SceDer tool

8.3.4 Example #3: introducing monitoring information in the scenario design

The first CSCL scripts were configured before their applications in real classrooms (Dillenbourg, 2002). Their monitoring techniques were generally based on *a posteriori* analyses of the data of students interaction collected during (and/or after) sessions. This presented some issues with unforeseen situations that could take place during scripts' unfolding (e.g., desynchronized teams, or students in a same group presenting relationship issues).

This motivated learning system designers to consider assistance functionalities during scripts execution, e.g., offering to teachers the enactment attendance. This included the possibility to adapt the learning designs on-the-fly to answer to unexpected situations unforeseen during the conception of pedagogical scenarios (Malzahn et al., 2008; Rodríguez-Triana et al., 2014; Soller et al., 2005).

Analysis of the script enactment may be supported by software modules that can more or less understand the current state of the script's unfolding and compare it with a predefined "ideal" specification. This is a basis to allow teachers to decide whether an adaptation would be required through operations such as subdividing a complex activity into simple ones or reallocating students to different tasks.

An interesting example of such techniques is the Script-Aware Monitoring Process (Rodríguez-Triana et al., 2014), developed by the GSIC team of the University of Valladolid (Spain), and which objective is allowing teachers to inform monitoring parameters since the design of CSCL scenarios based on collaborative patterns.

Following this approach, learning designers / teachers should provide into the initial script's conception information related to:

- The period of monitoring.
- The collaborative pattern(s) to be used.
- The activities: deadlines; people (participants/groups); if/when they should be monitored; social level of interaction (individual/group/class); participation level (optional/mandatory); interactivity (face-to-face/ through computers/blended); presentiality (face-to-face/remote/ blended); and technological support (total/partial/without support).
- The resources/tools to be used in the activities: if they should be monitored (and respective action, when applied); type of usage (individual/group) and utilization (optional/mandatory).
- Additional constraints specified by the teacher to complement the intrinsic constraints related to the chosen pattern, if any (e.g., some intrinsic jigsaw constraints have been presented in Table 6.1). They can be complemented by an additional constraint such as: "each individual activity must be composed by only one student".

Since this monitoring information could not be edited in their standard editor, the Spanish group was interested in using ediT2.

Adapting the ediT2 basic CSCL instantiation to include representation of the monitoring data required different actions: (1) Defining the items' data structure to include the requested information. (2) Allowing users to edit this data when building a design. (3) Implementing if/then controls and hints, i.e., pieces of code that are executed when the teacher edits the corresponding data and provides methodological input. (4) Modifying the "save" feature to export the design (the scenario) in the format requested by the monitoring device.

These changes were processed by a member of the GSIC group ("the adapter") as someone who had not participated in the editor design or implementation in any way.

The adapter managed aspects (1) and (2) using the ediT2 generic version from the definition of an empty design featuring the different attributes that were required to represent the monitoring information. This was straightforward (~1 hour).

To implement the controls (3), it was given to the adapter the editor code and a short presentation about it (~1 hour). She considered 10 controls/hints, for example: "If the type of resource is *A*, then the actions that can be monitored are *B*"; "If participation in the activity is mandatory and there are not enough actions that can be monitored to ensure student participation, then report the lack of data." Such controls require ~ 1 to 2 hours each. In a second version, she decided to modify again the code to check all controls at the same time after the editing, which was here again basic.

Editing the export format (4) took ~2 hours.

An important output of this research action was to provide evidence of the approach computational flexibility (to be discussed in Section 9.1.1).

9 Discussion, conclusions and perspectives

9.1 Discussion
9.1.1 Originality
9.1.2 Application scope
9.1.3 Relation with other works

9.2 Conclusions

9.3 Perspectives and exploratory work
9.3.1 General perspectives
9.3.2 Monitoring perspective: exploratory work

This chapter builds on and extends some ideas and concepts presented in our paper published in the Computers and Education journal (2015).

Dans ce chapitre nous proposons une discussion de l'originalité de ce travail, de son champ d'application et de sa relation avec d'autres approches existantes dans ce domaine de recherche. Nous concluons avec quelques perspectives générales et exploratoires pour des améliorations futures.

Le raisonnement sous-jacent au travail effectué dans cette recherche est qu'il est intéressant, dans le domaine de l'enseignement assisté par la technologie, d'examiner comment adapter des outils aux enseignants et à leurs contextes. Cette réflexion nous a amené à deux autres aspects originaux de ce travail. Le premier est d'étudier un type de représentation qui présente des caractéristiques intéressantes pour son adoption (simplicité et flexibilité) et peut, par ailleurs, être étendu avec des services supplémentaires. Le second est d'étudier, en plus du fait de proposer un éditeur de scénario directement utilisable, le fait de permettre une adaptation informatique facile de l'éditeur.

En termes de champs d'application et de limites, la représentation table/arbre est basée sur une sémantique structurelle bien adaptée à des modèles simples. La table présente une relation « est-associé-avec », et des aspects sémantiques supplémentaires peuvent être développés comme des services de « respect de contraintes ». Cette approche est plus limitée que d'autres, par exemple celles basées sur la spécification IMS-LD et, en particulier, n'est pas adaptée à des représentations nécessitant un grand nombre de notions, des séquencements complexes ou des documentations importantes. Le modèle table/arbre doit donc être considéré comme une alternative présentant des caractéristiques différentes des approches courantes, à utiliser si et lorsque nécessaire.

Lorsqu'on compare ediT2 et d'autres approches, les éléments suivants peuvent être notés :

- En ce qui concerne la représentation, plusieurs éditeurs de scénarios d'apprentissage proposent à l'utilisateur des interfaces basées sur des graphes. Dans ce travail, nous avons montré comment une représentation en table peut être complétée par une représentation en graphe, ce qui présente un intérêt lorsqu'on essaie de bénéficier, à la fois, de la simplicité des tables et de l'expressivité des graphes.
- En ce qui concerne l'opérationnalisation, l'approche basée sur un modèle-pivot que nous avons proposée permet de se concentrer sur des moyens de représentation et d'édition proposés à l'utilisateur, l'opérationnalisation étant considérée via le déploiement sur une plateforme tierce de mise en œuvre.
- En ce qui concerne les caractéristiques de l'interface utilisateur, la manipulation d'une table est, bien sûr, beaucoup plus simple que celle d'un modèle multi-niveau. Ceci est en contraste avec des interfaces qui doivent se conformer à des spécifications complexes.

- D'un point de vue technique, une alternative à un modèle en arbre pourrait être l'utilisation de graphes orientés acycliques. Nous avons opté pour l'utilisation d'arbres pour rester cohérents avec les principes de simplicité de l'approche.

Nous concluons ce chapitre en présentant quelques perspectives générales et exploratoires de cette recherche.

Comme perspectives générales, nous présentons de possibles travaux futurs : l'évaluation de la flexibilité et de l'utilisabilité d'ediT2 lors de l'introduction de services avancés ; l'évaluation des pertes/modifications sémantiques possibles lors de la mise en œuvre de scénarios pédagogiques sur des plateformes tierces d'opérationnalisation ; et l'étude des questions de soutien méthodologique dans l'éditeur, par exemple, en poussant plus loin les travaux sur la possibilité d'offrir aux utilisateurs des patrons/gabarits pédagogiques prédéfinis.

Finalement, comme une perspective plus précise, pour laquelle quelques éléments exploratoires sont présentés, il sera intéressant d'étudier si l'approche est utile à des fins de surveillance et d'orchestration, avec des questionnements, comme : « Quelles sont les informations dont les enseignants ont besoin ? » ; « Quelles sont les données que l'environnement de mise en œuvre peut fournir ? » ; et « Comment/quand l'enseignant peut-il déplacer un étudiant engagé dans une tâche en cours de route ? ».

9.1 Discussion

9.1.1 Originality

The rationale underlying the work conducted in this thesis is that it is of interest, in the TEL domain, to examine how to adapt tools to teachers and contexts (Tchounikine, 2011).

Taking into account LSEs, one may consider that these tools should reify good instructional practices, that teachers should use them according to their specifications and be trained for this. However, this is a theoretical solution, and in many cases not practical. Moreover, this approach may also be questioned. One may also consider that teachers, as the actors in charge of the actual settings, and as actors having developed a professional practice, should be able to use tools in line with their practices and perspectives.

This line of thinking let to two other original aspects of this work.

The first one is to study a LSE which is not designed to implement an *a priori* EML. We have focused on a type of representation that presents interesting characteristics (simplicity, flexibility and adoption) and how to address representation and additional services. Building on the governing decision to use table interfaces may thus be seen as a techno-centric approach, but this is not the case. From a historical perspective, the decision to explore this approach is based on the fact that, with respect to users such as teachers, designing tools that provide educational value does not guarantee that these tools will be adopted (Tchounikine, 2011). Criteria for adoption include user-friendliness and easy transition from existing practices and tools. This is the rationale for considering devices that teachers know and use (tables). It aligns with the “design for all” guiding principle: representation and manipulation means must be simple and intuitive, avoid complex constructions and offer high flexibility (Dillenbourg and Jermann, 2010).

The second one is to study the design of directly usable LSEs but, also, customization by going into the code. One may argue that editors should be usable off the shelf. This suggests offering generic architecture/languages or using full fledge meta-modelling means allowing to generate editors with no hand-coding at all. Generic architecture/languages are useful, and many already exists (e.g., IMS-LD). Albeit generic means are powerful, teachers may not want to reflect in generic terms, despite being able to. Not misunderstanding the power of full-fledge meta-modelling (Drira et al., 2012; Laforcade and Choquet, 2006), these techniques are however difficult to implement. The approach we studied is more in line with the idea to allow local teaching institutions or communities of practice to customize systems to integrate them in their local technical ecosystems.

This adaptation aspect makes sense if the system is computationally flexible, i.e., can be adapted at low engineering cost. Chapters 6 to 8 showed that

the table/tree model offers such flexibility. The interest of a pivotal-model approach is to natively consider adaptation and extensions.

In our implementation, a tree is adopted because it is a classic and simple data structure, to which manipulations (generation, transformation and analysis) can be implemented by simple and uniform (and thus reusable) algorithms. Basic table representation features (e.g., displacing items between cells or creating rows) correspond to trivial algorithms (displacing items between nodes or creating branches). Features such as splitting/merging nodes (i.e., creating/merging sub-branches) or displacing a column while maintaining consistency between the table representation and the tree model require the propagation of modifications via tree manipulation algorithms, which are less trivial but of standard practice. These aspects form the architectural foundation; they do not need to be adapted.

Creating a particular table/tree instantiation is straightforward if one is using the generic version of the editor. A specific instantiation such as the MediaWiki version required adapting the data structure to introduce the "topic" (etc.) notions, implementing the load-students-from-MediaWiki feature, implementing the operationalization service (configuration of the MediaWiki wiki from the scenario) and adapting the table's top-level menu (access to additional services). This took ~4 days of engineering work, half of which was needed for mapping. Also, the collaboration with the GSIC group showed that adaptation by an external actor was easy.

9.1.2 Application scope

In terms of application scope and limits, the table/tree representation is based on a structural semantics which is well adapted to simple models (which is the case of many scripts). The table introduces an is-associated-with relationship, taking more precise interpretations according to the involved notions (e.g., Activity is-carried-out-by Participant(s); Section is-edited-by Editor(s); etc.). Additional semantics may be implemented as "respect of constraints" services.

This is more limited than general languages/platforms such as the ones based on the IMS-LD specification (e.g., TELOS; Paquette, 2010) and, in particular, is not adapted to representations requiring a large number of notions (e.g., ReCourse (2013)) and/or complex inter-relations (e.g., MoCoLADe; Harrer et al., 2007). The table/tree model is thus to be considered as an alternative, presenting characteristics different from usual approaches, to be used if and when relevant.

With respect to define LSEs that meet the needs and requests of a group of users (and, in some sense, come close to a participative design process), a possible use of the ediT2 generic version (Chapter 8) is to explore local instantiations with informant practitioners. For instance, in an explorative study in which different teachers were asked to identify the notions and attributes they wanted to use, one of the teachers introduced notions such as teacher's role (e.g., "give instructions", "be a resource" or "be an animator"), student's role (e.g., "receive", "search" or "present"), resources, activity duration and objectives (e.g., "introduce" or "institutionalize"). For such a conceptual framework, a flat table representation

appeared to be an option. In contrast, the editor seemed poorly adapted to teachers' attempts to represent teaching sequences (i.e., approaching a course's type of learning design) for which they introduced notions such as "session", "objectives", "prerequisites", "transition" and "pedagogical organization". A flat table representation is not adapted to characteristics requiring rich documentation.

9.1.3 Relation with other works

If one refers to the classification framework proposed in (Botturi et al., 2006), ediT2 addresses a conceptual level of elaboration (a general, aggregate view on the design, indicating its rationale and main elements) and builds on the following principles: offering a flat stratification (entities of all types are collected into a single representation) and a single perspective (one view on the entities) via a visual notation system (the table).

