LeadFlow4LD: Learning and Data Flow Composition-Based Solution for Learning Design in CSCL

Luis Palomino-Ramírez, Miguel L. Bote-Lorenzo, Juan I. Asensio-Pérez, and Yannis A. Dimitriadis

School of Telecommunication Engineering, University of Valladolid, Camino del Cementerio s/n, 47011 Valladolid, Spain
lpalomin@ulises.tel.uva.es, {migbot,juaase,yannis}@tel.uva.es

Abstract. IMS-LD is the de facto standard for learning design (LD) specification which typically comprises an activity flow and a data flow. Nevertheless, the specification of the data flow between tools is an open issue in IMS-LD, especially in collaborative learning. In such case, handling shared data derived from individual and collaborative tools is error-prone for learners who suffer an extra cognitive load. Additionally, problems in the collaborative data flow specification affect the reusability of the whole learning design. In this paper, we present LeadFlow4LD, a solution of specification and enactment for LD in CSCL in order to address the aforementioned issues in an interoperable and standard way. Such a solution is based on approaches for the composition of the activity flow specified in IMS-LD and the data flow specified in a standard workflow language, such as BPEL. An architecture and a prototype for validating the propose solution through a case study based on a significant CSCL situation are also presented.

Keywords: Data Flow, Learning Design, Workflow, CSCL, IMS-LD.

1 Introduction

The LD approach [1] has evidenced and promoted a major shift in technology enhanced learning, since it pays special attention to the process of teaching and learning, instead of the previous approach to the delivery of educative contents, as learning objects. Thus, according to LD, a learning design formally defines a sequence of learning activities in which students and teachers play roles, individually or in groups, through the use of tools and services with the aim of accomplishing their learning goals [2]. With the aim of providing interoperability, the IMS-LD language [1, 3] has come up as the de facto standard in LD for a variety of pedagogical approaches including collaborative learning [4], although there are other competing non-standard Educational Modelling Languages [5].

LD shares several common features with workflow system [1], since it deals with the coordination of activities. Nevertheless, a fluent design and enactment of a learning design by the users (learners, tutors) of an e-course requires a major automated

R.O. Briggs et al. (Eds.): CRIWG 2008, LNCS 5411, pp. 266–280, 2008.

support with regard to the sequence of learning activities (the learning flow) and the flow of information (the data flow), as in document-oriented workflow systems [6]. However, the specification of data flow between tools is still an open issue for IMS-LD [7] [8] [9] [10] [11] [12]. The main problem lies in the fact that IMS-LD supports a human-oriented data flow approach [11], which means that the user, not the system, is responsible for managing the data flow between tools.

Although this human-oriented data flow approach may be partially valid for individual learning, serious drawbacks have been identified in CSCL situations [11] [12]. Due to the complex interactions between users and data, locating and handling shared data in different group structures (see e.g. the expert and super groups in a situation based on the jigsaw Collaborative Learning Flow Pattern (CLFP) [3]) is error-prone for users who suffer an additional cognitive load. Moreover, the reusability issue in collaborative data flow specification becomes essential [12].

As a response to these issues, LD and Workflow may be considered and exploited as similar but at the same time as complementary approaches supported by the corresponding technologies [11], due to their focus on human activities and system tasks, respectively. On the one hand, by delegating the responsibility of managing the shared data to a workflow-based solution (design formalization and enactment engine), it would be feasible to eliminate error-prone situations for users as well as to eliminate the excess of their cognitive load. On the other hand, by separating the declarative-level specification from the instance-level one, the reuse of both the learning and data flow specifications could be facilitated. Therefore, this so-called composition-based approach requires the coordinated execution of the learning flow specified in IMS-D and the data flow specified in a standard workflow language [11]. However, a concrete solution based on this approach is still missing [11] [12], due to the existence of several workflow standards and the complexity of the proposed approach. Such a concrete solution, the implementation of a prototype and its validation through significant case studies may enable a deeper understanding of the approach and foster a shared standards-based solution by the wider community.

In this paper, we present LeadFlow4LD (LEarning And Data FLOW composition-based solution FOR Learning Design) as a solution for addressing all the aforementioned issues related to the data flow problem of IMS-LD in CSCL situations. LeadFlow4LD comprises approaches for the learning flow specification in IMS-LD and the data flow specification in the selected workflow language. Furthermore, it defines an approach for the coordination of both streams, as well as a logical architecture for the enactment of such learning designs. Moreover, in order to validate the proposed solution, a prototype of the enactment architecture has been implemented, while distinct configurations of a significant case study have been carried out.