With respect to representation, many LSEs present end-users with graph-based interfaces (Botturi et al., 2008), in particular when considering complex sequencing, and build on the top of workflow engines (e.g., Flex-eL (Sadiq et al., 2002), COW (Vantroys and Peter, 2003), (Haake and Pfister, 2007), MoCoLADe (Harrer et al., 2007), LeadFlow4LD (Palomino-Ramírez et al., 2008)). LSEs related to XML-based representations (e.g., LD) usually build on according hierarchical representations.

Table and graph representations are different options, with different characteristics. We have shown how a table representation may be complemented by a graph-representation. This makes sense when attempting to benefit from both table simplicity and graph expressiveness. Users requiring complex representations and/or at ease with graph-representation should use editors natively based on graph representations such as MoCoLADe, and benefit from the specific associated features.

One may argue that graph-based LSEs can be used to represent both basic and complex scenarios. This is true but, with respect to adoption and smooth transitions, presenting practitioners with complex interfaces for the sake of providing features that may be used only rarely is exactly what should be avoided. More (comprehensive languages, all-in-one architecture offering a full range of services) is not necessarily better as it often comes with complexity and reduced flexibility. The approach we proposed allows designing simple devices, to be offered together with more comprehensive (and, for some users, more complex) ones.

With respect to operationalization, the pivotal-model approach suggests focusing on means to represent and edit scenarios, with operationalization being addressed via deployment on an enactment platform. One important advantage is allowing consideration of different platforms. This is also the rationale for the GLUE!-PS middleware and other proposals such as S-COL (Wecker et al. 2010). This approach is to be contrasted with design languages directly linked to enactment platforms such as LAMS (LAMS, 2012) or CeLS (Ronen et al., 2006).

The advantage of systems directly linked to technical platforms is that scenarios are by definition directly operationalized. A disadvantage, however, is that these languages/LSEs have to comply with computational constraints that hinder flexibility (e.g., in LAMS, teachers are forced to use the tools pre-defined by the platform).

With respect to operationalization and flexibility, the *ad hoc* implementation strategy is more flexible then using middleware, and allows specific features to be implemented. For example, in the specific MediaWiki implementation, we were able not only to choose the structure we wanted to use but also, to restrict editing to the students identified as editors. Other examples of attempts to limit flexibility issues are presented in works exploring learning systems related to a set of pre-defined scripts and for which some parameters can be adapted by teachers (Dillenbourg et al., 2009), or platforms within which new tools can dynamically be included during the script enactment (Belgiorno et al., 2008; Chiara et al., 2007). Also, some frameworks have pointed out to the use of pre-defined software components to the construction of dedicated CSCL scripts (Stegmann et al., 2009). As additional conceptual efforts targeting flexibility, we can mention the configuration of pedagogical patterns (Hernández-Leo et al., 2006, 2010) or the anticipation of adaptation patterns (Karakostas and Demetriadis, 2009).

Although the general perspective underlying this work has some similarities with the approach underlying Collage (Hernández-Leo et al., 2006) and related works (supporting teachers, considering flexibility, offering pattern instantiation and constraints-checking mechanisms), one important difference relates to the interface design and the underlying methodological concerns. In the Collage approach, the main idea is to offer teachers a pattern-based approach, this idea underlying the overall process and environment. In direct contrast, the ediT2 entry point is a generic table representation. We could think in improve the ediT2 editor with a list of predefined scenarios patterns, but this technique would have a similarity with the approaches developed in Collage and LDSE (Laurillard et al., 2013). The fact remains that these approaches are anchored in methodological and integrated pedagogical life-cycle perspectives, whereas the work presented here is more on the scenario edition level.

If one considers the end-user interface, visualizing a table is of course much simpler than a multi-layer model. By construction, the proposed approach suggests a flat interface, which may be enhanced, for instance, to introduce a menu item listing patterns and, for each of them, asking for the configuration data (as we have presented in Section 6.2). This is in direct contrast with interfaces that must comply with complex information (e.g., ReCourse or the LDSE interfaces).

With respect to software engineering, our use of a pivotal-model may be seen as a particular case of meta-modeling. In the Learning Design field, some works have explored using meta-modelling, e.g., (Laforcade and Choquet, 2006) or (Drira et al., 2012). Full-fledged meta-modeling, however, is technically difficult and often too complex to use.

From a technical perspective, with respect to a table representation, an alternative to a tree model could be Directed Acyclic Graphs (DAG). DAGs would

allow table constructions that, with a tree model, require some duplication. We opted for trees to stick to the simplicity principles of the approach. The way we use the tree model to implement different versions of LSEs is similar to the way MoCoLADe is based on a generic graph-based modeler.

A complementary analysis of our work with respect to the analysis grid proposed in (Tchounikine, 2011) is presented in Appendix E.

9.2 Conclusions

The fact that teachers use/adopt modeling tools is related to an array of aspects such as teachers' technical skills and professional practices, institutional aspects or schools equipment (Larnaca, 2012; Mor and Craft, 2012). One of these reasons is related to LSEs as computer-based systems. LSEs to be used by teachers must be simple, intuitive and flexible (Dillenbourg and Jermann, 2010).

The research question we considered in this work is: can one design and implement representation means that (1) are sufficiently simple and flexible to allow users (teachers) with basic ICT skills and no particular methodology training to edit scripts and adapt them to their view and context, and; (2) may be extended to match other needs and implement advanced features offered by other current classical languages/platforms such as simulation, checking constraints, supporting complex grouping mechanisms and scheduling structures, configuring the students enactment framework or monitoring the script's unfolding?

Our results show that offering a table representation coupled with a tree pivotal-model allows designing easy-to-use and adaptable LSEs. Within its scope of application, it can be used to offer users who are not specifically trained (e.g., teachers) an editor that is easy to use, adaptable to needs and/or practices, not more complex than needed and easy to interoperate with other (local or not local) systems if useful.

To study this approach, four objectives / evaluation criteria were investigated:

- Pedagogical expressiveness: "can table-based editors represent a wide range of scenarios?"
- Usability: "do teachers find the editor easy to use and intuitive?"
- Computational expressiveness: "does the approach allow implementation of complex services?"
- Computational flexibility: "is the editor easy to adapt to local needs?"

The pedagogical expressiveness of the table representation was tested by studying how the editor allowed representing different types of scenarios. Easiness-

of-use was tested via usability tests. Computational expressiveness and computational flexibility were demonstrated by proofs-of-concept developments.

9.3 Perspectives and exploratory work

9.3.1 General perspectives

This work opens different perspectives, which we would like to outline here.

We have shown the usability of table-based representation and, then, how such a representation may be enhanced. This raises the question of evaluating the extent to which the editor's usability is preserved when introducing add-ons. This requires setting-specific studies.

Within our perspective, the strategy should not be to integrate in the basic editor all the advanced services that can be thought of but, rather, within a specific context, and working with the users to preserve usability, introducing the services they identify as useful. Such a strategy will make it possible to study the extent to which the ease-of-use and flexibility features, coupled with other services as needed, influence adoption and practices.

As usual when considering editors, we did not consider the quality of the learning designs produced by teachers during the tests. The reason is that good and bad designs may be produced using any editor. Nevertheless, considering the quality of the produced learning designs in the sense of the respect with their rationale, and/or their fit to a considered setting as, for instance, an enactment platform integration (e.g., considering if it exists loss of meaning during the mapping between ediT2 and such environment) remains an important question. At this level, studying the effect of specific support such as instantiation mechanisms or checking constraints is also on the agenda.

In the experiments we considered, we did not prompt teachers with any methodology. This, of course, does not mean we believe providing methodological support would not be positive. There is probably a balance to be found between openness (intuitiveness, flexibility) and guidance (support from prompts and/or methodological training).

As a step forward to this direction, an experiment was conducted to test how users reacted to ediT2 openness vs. WebCollage guidance, which results pointed out that "there is no single tool or set of features that are globally perceived as better, rather preferring the use of the tool that best supports them [teachers] in their immediate context and constraints" (see Prieto et al., 2014, for details).

Another way to move forward on this question would be to mix these two approaches. Supporting teachers with repository of scenarios is an approach that is currently attracting growing interest (Larnaca, 2012). We showed how one can introduce scenario patterns via a friendly structure (specific configuration interface) from which a scenario (or a skeleton) is generated. This can then be further adapted

as necessary via the table representation, possibly with pattern-specific services such as constraints-checking mechanisms.

Another perspective is to study if the approach is useful for orchestration and monitoring purposes.

One aspect is how to provide teachers with interesting monitoring information. We showed how the data stored on an LMS such as Moodle could be accessed. This, however, is the technical side only. The open question is to define, via user-centered studies, what data teachers need.

The other aspect is run-time adaptations of the scenario. At this level, the two operationalization strategies we have presented in Chapter 7 are not equivalent. The middleware-based strategy is well adapted for static configuration (the framework is configured once before the session). However, for dynamic adaptations, middleware such as GLUE!-PS is not directly adapted (e.g., moving a student, removing an activity or adding a resource requires re-processing the overall process). Its design rationale is coherent with the Moodle principle of acting as a repository of resources, and run-time flexibility requirements are not considered as such. The *ad hoc* interoperation strategy allows both static and dynamic configuration (i.e., adapting the framework constructions at run-time). Such changes, however, raise consistency issues which must be studied as such (e.g., how/when can one move a student engaged in an on-going task from one group to another?).

9.3.2 Monitoring perspective: exploratory work

We analyzed in an exploratory study how monitoring features could be provided within our approach. For this purpose, we considered the ediT2-Moodle integration. Moodle offers stable and well-known monitoring capabilities supported by its community of collaborators such as, for instance, progress bars denoting activities achieved by students or review features to visualize students' answers to quizzes (Mazza et al., 2012).

The work we conducted does not aim at competing with these tools but, rather, at studying if teachers could be provided with pertinent information within the design editor. The targeted use-case is as follows:

- The teacher launches ediT2 and creates the items needed for each one of its notions (Activity, Group, Participant and Role, when needed).
- She/he imports from Moodle the resources to be used in the session (considering these resources have already been created previously). There is not a specific order between the steps 1 and 2, and the latter can be done before the former.
- She/he designs (totally or partially) the script. In this version, every row is associated with two new buttons, "play" and "monitor".
- She/he launches the first table row (considering that the first table row corresponds to the first activity to be run out) using the "play" button.

- She/he considers the activity progress (using the “monitor” button) and decides whether she/he steps forward, running out the next activity (i.e., hitting its “play” button), or adapts the script before continuing.

Figure 9.1 presents such an enhancement of ediT2. When an activity (a row or sub-row) is launched, its respective green operationalization button is disabled and a new monitoring button, specific to the (sub-)row under manipulation, is automatically showed at the ediT2 interface (top of Figure 9.1). When this new button is hit, a box with awareness information about the unfolding of the task under manipulation is popped up, presenting to the teacher whether her/his students have already accessed, or not, their respective resources in the corresponding Moodle course (bottom of Figure 9.1).

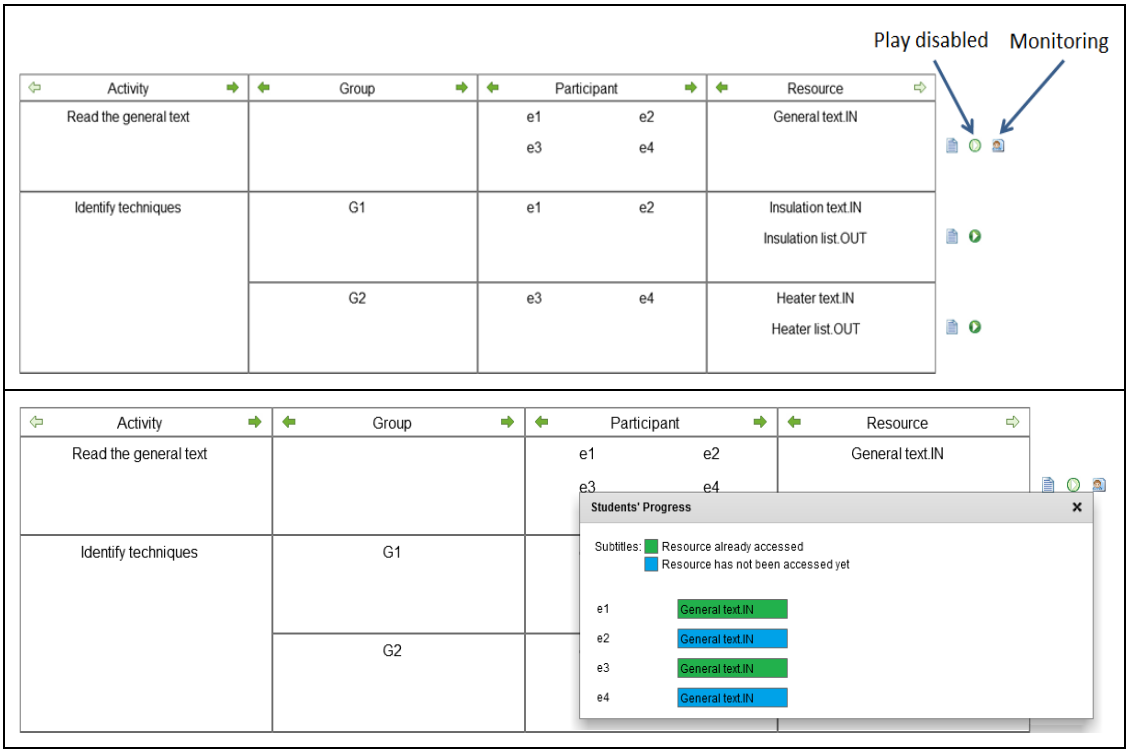


Figure 9.1 ediT2 used as a conducting cockpit (from the jigsaw script presented in Figure 4.3): the first activity is run out (top); the first activity is monitored (bottom)

Graphically, each participant is followed by her/his resource(s), represented by colored horizontal bar(s) as a way to indicate its(their) respective manipulation(s): a green bar means a resource has already been accessed, and a blue one, the opposite. All this information is recuperated from the “log” table at the Moodle data base, as, for instance, timestamp and action performed (e.g., login, view, edit, talk, update or upload). This new ediT2 component is inspired in the Progress Bar Moodle block (Progress Bar, 2013).

Taking into account again the jigsaw script depicted in Figure 4.3 as an example, we can observe in top of Figure 9.1 what happens when one hits the play button corresponding to the first row of the table: at this moment a course named “Read the general text” is created on Moodle (as discussed in Section 7.3; middle of Figure 7.5). The bottom of Figure 9.1 shows a hypothetical situation where two students allocated to such Moodle course (“e1” and “e3”) have already accessed their resources (“General text.IN” for both participants), and two other have not (“e2” and “e4”).

This work was explorative. It opens the door to, in some sense, transform the editor in a kind of conducting cockpit for designing, editing and managing the script unfolding. Within this approach, teachers can directly access the progress of their students through the ediT2 interface, avoiding log on and manipulations procedures to access such information from the Moodle interface.

This, however, opens an array of questions. One of them is run-time intervention issues. If a teacher moves a student from a group to another or reorganizes activities, the effect of such modifications in the Moodle data base may create odd situations to students. Another one is that the available information is limited to the data that can be collected from the Moodle database.

Appendix A – Approach's architectural considerations

In this appendix we present the general architecture used to implement the different modules developed in this work: a Rich Desktop/Internet Application (RDA/RIA) cross-operating different pieces of code through different technologies, as emphasized in Figure A.1. In this figure, blocks are numbered from 1 to 6 and are grouped by the technology on which they are developed: Adobe® (module 1), Haskell (module 2), Java (modules 3 and 4) and PHP (modules 5 and 6).

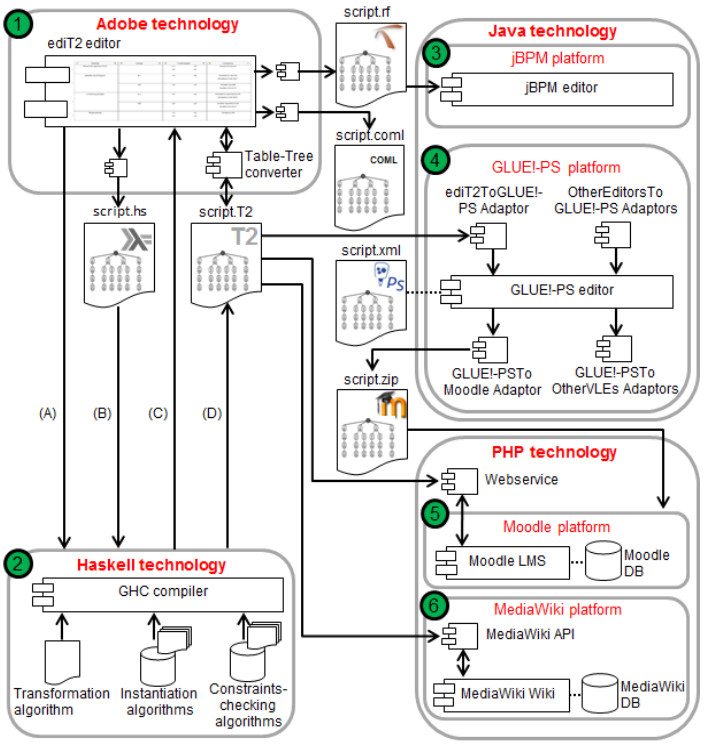


Figure A.1 ediT2 general architecture

The main component of our proposal is the editor itself (module 1), whose implementation requires means to: create rich interfaces (tables); generate (and visually interpret) script-tree structures; access the file system to local storage, and; interoperate the editor with other platforms. In this module, we have used Adobe® Flash® (2014) and Adobe® AIR® (2014) technologies to implement, respectively: the basic/generic Internet-based ediT2 versions (the table/tree editor/model), and; its enhanced desktop-based version, when performing advanced services with direct access to the local file system.

The first versions of ediT2 were based on an own format (named "t2") conceived to computationally represent learning scenarios modeled on its interface

and also, as a way to interoperate ediT2 and other platforms. This format, however, presented two main issues: its readability, and the complexity involved in the construction of parsers required to such interoperations. Therefore, we improved ediT2 in order to manipulate an exchangeable machine-readable format based on XML (named "t2ML"; see Appendix B, for details), specially developed in this thesis as a way to improve the readability of scenarios' files from a standard format, and also to, in future versions, improve the interoperability between ediT2 and new platforms.

ediT2 has other output/intermediary formats used with distinct purposes (see Figure A.1): a "coml" output format, used as a way to show that our system complains to the SceDer Authoring tool; and intermediary Haskell ("hs") and jBPM ("rf") oriented formats, which include additional data to allow interpretation and operationalization within the corresponding technologies as, for instance, Haskell headers and Java objects declarations in modules 2 and 3, respectively.

Some manipulations of the tree structure, such as generating the instantiation of a script, checking constraints or transforming a script structure, can conveniently be developed with a lambda-calculus based language. With this purpose, we have used the Haskell language (Haskell, 2012) to specifically implement such algorithms, and the GHC compiler (GHC, 2012) to interpret and generate tree-structures (module 2).

The Adobe/Haskell data flow between the modules 1 and 2 could be described as follows: (A) pattern parameters (instantiation module input); (B) script transformation and constraint-module's input; (C) constraint-module's output messages, and; (D) transformation and instantiation module's output. Scripts in the "hs" format are manipulated by the constraints-checking algorithm (which output is data to be popped-up in the editor). Transformation and instantiation algorithms produce intermediary "t2" representations, to be presented as tables in the ediT2 interface.

The interoperation between ediT2 and the middleware GLUE!-PS was possible thanks to the construction of the ediT2ToGLUE!-PS component, developed in Java. In this way, scripts produced by ediT2 can then be deployed into Moodle through its course restoration process (modules 4 and 5, respectively).

We have also interoperated ediT2 and Moodle from a web service created in PHP to directly manipulate its database through SQL queries aiming to perform operations such as, for example: creation of courses, enrollment of students, association of resources and students to courses, etc. This integration allows us to think about ediT2 as a very basic learning conducting cockpit, where teachers are able to directly execute (and monitor in real time) scripts from its table interface.

Finally, the modules 3 and 6 were integrated in our proposed architecture to, respectively: (1) allow teachers to represent and simulate scripts requiring complex scheduling mechanisms and; (2) operationalize scripts from the MediaWiki wiki, thanks to the services offered by its API of easy manipulation.

Appendix B – t2ML

B.1 t2ML Schema

The Figure B.1 presents the t2ML Schema diagram. According to it, a learning scenario represented in t2ML is composed by the “script” and the “pattern” elements.

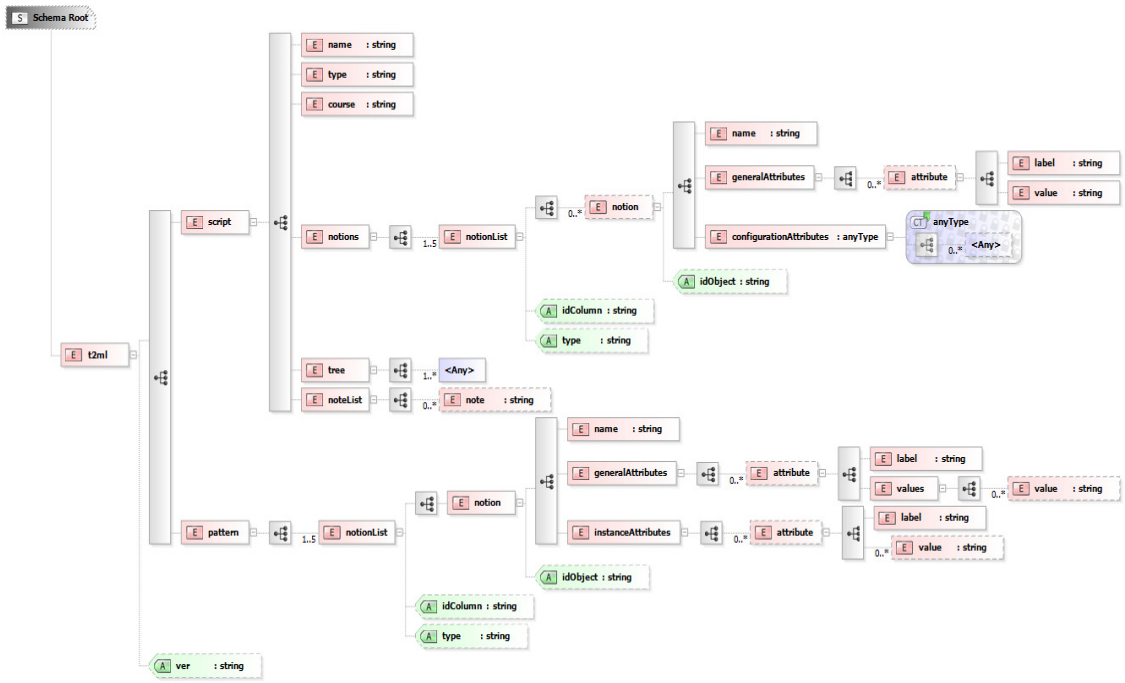


Figure B.1. The t2ml Schema diagram

The first of them (“script”) describes: (1) the course on which participants are imported, when applied; (2) the list of notions used to model ediT2 scenarios and for each of them, its general and configuration attributes’ values, when applied (configuration attributes are parameters used to the integration between ediT2 and external platforms); (3) the tree structure and the instances of the notions’ items displaced on table cells (for each of them, its instance attributes and respective values, when applied), and; (4) the list of teacher notes.

The second element (“pattern”) presents the list of notions and for each of them, the list of general and instance attributes, when applied. This element also identifies for each one of these attributes if this is represented by a label (default) followed by a list of pre-defined values (e.g., for a hypothetical “Interactivity” attribute, “Face-to-Face”, “Through Computers” and “Blended” as possible values).

B.2 t2ML Document

In a way to illustrate an example of t2ml Document, we reconsider the jigsaw script as depicted in Figure 4.3, taking into consideration for each one of its notions the following attributes (Figure B.2; “GAs" and “IAs” represent the general and instance attributes, respectively):

- Activity: “Name”, GAs (“Type”, “Duration”) and IAs (“Beginning time”, “Ending time”);
- Group: “Name”, GAs (“Description”) and IAs (empty);
- Participant: “Name”, GAs (“Age”, “Genre”, “Nationality”) and IAs (empty);
- Resource: “Name”, GAs (“Description”) and IAs (empty);
- Role: “Name”, GAs (“Description”) and IAs (empty).

```
<?xml version="1.0" encoding="UTF-8"?>

<t2ml ver="1.0">
   <script>
      <name>jigsaw script</name>
      <type>glueps</type>
      <course></course>
      <notions>
         <notionList idColumn="0" type="Activity">
            <notion idObject="0">
               <name>Read the general text</name>
               <generalAttributes>
                  <attribute>
                     <label>Type</label>
                     <value>Class</value>
                  </attribute>
                  <attribute>
                     <label>Duration</label>
                     <value>30min</value>
                  </attribute>
               </generalAttributes>
               <configurationAttributes></configurationAttributes>
            </notion>
            <notion idObject="1">
               <name>Identify techniques</name>
               <generalAttributes>
                  <attribute>
                     <label>Type</label>
                     <value>Group</value>
                  </attribute>
                  <attribute>
                     <label>Duration</label>
                     <value>60min</value>
                  </attribute>
               </generalAttributes>
               <configurationAttributes></configurationAttributes>
            </notion>
            <notion idObject="2">
```

```
            <name>Crossing groups</name>
            <generalAttributes>
               <attribute>
                  <label>Type</label>
                  <value>Group</value>
               </attribute>
               <attribute>
                  <label>Duration</label>
                  <value>60min</value>
               </attribute>
            </generalAttributes>
            <configurationAttributes></configurationAttributes>
         </notion>
         <notion idObject="3">
            <name>Regrouping</name>
            <generalAttributes>
               <attribute>
                  <label>Type</label>
                  <value>Class</value>
               </attribute>
               <attribute>
                  <label>Duration</label>
                  <value>120min</value>
               </attribute>
            </generalAttributes>
            <configurationAttributes></configurationAttributes>
         </notion>
      </notionList>
      <notionList idColumn="1" type="Group">
         <notion idObject="0">
            <name>G1</name>
            <generalAttributes>
               <attribute>
                  <label>Description</label>
                  <value>G1 dyad</value>
               </attribute>
            </generalAttributes>
            <configurationAttributes></configurationAttributes>
         </notion>
         <notion idObject="1">
            <name>G2</name>
            <generalAttributes>
               <attribute>
                  <label>Description</label>
                  <value>G2 dyad</value>
               </attribute>
            </generalAttributes>
            <configurationAttributes></configurationAttributes>
         </notion>
      </notionList>
      <notionList idColumn="2" type="Participant">
         <notion idObject="0">
            <name>e1</name>
            <generalAttributes>
               <attribute>
                  <label>Age</label>
                  <value>15</value>
               </attribute>
               <attribute>
```

```
                <label>Genre</label>
                <value>Male</value>
            </attribute>
            <attribute>
                <label>Nationality</label>
                <value>French</value>
            </attribute>
        </generalAttributes>
        <configurationAttributes></configurationAttributes>
    </notion>
    <notion idObject="1">
        <name>e2</name>
        <generalAttributes>
            <attribute>
                <label>Age</label>
                <value>15</value>
            </attribute>
            <attribute>
                <label>Genre</label>
                <value>Female</value>
            </attribute>
            <attribute>
                <label>Nationality</label>
                <value>French</value>
            </attribute>
        </generalAttributes>
        <configurationAttributes></configurationAttributes>
    </notion>
    <notion idObject="2">
        <name>e3</name>
        <generalAttributes>
            <attribute>
                <label>Age</label>
                <value>15</value>
            </attribute>
            <attribute>
                <label>Genre</label>
                <value>Female</value>
            </attribute>
            <attribute>
                <label>Nationality</label>
                <value>French</value>
            </attribute>
        </generalAttributes>
        <configurationAttributes></configurationAttributes>
    </notion>
    <notion idObject="3">
        <name>e4</name>
        <generalAttributes>
            <attribute>
                <label>Age</label>
                <value>15</value>
            </attribute>
            <attribute>
                <label>Genre</label>
                <value>Female</value>
            </attribute>
            <attribute>
                <label>Nationality</label>
```

```
                <value>Brazilian</value>
              </attribute>
            </generalAttributes>
            <configurationAttributes></configurationAttributes>
          </notion>
        </notionList>
        <notionList idColumn="3" type="Resource">
          <notion idObject="0">
            <name>General text.IN</name>
            <generalAttributes>
              <attribute>
                <label>Description</label>
                <value>Document introducing energy saving principles</value>
              </attribute>
            </generalAttributes>
            <configurationAttributes></configurationAttributes>
          </notion>
          <notion idObject="1">
            <name>Insulation text.IN</name>
            <generalAttributes>
              <attribute>
                <label>Description</label>
                <value>Document focusing on insulation</value>
              </attribute>
            </generalAttributes>
            <configurationAttributes></configurationAttributes>
          </notion>
          <notion idObject="2">
            <name>Insulation list.OUT</name>
            <generalAttributes>
              <attribute>
                <label>Description</label>
                <value>Text listing different possible insulation techniques</value>
              </attribute>
            </generalAttributes>
            <configurationAttributes></configurationAttributes>
          </notion>
          <notion idObject="3">
            <name>Heater text.IN</name>
            <generalAttributes>
              <attribute>
                <label>Description</label>
                <value>Document focusing on heating</value>
              </attribute>
            </generalAttributes>
            <configurationAttributes></configurationAttributes>
          </notion>
          <notion idObject="4">
            <name>Heater list.OUT</name>
            <generalAttributes>
              <attribute>
                <label>Description</label>
                <value>Text listing different possible heating techniques</value>
              </attribute>
            </generalAttributes>
            <configurationAttributes></configurationAttributes>
          </notion>
          <notion idObject="5">
            <name>Insulation questions.IN</name>
```

```
            <generalAttributes>
               <attribute>
                  <label>Description</label>
                  <value>Document listing a set of insulation questions</value>
               </attribute>
            </generalAttributes>
            <configurationAttributes></configurationAttributes>
         </notion>
         <notion idObject="6">
            <name>Heater questions.IN</name>
            <generalAttributes>
               <attribute>
                  <label>Description</label>
                  <value>Document listing a set of heating questions</value>
               </attribute>
            </generalAttributes>
            <configurationAttributes></configurationAttributes>
         </notion>
         <notion idObject="7">
            <name>Answers.IN</name>
            <generalAttributes>
               <attribute>
                  <label>Description</label>
                  <value>The final team answer</value>
               </attribute>
            </generalAttributes>
            <configurationAttributes></configurationAttributes>
         </notion>
      </notionList>
      <notionList idColumn="4" type="Role"></notionList>
   </notions>
   <tree>
      <node idColumn="0">
         <instances>
            <instance idInstance="0" idObject="0">
               <instanceAttributes>
                  <attribute>
                     <label>Beginning time</label>
                     <value>10:30</value>
                  </attribute>
                  <attribute>
                     <label>Ending time</label>
                     <value>11:00</value>
                  </attribute>
               </instanceAttributes>
            </instance>
         </instances>
         <tree>
            <node idColumn="1">
               <instances></instances>
               <tree>
                  <node idColumn="2">
                     <instances>
                        <instance idInstance="0" idObject="0">
                           <instanceAttributes></instanceAttributes>
                        </instance>
                        <instance idInstance="1" idObject="1">
                           <instanceAttributes></instanceAttributes>
                        </instance>
```

```
                        <instance idInstance="2" idObject="2">
                            <instanceAttributes></instanceAttributes>
                        </instance>
                        <instance idInstance="3" idObject="3">
                            <instanceAttributes></instanceAttributes>
                        </instance>
                    </instances>
                    <tree>
                        <node idColumn="3">
                            <instances>
                                <instance idInstance="0" idObject="0">
                                    <instanceAttributes></instanceAttributes>
                                </instance>
                            </instances>
                            <tree></tree>
                        </node>
                    </tree>
                </node>
            </tree>
        </node>
    </tree>
</node>
<node idColumn="0">
    <instances>
        <instance idInstance="1" idObject="1">
            <instanceAttributes>
                <attribute>
                    <label>Beginning time</label>
                    <value>11:15</value>
                </attribute>
                <attribute>
                    <label>Ending time</label>
                    <value>12:15</value>
                </attribute>
            </instanceAttributes>
        </instance>
    </instances>
    <tree>
        <node idColumn="1">
            <instances>
                <instance idInstance="0" idObject="0">
                    <instanceAttributes></instanceAttributes>
                </instance>
            </instances>
            <tree>
                <node idColumn="2">
                    <instances>
                        <instance idInstance="4" idObject="0">
                            <instanceAttributes></instanceAttributes>
                        </instance>
                        <instance idInstance="5" idObject="1">
                            <instanceAttributes></instanceAttributes>
                        </instance>
                    </instances>
                    <tree>
                        <node idColumn="3">
                            <instances>
                                <instance idInstance="1" idObject="1">
                                    <instanceAttributes></instanceAttributes>
```

```
                        </instance>
                        <instance idInstance="2" idObject="2">
                           <instanceAttributes></instanceAttributes>
                        </instance>
                     </instances>
                     <tree></tree>
                  </node>
               </tree>
            </node>
         </tree>
      </node>
      <node idColumn="1">
         <instances>
            <instance idInstance="1" idObject="1">
               <instanceAttributes></instanceAttributes>
            </instance>
         </instances>
         <tree>
            <node idColumn="2">
               <instances>
                  <instance idInstance="6" idObject="2">
                     <instanceAttributes></instanceAttributes>
                  </instance>
                  <instance idInstance="7" idObject="3">
                     <instanceAttributes></instanceAttributes>
                  </instance>
               </instances>
               <tree>
                  <node idColumn="3">
                     <instances>
                        <instance idInstance="3" idObject="3">
                           <instanceAttributes></instanceAttributes>
                        </instance>
                        <instance idInstance="4" idObject="4">
                           <instanceAttributes></instanceAttributes>
                        </instance>
                     </instances>
                     <tree></tree>
                  </node>
               </tree>
            </node>
         </tree>
      </node>
   </tree>
</node>
<node idColumn="0">
   <instances>
      <instance idInstance="2" idObject="2">
         <instanceAttributes>
            <attribute>
               <label>Beginning time</label>
               <value>14:00</value>
            </attribute>
            <attribute>
               <label>Ending time</label>
               <value>15:00</value>
            </attribute>
         </instanceAttributes>
      </instance>
```

```
            </instances>
            <tree>
               <node idColumn="1">
                  <instances>
                     <instance idInstance="2" idObject="0">
                        <instanceAttributes></instanceAttributes>
                     </instance>
                  </instances>
                  <tree>
                     <node idColumn="2">
                        <instances>
                           <instance idInstance="8" idObject="0">
                              <instanceAttributes></instanceAttributes>
                           </instance>
                           <instance idInstance="9" idObject="3">
                              <instanceAttributes></instanceAttributes>
                           </instance>
                        </instances>
                        <tree>
                           <node idColumn="3">
                              <instances>
                                 <instance idInstance="5" idObject="5">
                                    <instanceAttributes></instanceAttributes>
                                 </instance>
                                 <instance idInstance="6" idObject="2">
                                    <instanceAttributes></instanceAttributes>
                                 </instance>
                              </instances>
                              <tree></tree>
                           </node>
                        </tree>
                     </node>
                  </tree>
               </node>
               <node idColumn="1">
                  <instances>
                     <instance idInstance="3" idObject="1">
                        <instanceAttributes></instanceAttributes>
                     </instance>
                  </instances>
                  <tree>
                     <node idColumn="2">
                        <instances>
                           <instance idInstance="10" idObject="1">
                              <instanceAttributes></instanceAttributes>
                           </instance>
                           <instance idInstance="11" idObject="2">
                              <instanceAttributes></instanceAttributes>
                           </instance>
                        </instances>
                        <tree>
                           <node idColumn="3">
                              <instances>
                                 <instance idInstance="7" idObject="6">
                                    <instanceAttributes></instanceAttributes>
                                 </instance>
                                 <instance idInstance="8" idObject="4">
                                    <instanceAttributes></instanceAttributes>
                                 </instance>
```

```
                        </instances>
                        <tree></tree>
                     </node>
                  </tree>
               </node>
            </tree>
         </node>
         <node idColumn="0">
            <instances>
               <instance idInstance="3" idObject="3">
                  <instanceAttributes>
                     <attribute>
                        <label>Beginning time</label>
                        <value>15:30</value>
                     </attribute>
                     <attribute>
                        <label>Ending time</label>
                        <value>17:30</value>
                     </attribute>
                  </instanceAttributes>
               </instance>
            </instances>
            <tree>
               <node idColumn="1">
                  <instances></instances>
                  <tree>
                     <node idColumn="2">
                        <instances>
                           <instance idInstance="12" idObject="0">
                              <instanceAttributes></instanceAttributes>
                           </instance>
                           <instance idInstance="13" idObject="1">
                              <instanceAttributes></instanceAttributes>
                           </instance>
                           <instance idInstance="14" idObject="2">
                              <instanceAttributes></instanceAttributes>
                           </instance>
                           <instance idInstance="15" idObject="3">
                              <instanceAttributes></instanceAttributes>
                           </instance>
                        </instances>
                        <tree>
                           <node idColumn="3">
                              <instances>
                                 <instance idInstance="9" idObject="7">
                                    <instanceAttributes></instanceAttributes>
                                 </instance>
                              </instances>
                              <tree></tree>
                           </node>
                        </tree>
                     </node>
                  </tree>
               </node>
            </tree>
         </node>
      </tree>
```

```
    <noteList>
      <note>All students must read a common document</note>
      <note>e1 and e2 must produce a text listing different possible techniques</note>
      <note>e3 and e4 must produce a text listing different possible techniques</note>
      <note>e1 and e4 must write a text</note>
      <note>e2 and e3 must write a text</note>
      <note>All students must prepare a final team answer</note>
    </noteList>
  </script>
  <pattern>
    <notionList idColumn="0" type=" Activity">
      <notion>
        <name></name>
        <generalAttributes>
          <attribute>
            <label>Type</label>
            <values>
            </values>
          </attribute>
          <attribute>
            <label>Duration</label>
            <values>
            </values>
          </attribute>
        </generalAttributes>
        <instanceAttributes>
          <attribute>
            <label>Beginning time</label>
            <values>
            </values>
          </attribute>
          <attribute>
            <label>Ending time</label>
            <values>
            </values>
          </attribute>
        </instanceAttributes>
      </notion>
    </notionList>
    <notionList idColumn="1" type=" Group">
      <notion>
        <name></name>
        <generalAttributes>
          <attribute>
            <label>Description</label>
            <values>
            </values>
          </attribute>
        </generalAttributes>
        <instanceAttributes>
        </instanceAttributes>
      </notion>
    </notionList>
    <notionList idColumn="2" type=" Participant">
      <notion>
        <name></name>
        <generalAttributes>
          <attribute>
            <label>Age</label>
```

```
                <values>
                </values>
            </attribute>
            <attribute>
                <label>Genre</label>
                <values>
                </values>
            </attribute>
            <attribute>
                <label>Nationality</label>
                <values>
                </values>
            </attribute>
        </generalAttributes>
        <instanceAttributes>
        </instanceAttributes>
    </notion>
</notionList>
<notionList idColumn="3" type=" Resource">
    <notion>
        <name></name>
        <generalAttributes>
            <attribute>
                <label>Description</label>
                <values>
                </values>
            </attribute>
        </generalAttributes>
        <instanceAttributes>
        </instanceAttributes>
    </notion>
</notionList>
<notionList idColumn="4" type=" Role">
    <notion>
        <name></name>
        <generalAttributes>
            <attribute>
                <label>Description</label>
                <values>
                </values>
            </attribute>
        </generalAttributes>
        <instanceAttributes>
        </instanceAttributes>
    </notion>
</notionList>
</pattern>
```

Figure B.2. The t2ml Document of the jigsaw script as depicted in Figure 4.3

Appendix C – Trace files considerations

All actions performed by teachers during ediT2 scenarios' manipulation (e.g., displacement of items between cells, insertion/removal of columns, etc.) are registered and can be recuperated afterwards by specialists interested in observing, for instance, when users define or adapt the structure of a learning scenario (i.e., when teachers insert, move or remove table columns).

With this purpose, ediT2 allows such professionals to get this trace information from a .csv file containing such actions distributed through a spreadsheet structured by the following columns: "Timestamp", "Actor", "Action", "Table Dimension", "Tree Dimension" and "Operation Description".

The figure C.1 presents part of a .csv file indicating possible actions performed by an ediT2 user when modeling the jigsaw script as depicted in Figure 4.3. In this figure some rows and columns ("Table Dimension" and "Tree Dimension") were hidden in order to facilitate its readability.

	1	2	3	6
1	Timestamp	Actor	Action	Operation Description
10	03/02/2014 21:53	Teacher	Insert row	
11	03/02/2014 21:53	System	Row inserted	Cell(s) created: "Acti.?" (11), "Group?" (111), "Part.?" (1111), "Reso.?" (11111)
12	03/02/2014 21:53	Model	Subtree created	Node(s) created: "Acti.?" (11), "Group?" (111), "Part.?" (1111), "Reso.?" (11111)
22	03/02/2014 21:53	Teacher	Split cell	"Group?"(121) cell in 2 parts
23	03/02/2014 21:53	System	Cell split	Cell(s) created: "Group?" (122), "Part.?" (1221), "Reso.?" (12211)
24	03/02/2014 21:53	Model	Subtree(s) created	Node(s) created: "Group?" (122), "Part.?" (1221), "Reso.?" (12211)
30	03/02/2014 21:57	Teacher	Insert component(s) into a cell	"Read the general text" component(s) into "Acti.?"(11) cell
31	03/02/2014 21:57	System	Component(s) inserted into the cell	"Read the general text" component(s) into "Acti.?"(11) cell
32	03/02/2014 21:57	Model	Component(s) inserted into the node	"Read the general text" component(s) into "Acti.?"(11) node

Figure C.1. Spreadsheet indicating the actions performed by an ediT2 user when modeling the jigsaw script as depicted in Figure 4.3

In this figure, the operation presented in row #10 corresponds to the action of hitting the "Insert row" button. Once the teacher clicks on this button, the system informs that a new table row has been inserted in the interface (row #11), and the model informs that a new sub-tree has been created in the tree structure (row #12). Similar examples are the "Split cell" (rows #22, #23 and #24) and the "Insert component(s) into a cell" (rows #30, #31 and #32) operations.

Appendix D – Using ediT2 to represent CSCL scripts

As described in Section 5.4, we performed in this work some tests and exploratory studies questioning the power of our model/system to represent a wide range of learning scenarios. Among them, one study was the gathering (and modeling) of 25 CSCL scripts existing in the literature, as a way to indicate the potential of our approach when considering this feature.

In this appendix we introduce only 10 of them. They have been chosen based on the presence of particular characteristics, as, for example: representation of complex scheduling structures (repetitions, parallelisms and/or conditions); technological interrelation with their respective platforms of operationalization, and/or; dynamic instantiations. As we will observe in these scripts, their representations and/or technological interoperability with enactment platforms (when applied) depend on the complexity involved in such interactions.

Some of these scripts allow/demand the repetition of some of their activities several times, according to the needs of the teacher and/or the pedagogical context (e.g., in the “MagicBook”, “RSC” and “Signifié-Signifiant Collaborative Play” scripts, Sections D.6, D.7 and D.8, respectively). In these cases, such steps have been represented only one time in the table, as a way to simplify the readability of their representations.

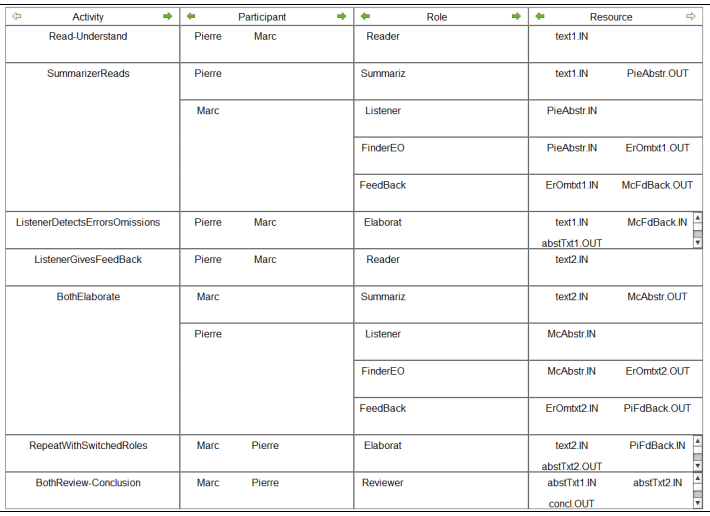
Examples of deployment dependence and complex scheduling features can be found in the script proposed by Roschelle and his colleagues (Section D.9). Another example of the last feature is also found in the script developed by Ioannidou and Dimitracopoulou (Section D.8). Other proposals present instantiation dynamic characteristics, as in the “ArgueGraph” script (Section D.10), where groups must be composed by students with conflicting opinions developed during its unfolding.

As discussed in the previous chapters, some alternatives to overcome these issues were proposed and implemented. Taking into account the basic ediT2 implementation (e.g., without advanced features as representation of complex scheduling), repetitions and rotation mechanisms could be designed from the duplication of some table rows/items. The level of difficulty increases if the flow conditions, deployment completeness or dynamic instantiations, which demands the extension of ediT2 functionalities through advanced services, must be considered.

In all examples discussed in this appendix, we have used the labels .IN/.OUT suffixing the resources’ names to indicate that they represent respectively input/output artefacts or, in another way, that they are resources to be used/produced by the students involved in a certain CSCL activity (as the same way that it has been done in Figure 4.3, for the jigsaw script).

D.1 MURDER

"MURDER is short for Mood, Understanding, Recall, Detection, Elaboration, and Review. Students learn in pairs from a textbook, one being the summarizer and the other the listener. After setting the mood for studying, both read a text passage for understanding. The Summarizer recalls what has been read while the listener detects errors or omissions and gives feedback. Then both elaborate on the read passage and repeat everything with switched roles for the next passage of text. Finally, both review the read passages and reflect on what they have learned."

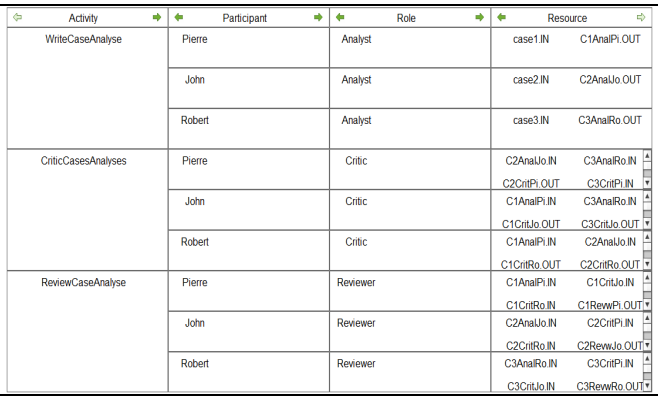


Activity	Participant	Role	Resource
Read-Understand	Pierre Marc	Reader	text1.IN
SummarizerReads	Pierre	Summariz	text1.IN PieAbstr.OUT
	Marc	Listener	PieAbstr.IN
		FinderEO	PieAbstr.IN ErOmtxt1.OUT
		FeedBack	ErOmtxt1.IN McFdBack.OUT
ListenerDetectsErrorsOmissions	Pierre Marc	Elaborat	text1.IN McFdBack.IN absTxt1.OUT
ListenerGivesFeedBack	Pierre Marc	Reader	text2.IN
BothElaborate	Marc	Summariz	text2.IN McAbstr.OUT
	Pierre	Listener	McAbstr.IN
		FinderEO	McAbstr.IN ErOmtxt2.OUT
		FeedBack	ErOmtxt2.IN PiFdBack.OUT
RepeatWithSwitchedRoles	Marc Pierre	Elaborat	text2.IN PiFdBack.IN absTxt2.OUT
BothReview-Conclusion	Marc Pierre	Reviewer	absTxt1.IN absTxt2.IN concl.OUT

Figure D.1 The "MURDER" script narrative (top) and representation (bottom) (Dansereau et al., 1979 *apud* Kobbe et al., 2007)

D.2 Social

"Three case studies are analyzed and reviewed by groups of three students in parallel. Each student writes a case analysis, then critiques the other two written case analyses and finally revises his/her own case analysis based on the critiques received by the other students. Both roles of case analyst and constructive critic are additionally supported with text prompts that learners are supposed to act out."

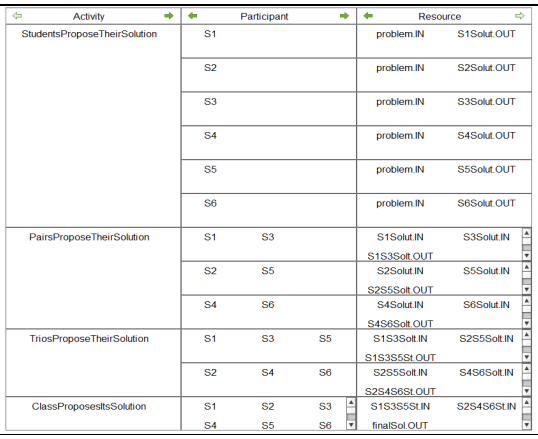


Activity	Participant	Role	Resource
WriteCaseAnalyse	Pierre	Analyst	case1.IN C1AnalPi.OUT
	John	Analyst	case2.IN C2AnaUo.OUT
	Robert	Analyst	case3.IN C3AnalRo.OUT
CriticCasesAnalyses	Pierre	Critic	C2AnaUo.IN C3AnalRo.IN C2CritPi.OUT C3CritPi.IN
	John	Critic	C1AnalPi.IN C3AnalRo.IN C1CritJo.OUT C3CritJo.OUT
	Robert	Critic	C1AnalPi.IN C2AnaUo.IN C1CritRo.OUT C2CritRo.OUT
ReviewCaseAnalyse	Pierre	Reviewer	C1AnalPi.IN C1CritJo.IN C1CritRo.IN C1RewPi.OUT
	John	Reviewer	C2AnaUo.IN C2CritPi.IN C2CritRo.IN C2RewJo.OUT
	Robert	Reviewer	C3AnalRo.IN C3CritPi.IN C3CritJo.IN C3RewRo.OUT

Figure D.2 The "Social" script narrative (top) and representation (bottom) (Weinberger et al., 2005 *apud* Kobbe et al., 2007)

D.3 Pyramid

Initially, each leaner studies a problem and proposes an initial solution. Then, the teacher encourages the students to compose groups (usually pairs) in order to expand and also deepen their perspective in the domain to propose a new shared solution. After, the students must be guided to join in larger groups in order to generate new agreed proposals. This process must be repeated until that, at the end, all students are put in a debriefing session to propose a final and agreed solution at the classroom level.

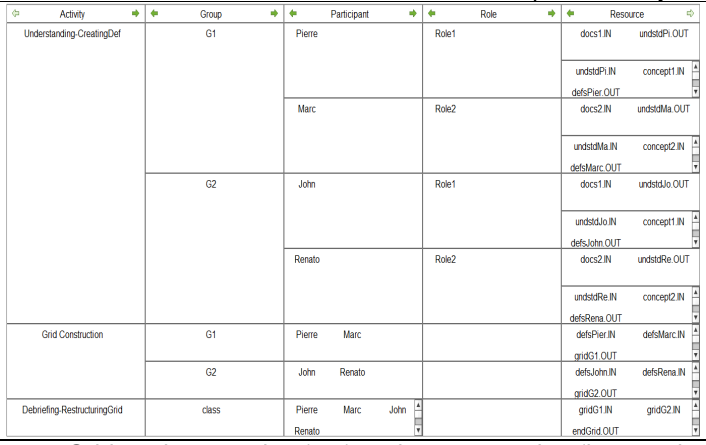


Activity	Participant	Resource
StudentsProposeTheirSolution	S1	problem.IN S1Solut.OUT
	S2	problem.IN S2Solut.OUT
	S3	problem.IN S3Solut.OUT
	S4	problem.IN S4Solut.OUT
	S5	problem.IN S5Solut.OUT
	S6	problem.IN S6Solut.OUT
PairsProposeTheirSolution	S1 S3	S1Solut.IN S3Solut.IN S1S3Solt.OUT
	S2 S5	S2Solut.IN S5Solut.IN S2S5Solt.OUT
	S4 S6	S4Solut.IN S6Solut.IN S4S6Solt.OUT
TriosProposeTheirSolution	S1 S3 S5	S1S3Solt.IN S2S5Solt.IN S1S3S5St.OUT
	S2 S4 S6	S2S5Solt.IN S4S6Solt.IN S2S4S6St.OUT
ClassProposesItsSolution	S1 S2 S3 S4 S5 S6	S1S3S5St.IN S2S4S6St.IN finalSol.OUT

Figure D.3 The "Pyramid" script narrative (top) and representation (bottom; instanced for 6 students) (adapted from Karakostas and Demetriadis, 2009)

D.4 ConceptGrid

Groups of *n* students are composed. Each of these groups has to play *n* roles, one for each of its participants, associated with papers to read. After, each student reads the papers associated with his or her role. Then, the group distributes the concepts to be defined among its members. Each student enters a 5-10-line definition of the concepts that she/he has chosen to define. The group constructs a grid with these concepts ordered on a map in such a way that two neighboring concepts can be explained in just a few sentences. Finally, a debriefing section takes place to reformulate the definitions and relations provided by the students.



Activity	Group	Participant	Role	Resource
Understanding-CreatingDef	G1	Pierre	Role1	docs1.IN undstdPi.OUT undstdPi.IN concept1.IN defsPier.OUT
		Marc	Role2	docs2.IN undstdMa.OUT undstdMa.IN concept2.IN defsMarc.OUT
	G2	John	Role1	docs1.IN undstdJo.OUT undstdJo.IN concept1.IN defsJohn.OUT
		Renato	Role2	docs2.IN undstdRe.OUT undstdRe.IN concept2.IN defsRena.OUT
Grid Construction	G1	Pierre Marc		defsPier.IN defsMarc.IN gridG1.OUT
	G2	John Renato		defsJohn.IN defsRena.IN gridG2.OUT
Debriefing-RestructuringGrid	class	Pierre Marc John Renato		gridG1.IN gridG2.IN endGrid.OUT

Figure D.4 The "ConceptGrid" script narrative (top) and representation (bottom; instanced for 2 groups composed by pairs) (adapted from Dillenbourg et al., 2009).

D.5 Universanté

"Students from different nations solve problem cases in mixed and changing teams. Each case is first read and discussed in teams of mixed nations. National teams then inform each other about the cases read and create a national fact sheet. These national fact sheets are then compared and compiled by teams of students with thematically similar cases (mixed nations). These same teams present their compiled fact sheets to other teams of their same nationality and receive feedback. Finally, students return to their initial case group and work out a case solution."

Activity	Group	Participant			Resource	
Read-Discussion	InternationalTeam1	Pierre	Maria	John	case1.IN	cas1Disc.OUT
	InternationalTeam2	Marc	Renato	Peter	case2.IN	cas2Disc.OUT
	InternationalTeam3	Robert	Paulo	Edward	case3.IN	cas3Disc.OUT
NationalSheetCreation	NationalTeam1	Pierre	Marc	Robert	cas1Disc.IN	cas2Disc.IN
					cas3Disc.IN	NatSht1.OUT
	NationalTeam2	Maria	Renato	Paulo	cas1Disc.IN	cas2Disc.IN
					cas3Disc.IN	NatSht2.OUT
	NationalTeam3	John	Peter	Edward	cas1Disc.IN	cas2Disc.IN
					cas3Disc.IN	NatSht3.OUT
Comparison-Compilation	InternationalTeam1	Pierre	Maria	John	NatSht1.IN	NatSht2.IN
					NatSht3.IN	IntComp1.OUT
	InternationalTeam2	Marc	Renato	Peter	NatSht1.IN	NatSht2.IN
					NatSht3.IN	IntComp2.OUT
	InternationalTeam3	Robert	Paulo	Edward	NatSht1.IN	NatSht2.IN
					NatSht3.IN	IntComp3.OUT
PresentSheet-ReceiveFeedBack	NationalTeam1	Pierre	Marc	Robert	IntComp1.IN	IntComp2.IN
					IntComp3.IN	NatFbBc1.OUT
	NationalTeam2	Maria	Renato	Paulo	IntComp1.IN	IntComp2.IN
					IntComp3.IN	NatFbBc2.OUT
	NationalTeam3	John	Peter	Edward	IntComp1.IN	IntComp2.IN
					IntComp3.IN	NatFbBc3.OUT
FindOutCaseSolution	InternationalTeam1	Pierre	Maria	John	NatFbBc1.IN	NatFbBc2.IN
					NatFbBc3.IN	caseSol1.OUT
	InternationalTeam2	Marc	Renato	Peter	NatFbBc1.IN	NatFbBc2.IN
					NatFbBc3.IN	caseSol2.OUT
	InternationalTeam3	Robert	Paulo	Edward	NatFbBc1.IN	NatFbBc2.IN
					NatFbBc3.IN	caseSol3.OUT

Figure D.5 The "Universanté" script narrative (top) and representation (bottom; instanced for triplets from 3 different countries) (Dillenbourg and Jermann, 2007 *apud* Kobbe et al., 2007)

D.6 MagicBook

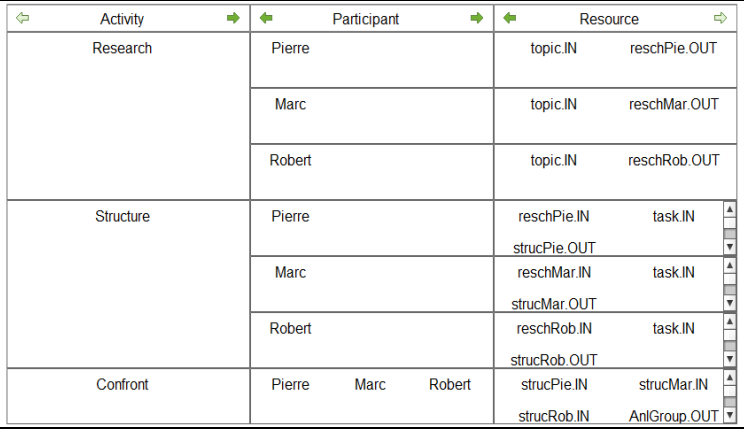
The teacher writes the beginning of a story and all participants read it (a participant can be defined as a student or a whole class of students) – this step can be iterated several times. After, all participants write a second chapter and propose it as a continuation of the story. The proposals for the next chapter are read by the participants that finally vote for their favorite, which will be elected as the official chapter 2.

Activity	Participant			Resource	
ReadChapter1-WriteChapter2	Pierre			chapter1.IN	chap2Pie.OUT
	Marc			chapter1.IN	chap2Mar.OUT
	Robert			chapter1.IN	chap2Rob.OUT
ReadAllChapProd-ChooseTheBest	Pierre	Marc	Robert	chap2Pie.IN	chap2Mar.IN
				chap2Rob.IN	chap2Rob.OUT

Figure D.6 The "MagicBook" script narrative (top) and representation (bottom; instanced for 3 students) (adapted from Dillenbourg, 2002)

D.7 RSC

This approach includes 3 phases (corresponding to its RSC acronym – Research-Structure-Confront), where the output of a phase is the input of the next one, and which can be repeated several times. Firstly, each student of a group has to freely research the Internet to discover some information and become familiar with a given same topic. After, each student has to structure and/or use the data he/she has recovered according to a task. Finally, the group has to elaborate a collective construction from the individual productions (confronting the individual productions and collectively constructing an analysis of the topic researched).



Activity	Participant	Resource	
Research	Pierre	topic.IN	reschPie.OUT
	Marc	topic.IN	reschMar.OUT
	Robert	topic.IN	reschRob.OUT
Structure	Pierre	reschPie.IN	task.IN
		strucPie.OUT	
	Marc	reschMar.IN	task.IN
		strucMar.OUT	
	Robert	reschRob.IN	task.IN
		strucRob.OUT	
Confront	Pierre Marc Robert	strucPie.IN	strucMar.IN
		strucRob.IN	AnlGroup.OUT

Figure D.7 The "RSC" script narrative (top) and representation (bottom; instanced for 3 students) (Betbeder and Tchounikine, 2003 *apud* Dillenbourg and Tchounikine, 2007 (adapted))

D.8 Signifié-Signifiant Collaborative Play

Two groups of students communicate each other while working in one of two concept representational modes: for example, a base team (BT, working on a 2D map) and a field team (FT, acting in a 3D natural space). Initially, students agree on the general nature of the task, the starting point of motion and the instructions of motion execution. BT's participants formulate and express verbal instructions to FT's participants. These negotiate among them the transmitted instructions, assess their appropriateness and execute them (in case of acceptation). BT's participants interpret the oral reactions of the other team, translate the feedback represented on the screen and rethink the situation in case of wrong. Groups negotiate the next step of the motion until the final landmark. They discuss in a debriefing session to clarify eventual problems appeared during the script execution. Finally, team roles are changed and the process is re-started.

Activity	Participant	Role	Resource
Agreement	BTeam FTeam		task.IN agreemen.OUT
BTeamAsks-FTeamReacts	BTeam	Asker	task.IN movInst1.OUT
	FTeam	Executer	movInst1.IN assessI1.OUT
			assessI1.IN react1.OUT
BTeamObserves-RethinkSituation	BTeam		react1.IN fbBack1.IN rThinkS1.OUT
GroupsNegotiateNextStep	BTeam FTeam		task.IN rThinkS1.IN nextStep.OUT
BTeamAsks-FTeamReacts	BTeam	Asker	task.IN nextStep.IN monInst2.OUT
	FTeam	Executer	monInst2.IN assessI2.OUT
			assessI2.IN react2.OUT
BTeamObserves-RethinkSituation	BTeam		react2.IN fdBack2.IN rThinkS2.OUT
DebriefingToClarifyProblems	BTeam FTeam		rThinkS1.IN rThinkS2.IN problem1.OUT

Figure D.8 The "Signifié-Signifiant Collaborative Play" script narrative (top) and representation (bottom; here, the change of the teams' roles is not modeled for the figure readability sake) (Ioannidou and Dimitracopoulou, 2003, 2004 *apud* Van de Velde et al., 2004 (adapted))

D.9 Scaffolding Group Feedback and Explanation During Practice Time

This approach "presents tasks to groups of three students via wireless handheld devices. To complete a task, a group must work cooperatively. Each student enters an answer independently; however, the system requires that students agree on an answer and provides feedback only at the group level. If students do not choose the same answer, the software tells them they must agree, which generates much discussion. Once students agree, the software tells them whether they were all right or all wrong. If wrong, they must try again, while the software makes the previously incorrect choice unavailable so that students individually select a different answer until they select the correct one."

Activity	Participant	Resource
StudentsAnswerTasks	S1	tasks.IN S1Answer.OUT
	S2	tasks.IN S2Answer.OUT
	S3	tasks.IN S3Answer.OUT
StudentsCreateCooperatAnswer	S1 S2 S3	S1Answer.IN S2Answer.IN S3Answer.IN coopSolt.OUT

Figure D.9 The "Scaffolding Group Feedback and Explanation During Practice Time" script narrative (top) and representation (bottom; instanced for 3 students and considering that in the second activity all students have agreed on a same correct answer) (adapted from Roschelle et al., 2009)

D.10 ArgueGraph

"Students first individually argue for or against items on a questionnaire. Their opinion is plotted onto a two-dimensional graph. Students with highly conflicting opinions (point distance in the graph) are grouped together in pairs and receive another copy of the questionnaire to fill out. Students discuss what arguments to write for each item. The teacher collects the questionnaires and helps each small group in turn to elaborate on and revise their arguments. The teacher then groups all arguments by item. Finally, each student is assigned one item for which to write a synthesis of all arguments."

Activity	Participant	Resource	
ArguingIndividuallyQuestions	Pierre	question.IN	PierArgm.OUT
	Marc	question.IN	MarcArgm.OUT
	Maria	question.IN	MariArgm.OUT
	John	question.IN	JohnArgm.OUT
Discussion-AgreementInPairs	Pierre John	question.IN	PierArgm.IN
		JohnArgm.IN	Pair1Dsc.OUT
		Pair1Dsc.IN	qtnP1Agr.OUT
	Marc Maria	question.IN	MarcArgm.IN
		MariArgm.IN	Pair2Dsc.OUT
		Pair2Dsc.IN	qtnP2Agr.OUT
DebriefingInClass	Pierre Marc Maria John	qtnP1Agr.IN	qtnP2Agr.IN
		DebfElab.OUT	
SynthesizingIndividually	Pierre	DebfElab.IN	PierSynt.OUT
	Marc	DebfElab.IN	MarcSynt.OUT
	Maria	DebfElab.IN	MariSynt.OUT
	John	DebfElab.IN	JohnSynt.OUT

Figure D.10 The "ArgueGraph" script narrative (top) and representation (bottom; instanced for 4 students) (Dillenbourg and Jermann, 2007 *apud* Kobbe et al., 2007)

Appendix E – Analysis of the work

In (Tchounikine, 2011), an analysis grid for TEL works is proposed. With respect to this grid, the work conducted in this thesis can be analyzed considering the design context and the computer-based system (Table E.1) – in this table, CBPS stands for "Computer Based Pedagogical Setting".

Table E.1 Analysis grid for TEL works (inspired from Tchounikine, 2011)

Characterizing the design context
1. Nature of the work: research work
2. Theoretical background: N/A

3. Nature of the targeted outcome
• Construction of an object, a product or a service
  o Constructing a model of some dimension transversal to different domains
  o Constructing a piece of software
  o Providing teachers with editing tools
• Addressing of a pedagogical objective:
  o Some teachers' tasks have been facilitated or supported
• Elaboration of some statement or lesson learned
• The elaboration of a model:
  o Model as a CS design or specification tool
  o Model as an intermediate structure in between two objects (other models, software components, etc.)
  o Model as a run-time control or configuration tool for a process (interface adaptation, data flow control, feedback and support control, etc.)
• Elaboration of a process or methodological considerations:
  o A description of issues
  o A general approach to some issue
  o An engineering or re-engineering process
• Implementation of some software / software components:
  o Implement a visualization tool that allows a particular data presentation

4. Rationale for designing software • Large considerations:

- Obtain data related to users' usage in order to elaborate some understanding of something
- Obtain an experimental setting that allows testing of a hypothesis or a model
- Allow understanding of whether some construction can be made computable (operationalization of a model or process, interoperability means, etc.)
- Allow exploration of an idea

5. How software is considered within the CBPS

- Role of software in the CBPS:
  - Software is considered to be merely a resource
- Constraints applying to the design/implementation processes:
  - Obtaining software that is usable (i.e., usability may be shown or argued)

6. Design approach

- Design entry point:
  - An innovative idea
  - A technology allowing new possibilities
  - An analysis of some teacher's (tutor's, etc.) tasks to be supported
  - The identification of an anomaly to be reduced
  - A compilation of empirical results allowing elaboration of design guidelines
- The design model:
  - Iterative design (testing incrementally different versions)
  - User-centered design (placing users at the center of design)
- How users are concerned in design:
  - The conditions under which users are involved (by whom, at what time, what to do, etc.): tests in lab

7. Actors concerned

- Types of actors concerned (or actors' roles):
  - Researcher
  - Teacher

8. Context and historical dimensions

- Elements forming the original context of software design:
  - An identified learning or teaching issue
  - A technology
  - An innovative idea
- General context and the history of the project:

○ The history and/or evolution of the project: focused on CSCL, and then generalized
Characterizing the computer-based system
9. Level of analysis of software properties • Software is considered at the level of the objects and features detailed properties
10. Actions considered at the level of software • Teachers' actions when editing a scenario
11. Reification of the pedagogical intention in software • Features related to pedagogical considerations
12. Nature of the CS treatments • Acquire data: ○ Acquire the computational events (logs) generated by teachers' editing actions • Elaborate data: ○ Elaborate a tree representation for configuration data • Visualize data: ○ Interface following a particular structure ○ Simulation interface ○ Supervision tool • Manage access to data ○ Manage access to enactment frameworks data • Handle a complex task: ○ Manage a sequence editing / operationalization / supervision
13. Level of achievement • Different level of achievement are possible: ○ Prototype implementing main functions
14. Results • Results are related to: ○ The artifact (software, CBPS) that has been designed, for instance the fact that software presents some properties ○ The satisfaction of the actors (teachers) ○ The fact that a method, a model, a theory or a technology appears to be a base for, guides, allows or facilitates design of pedagogical-setting support software

Bibliography

.LRN, 2013. Learn, Research, Network Consortium (www.dotlrn.org), visited in 12/2013.

AIR, 2014. Adobe AIR 3 (www.adobe.com/fr/products/air.html), visited in 03/2014.

Baker, M. & Lund, K. (1997). Promoting reflective interactions in a CSCL environment. Journal of Computer Assisted Learning, 13(3), 175–193.

Bakia, M., Murphy, R., Anderson, K. & Trinidad, G.E. (2011). International Experiences With Technology in Education: Final Report. U.S. Department of Education, Office of Educational Technology. Washington, D.C.

Belgiorno, F., Chiara, R.D., Manno, I. & Scarano, V. (2008). A Flexible and Tailorable Architecture for Scripts in F2F Collaboration. In Proceedings of the 3rd European Conference on Technology Enhanced Learning: Times of Convergence: Technologies Across Learning Contexts. Springer-Verlag Berlin, pp. 401-412.

Blackboard. Blackboard Learning System (www.blackboard.com), visited in 12/2013.

Boloudakis, M., Katsamani, M., Retalis, S., Georgiakakis P. (2012). CADMOS: A learning design tool for Moodle courses. First Moodle Research Conference, Heraklion, Crete, Greece.

Botturi, L., Derntl, M., Boot, E. & Figl, K. (2006). A Classification Framework for Educational Modeling Languages in Instructional Design. In Proceedings of 6th IEEE International Conference on Advanced Learning Technologies, pp. 1216-1220.

Botturi, L., Burgos, D., Caeiro, M., Derntl, M., Koper, R., Parrish, P., Sodhi, T., Tattersal, C. (2008). Comparing Visual Instructional Design Languages, A Case Study, in: L. Botturi, T. Stubbs (Eds.), Handbook of Visual Languages for Instructional Design: Theories and Practices. Information Science Reference, Hershey, pp. 155-184.

Chiara, R., Matteo, A., Manno, I., Scarano, V. (2007). CoFFEE: Cooperative Face2Face educational environment, in: International Conference on Collaborative Computing: Networking, Applications and Worksharing, pp. 243-252.

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

Coppercore. The IMS Learning Design Engine (coppercore.sourceforge.net), visited in 06/2012.

De Wever, B., Van Keer, H., Schellens, T., & Valcke, M. (2009). Structuring asynchronous discussion groups: The impact of role assignment and self-assessment on students' levels of knowledge construction through social negotiation. Journal of Computer Assisted Learning, 25, 177–188.

De-la-Fuente-Valentín, L., Pardo, A., Kloos, C. D. (2007). Experiences with GRAIL: Learning Design support in. LRN. TENCompetence Open workshop on current research in IMS Learning Design and lifelong competence development infrastructures, Citeseer.

Derntl, M. (2011). OpenGLM: The Open Graphical Learning Modeller. Proceedings of the Art and Science of Learning Design International Workshop. ASLD 2011.

Derntl, M., Neumann, S., Griffiths D. & Oberhuemer, P. (2011). The conceptual structure of IMS Learning Design does not impede its use for authoring. IEEE Transactions on Learning Technologies, 5(1), 74-86.

Dillenbourg, P. (2002). Over-scripting CSCL: The risks of blending collaborative learning with instructional design. In P. A. Kirschner (Ed.), pp.61-91.

Dillenbourg, P., Dimitriadis, Y., Nussbaum, M. (2013). Design for Classroom Orchestration. Computers & Education, 69, 485-492.

Dillenbourg, P. & Hong, F. (2008). The mechanics of CSCL macro scripts. International Journal of Computer-Supported Collaborative Learning, Springer 3 (1), 5-23.

Dillenbourg, P., Hong, F. & Brahm, T. (2009). The Manyscripts Pedagogical Handbook: How to build scripts for collaborative learning?. scil Arbeitsbericht. St. Gallen: scil, Universität St. Gallen.

Dillenbourg, P., Jermann, P. (2010). Technology for classroom orchestration, in: Khine, I. M. (Ed.), The New Science of Learning: Computers, Cognition and Collaboration in Education, Springer, Berlin, pp. 525-552.

Dillenbourg, P. & Jermann, P. (2007). Designing Integrative Scripts. F. Fischer, I. Kollar, H. Mandl & J. M. Haake (Eds.). Computer-Supported Collaborative Learning Series, Springer, pp.275-301.

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

Diziol, D., Walker, E., Rummel, N. & Koedinger, K (2010) Using intelligent tutor technology to implement adaptive support for student collaboration. Educational Psychology Review. 22 (1), 89-102.

Drira, R., Laroussi, M., Le Pallec, X., Warin, B., 2012. Contextualizing Learning Scenarios According to Different Learning Management Systems. IEEE Transactions on Learning Technologies 5(3), 213-225.

Eclipse. The Eclipse Foundation open source community website (www.eclipse.org), visited in 12/2012.

Fischer, F., Kollar, I., Haake, J.M. & Mandl, H. (2007). Perspectives on collaboration scripts. In F. Fischer, I. Kollar, H. Mandl & J. M. Haake (Eds.), Scripting computer-supported communication of knowledge – cognitive, computational, and educational perspectives, New York: Springer, pp. 1-10.

Flash, 2014. Flash Professional CC (www.adobe.com/fr/products/flash.html), visited in 03/2014.

GHC. The Glasgow Haskell Compiler (www.haskell.org/ghc), visited in 02/2012.

Goodyear, P. & Retalis, S. (2010). Technology-enhanced learning: design patterns and pattern languages. Sense Publishers, Rotterdam.

Group Scribbles. Rapidly Design Collaborative Learning Activities (groupscribbles.sri.com), visited in 11/2013.

Haake, J.M. & Pfister, H.-R. (2007). Flexible scripting in Net-Based Learning Groups. F. Fischer, I. Kollar, H. Mandl & J. M. Haake (Eds.). Computer-Supported Collaborative Learning Series, Springer, pp.155-175.

Harrer, A. & Hoppe, H. U. (2008). Visual Modeling of Collaborative Learning Processes – Uses, Desired Properties and Approaches, in: L. Botturi & T. Stubbs (Eds.). Handbook of Visual Languages for Instructional Design: Theory and Practices, Information Science Reference, Hershey, pp. 281-298.

Harrer, A., Kohen-Vacs, D., Roth, B., Malzahn, N., Hoppe, H.U. & Ronen, M. (2009). Design and enactment of collaboration scripts – an integrative approach with graphical notations and learning platforms. International Society of the Learning Sciences, CSCL Conference, pp. 198-200.

Harrer, A., Malzahn, N. & Hoppe, H.U. (2007). Graphical Modeling and Simulation of Learning Designs. In T. Hirashima, H. U. Hoppe and S. S. Young (Eds.). Supporting Learning Flow through Integrative Technologies. Amsterdam, IOS Press, pp. 291-294.

Haskell. The Haskell Programming Language (www.haskell.org), visited in 02/2012.

Hernández-Leo, D., Asensio-Pérez, J.I., Dimitriadis, Y., Villasclaras-Fernández, E.D. (2010). Generating CSCL Scripts: From a Conceptual Model of Pattern Languages to the Design a real situation (Appendix). In: Goodyear, P., Retalis, S. (Eds) Technology Enhanced Learning, Design Patterns and Pattern Languages, SensePublishers.

Hernández-Leo, D., Villasclaras-Fernández, E.D., Asensio-Pérez, J.I., Dimitriadis, Y., Jorrín-Abellán, I.M., Ruiz-Requies, I. & Rubia-Avi, B. (2006). COLLAGE: A collaborative Learning Design editor based on patterns. Journal of Educational Technology and Society, 9(1), 58-71.

Honegger, B. D., Notari, M. P. (2009). Over-computing CSCL Macro scripts? Gaining flexibility by using WikiPlus instead of specialized tools for authoring macro scripts. Proceedings of the 9th International Conference on Computer Supported Collaborative Learning. International Society of the Learning Sciences, v.1, pp 482-486.

IMS GLC. Instructional Management Systems Global Learning Consortium (www.imsglobal.org), visited 12/2013.

IMS-LD. IMS Learning Design (http://www.imsglobal.org/learningdesign), visited 08/2012.

jBPM. A flexible Business Process Management Suite (www.jboss.org/jbpm), visited in 12/2012.

Jones, C., Dirckinck-Holmfeld, L. & Lindström, B. (2006). A relational, indirect, meso-level approach to CSCL design in the next decade. International Journal of Computer-Supported Collaborative Learning. 1(1), 35-56.

Karakostas, A. & Demetriadis, S. (2009). Adaptation Patterns in Systems for Scripted Collaboration. Proceedings of the 9th International Conference on Computer-Supported Collaborative Learning. International Society of the Learning Sciences, pp. 477-481.

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

Kohen-Vacs, D., Ronen, M., Hammer, R. (2011). Designing, Enacting & Sharing Collaborative Online Activities with CeLS. Proceedings of the Art and Science of Learning Design International Workshop. ASLD 2011.

Kollar, I., Fischer, F., Hesse, F. (2006). Collaboration Scripts – A Conceptual Analysis. Educational Psychology Review, 18(2), pp. 159-185.

Kollar, I., Fischer, F., & Slotta, J.D. (2007). Internal and external scripts in computer-supported collaborative inquiry learning. Learning and Instruction, 17(6), 708–721.

Laforcade, P. & Choquet, C. (2006). Next Step for Educational Modeling Languages: The Model Driven Engineering and Reengineering Approach, in: Proceedings of the 6th IEEE International Conference on Advanced Learning Technologies, pp.745-747.

LAMS. Learning Activity Management System (www.lamsfoundation.org), visited in 08/2013.

Larnaca. The Larnaca Declaration on Learning Design. (www.larnacadeclaration.org), visited in 12/2012.

Larusson, J. A., Alterman, R. (2009). Wikis to support the "collaborative" part of collaborative learning. International Journal of Computer-Supported Collaborative Learning. 4(4), 371-402.

Laurillard, D., Charlton, P., Craft, B., Dimakopoulos, D., Ljubojevic, D., Magoulas, G., Masterman, E., Pujadas, R., Whitley, E.A., Whittlestone, K. (2013). A constructionist learning environment for teachers to model learning designs. Journal of Computer Assisted Learning 29(1), 15-30.

Lejeune, A. (2004). IMS Learning Design. Distances et savoirs, 2, 409-450.

Malzahn, N., Pokrandt, M., & Hoppe, H. U. (2008) Extending a Learning Design Editor with a Monitoring Component. International Conference on Computers in Education, 499-504.

Mangano, N., Ossher, H., Simmonds, I., Callery, M., Desmond, M., Krasikov, S. (2011). Blending freeform and managed information in tables: NIER track. 33rd International Conference on Software Engineering, pp. 840-843.

Mazza, R., Bettoni, M., Faré, M., Mazzola, L. (2012). MOCLog – Monitoring Online Courses with log data. In: S. Retalis and M. Dougiamas. 1st Moodle Research Conference Proceedings. ISBN: 978-960-98516-2-6

McLaren, B.M., Scheuer, O. & Mikšátko, J. (2010) Supporting Collaborative Learning and e-Discussions Using Artificial Intelligence Techniques. International Journal of Artificial Intelligence in Education. 20, 1-46.

MediaWiki. A free open source wiki package (www.mediawiki.org), visited in 02/2013.

Miao, Y., Hoeksema, K., Hoppe, H.U. & Harrer, A. (2005). CSCL Scripts: Modelling Features and Potential Use. International Conference on Computer Supported Collaborative Learning, Taipei, Taiwan, pp. 423-432.

Moodle. Open-source community-based tools for learning (www.moodle.org), visited in 08/2012.

Mor, Y., Craft, B. (2012). Learning Design: Reflections upon the Current Landscape. Research in Learning Technology, 20, 85-94.

Muñoz-Cristóbal, J.A., Prieto, L.P., Asensio-Pérez, J.I., Jorrín-Abellán, I.M., Dimitriadis, Y. (2012) Lost in Translation from Abstract Learning Design to ICT Implementation: A Study Using Moodle for CSCL. Lecture Notes in Computer Science. 7563, 264-277.

Nardi, B.A., Zarmer, C.L. (1993). Beyond Models and Metaphors: Visual Formalism in User Interface Design. Journal of Visual Languages and Computing, 4, 5-33.

Neumann, S., Klebl, M., Griffiths, D., Hernández-Leo, D., De-La-Fuente Valentín, L., Hummel, H., Brouns, F., Derntl, M., & Oberhuemer, P. (2010). Report of the Results of an IMS Learning Design Expert Workshop. International Journal of Emerging Technologies In Learning (IJET), 5(1), 58-72.

Neumann, S., Oberhuemer, P., 2009. User Evaluation of a Graphical Modeling Tool for IMS Learning Design. Lecture Notes in Computer Science, 5686, 287-296.

Niramitranon, J. (2009). Orchestrating Learning in a one-to-one Technology Classroom. PhD Thesis. University of Nottingham.

Niramitranon, J., Sharples, M. & Greenhalgh, C. (2010). Orchestrating Learning in a one-to-one Technology Classroom. In Khine, M.S. & Saleh, I.M. (eds.) New Science of Learning: Cognition, Computers and Collaboration in Education. NY: Springer, pp. 451-468.

Palomino-Ramírez, L., Bote-Lorenzo, M., Asensio-Pérez, J., Dimitriadis, Y. (2008). LeadFlow4LD: Learning and Data Flow Composition-Based Solution for Learning Design in CSCL, in: Groupware: Design, Implementation, and Use, vol. 5411, pp. 266-280.

Paquette, G. (2010). Ontology-Based Educational Modelling – Making IMS-LD Visual. Technology, Instruction, Cognition and Learning, 7(3-4), 263-296.

Paquette, G., Léonard, M., Lundgren-Cayrol, K. (2008). The MOT+ Visual Language for Knowledge Based Instructional Design, in: L. Botturi, T. Stubbs (Eds.), Handbook of Visual Languages for Instructional Design: Theories and Practices. Information Science Reference, Hershey, pp. 133-154.

Pérez-Sanagustín, M., Burgos, J., Hernández-Leo, D., Blat, J. (2009). Considering the intrinsic constraints for groups management of TAPPS & Jigsaw CLFPs. In: Proceedings of the International Conference on Intelligent Networking and Collaborative Systems, pp. 317-322.

Prieto, L.P., Asensio-Pérez, J.I., Dimitriadis, Y.A., Gómez-Sánchez, E. & Muñoz-Cristóbal, J.A. (2011). GLUE!-PS: A Multi-language Architecture and Data Model to Deploy TEL Designs to Multiple Learning Environments. European Conference on Technology Enhanced Learning, pp. 285-298.

Prieto, L.P., Tchounikine, P., Asensio-Pérez, J.I., Sobreira, P., Dimitriadis, Y. (2014). Exploring teachers' perceptions on different CSCL script editing tools. Computers and Education. 78, 383-396.

Progress Bar. A Moodle time management tool. (moodle.org/plugins/view.php?plugin=block_progress), visited in 12/2013.

ReCourse. ReCourse Learning Design Editor (www.tencompetence-project.bolton.ac.uk/ldauthor), visited 12/2013.

RELOAD. Reusable eLearning Object Authoring & Delivery Learning Design Editor (www.reload.ac.uk/ldeditor.html), visited in 12/2012.

Rodríguez-Triana, M.J., Martínez-Monés, A., Asensio-Pérez, J.I., Dimitriadis, Y. (2014). Towards a Script-Aware Monitoring Process of Computer-Supported Collaborative Learning Scenarios. International Journal on Technology Enhanced Learning. Special issue on: Learning Analytics. (to appear).

Ronen, M., Kohen-Vacs, D. & Raz-Fogel, N. (2006). Adopt & Adapt: Structuring, Sharing and Reusing Asynchronous Collaborative Pedagogy. International Conference of the Learning Sciences, pp. 599-605.

Roschelle, J., Rafanan, K., Bhanot, R., Estrella, G., Penuel, B., Nussbaum M. & Claro, S. (2009). Scaffolding group explanation and feedback with handheld technology: impact on students' mathematics learning. Education Tech Research Dev, Springer, 58(4), pp. 399-419.

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

Sadiq, S. W., Sadiq, W. & Orlowska, M. E. (2002). Workflow driven e-learning beyond collaborative environments, in: International NAISO Congress on Networked Learning in a Global Environ-ment, Challenges and Solutions for Virtual Education, pp. 1-7.

Sakai. Sakai Project (https://sakaiproject.org), visited in 12/2013.

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

Schellens, T., Van Keer, H., De Wever, B., & Valcke, M. (2007). Scripting by assigning roles: Does it improve knowledge construction in asynchronous discussion groups? International Journal of Computer-Supported Collaborative Learning, 2(2–3), 225–246.

Schoonenboom, J. (2008). The effect of a script and an interface in grounding discussions. International Journal of Computer-Supported Collaborative Learning, 3(3), 327–341.

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

Slof, B., Erkens, G., Kirschner, P. A., Jaspers, J.G. M., & Janssen, J. (2010). Guiding students' online complex learning-task behavior through representational scripting. Computers in Human Behavior, 26 (5), 927–939.

Sobreira, P. & Tchounikine, P. (2015). Table-based representations can be used to offer easy-to-use, flexible, and adaptable learning scenario editors. Computers and Education. 80, 15-27.

Sobreira, P. & Tchounikine, P. (2013). CSCL scripts: articulating table and graph representations. Proceedings of the 10th International Conference on Computer Supported Collaborative Learning. International Society of the Learning Sciences, v.2, pp 165-168.

Sobreira, P. & Tchounikine, P. (2012). A model for flexibly editing CSCL scripts. International Journal of Computer-Supported Collaborative Learning. 7(4), 567-592.

Soller, A., Martínez-Monés, A., Jermann, P. & Muehlenbrock, M. (2005). From Mirroring to Guiding: A Review of State of the Art Technology for Supporting Collaborative Learning. International Journal of Artificial Intelligence in Education, 15 (4), 261-290.

Stahl, G. & Hesse, F. (2007). Welcome to the future. International Journal of Computer-Supported Collaborative Learning, 2(1), 1–5.

Stahl, G., Law, N., Hesse, F. (2013). Reigniting CSCL flash themes. International Journal of Computer-Supported Collaborative Learning. 8(4), 369–374.

Stegmann, K., Streng, S., Halbinger, M., Koch, J., Fischer, F., & Hussmann, H. (2009). eXtremely Simple Scripting (XSS): a framework to speed up the development of computer-supported collaboration scripts. International Conference on Computer Supported Collaborative Learning. International Society of the Learning Sciences, v.2, pp. 195-197.

Stegmann, K., Weinberger, A., & Fischer, F. (2007). Facilitating argumentative knowledge construction with computer-supported collaboration scripts. International Journal of Computer-Supported Collaborative Learning, 2(4), 421–447.

Tchounikine, P. (2011). Computer Science and Educational Software Design – A Resource for Multidisciplinary Work in Technology Enhanced Learning. Springer. DOI 10.1007/978-3-642-20003-8_6

Tchounikine, P. (2008). Operationalizing macro-scripts in CSCL technological settings. International Journal of Computer-Supported Collaborative Learning, Springer, 3(2), 193-33.

Tchounikine, P., Mørch, A.I., Bannon, L. (2009). A computer science perspective on TEL research. In: N. Balacheff, S. Ludvigsen, T. de Jong, A. Lazonder, S. Barnes (Eds) Technology-Enhanced Learning – Principles and Products. Springer, 271-284.

TENCompetence. Building the European Network for Lifelong Competence Development. (tencompetence-project.bolton.ac.uk), visited 12/2013

Trimmer, K. (2006). Teacher ICT Skills: Evaluation of the Information and Communication Technology Knowledge and Skill Levels of Western Australian Government School Teachers. Autralian Evaluation Society International Conference, Darwin, Australia. (retrieved from http://www.aes.asn.au, April 2012).

Van de Velde, W., Häkkinen, P., Järvela, S., Stegmann, K. (2004). Examples of CSCL scripts within an organisational framework. WP 23 – Mobile Support for Integrate Learning. Kaleidoscope (JEIRP MOSIL). Deliverable 23.3.1. 52p.

Vantroys, T., Peter, Y. (2003). COW: A flexible platform for the enactment of learning scenarios, in: International Workshop on Groupware, Lecture Notes on Computer Science, Springer-Verlag, 168-182.

Villasclaras-Fernández, E. D., Hernández-Gonzalo, J. A., Hernández-Leo, D., Asensio-Pérez, J. I., Dimitriadis, Y., Martínez-Monés, A. (2009). InstanceCollage: a tool for the particularization of collaborative IMS-LD scripts. Educational Technology & Society, 12(4) pp. 56–70.

Villasclaras-Fernández, E., Hernández-Leo, D., 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.

Wecker, C., Stegmann, K., Bernstein, F., Huber, M.J., Kalus, G., Kollar, I., Rathmayer, S. & Fischer, F. (2010) S-COL: A Copernican turn for the development of flexibly reusable collaboration scripts. International Journal of Computer-Supported Collaborative Learning, Springer New York, 5, 321-343.

Weinberger, A., Ertl, B., Fischer, F., & Mandl, H. (2005). Epistemic and social scripts in computer-supported collaborative learning. Instructional Science, 33(1), 1–30.

Weinberger, A., Kollar, I., Dimitriadis, Y., Mäkitalo-Siegl, K., & Fischer, F. (2009). Computer-supported collaboration scripts: Theory and practice of scripting CSCL. In N. Balacheff, S. Ludvigsen, T. de Jong, A. Lazonder, S. Barnes & L. Montandon (Eds.), Technology-Enhanced Learning. Principles and Products: Berlin:Springer, 155-174.

Weinberger, A., Stegmann, K., & Fischer, F. (2010). Learning to argue online: Scripted groups surpass individuals (unscripted groups do not). Computers in Human Behavior, 26, 506–515.

Williams, D.L., Boone, R., & Kingsley, K.V. (2004). Teacher beliefs about educational software: A Delphi study. Journal of Research on Technology in Education, 36(3), 213-229.