The paper is structured as follows. In section 2 the data flow problem of IMS-LD in collaborative learning through a case study is presented. In section 3 we describe LeadFlow4LD as a technology independent approach. Besides, the implementation of a prototype for the enactment of learning designs specified according to LeadFlow4LD is also described. In section 4 the evaluation of the proposed solution through different configurations of the case study is presented, while in section 5 the related work to the IMS-LD data flow problem is reviewed. Finally, in section 6 we present our conclusions and future work.

2 The Data Flow Problem of IMS-LD in Collaborative Learning

As already known, tools and services used in individual or collaborative activities may need data as input or output. Such data artifacts can be, for example, the conclusions of a discussion, the answers of a test, the specification data for a simulator, etc. Since these tools may span through various activities or be handled by different actors in the same or different activities, a data flow is generated that is related to the learning flow. However, such data flow cannot be specified through the mechanisms provided by IMS-LD [8]. Instead, since IMS-LD follows a human-oriented approach, the participants of the learning process should be responsible for handling this data flow [11].

In this section we are going to illustrate the data flow problem in collaborative learning settings, which has been documented in literature [12], through a case study that is both significant and authentic. Then, the presentation of the proposed solution can be validated on this case study in a later section, and subsequently support and shed light to the posterior discussion on advantages and limitations.

The case study, called *CNS2* (Collaborative Network Simulator 2) and illustrated as an activity diagram in Fig. 1a, corresponds to a collaborative learning scenario in which learners perform different activities based mainly on the well-known network simulator *ns-2* [13], in order to evaluate and analyze different network protocols.

In the first activity, a simulation *Tcl/Tk* script (D1) is generated individually by each learner using a generic editor. Then, the generated *Tcl/Tk* script becomes an input to the *ns-2* tool during the simulation (second) activity. As a result, two types of

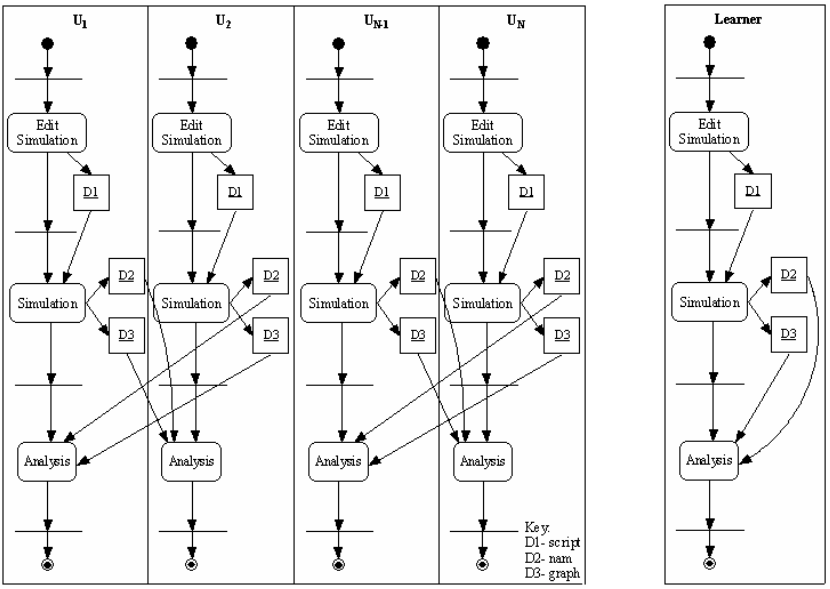


Fig. 1. Activity diagram for the case study CNS2 (a) at instance-level (b) at declarative-level (not sufficient)

output files are generated, named the *graph* (D2) and *nam* (D3) files. The first file contains different measures taken during the simulation that can be plotted as *x-y* graphs, while the second one contains the behavior of different elements defined in the scenario that can be visualized using a network animator tool [14]. Finally, the simulation analysis activity is carried out, in a collaborative way, since each learner analyzes the simulation results of one of his peers. For example, the user *u1* should analyze the *graph* and *nam* data files that belong to the user *u2*, and so on.

In IMS-LD, such collaborative data flow specification is carried out through a verbal (textual) instruction to each user included in the activity description regarding the path where the right artifacts are located. This is clearly an error-prone approach for users [12], since the instructions could be misunderstood, badly applied, ignored or forgotten by the users [15]. The error-prone character of this approach reveals an important issue, since the learning objectives may not be accomplished as expected. Furthermore, this approach produces an additional cognitive load to users, who should be concerned about understanding well the instructions, locating and retrieving the right artifact, instead of an automatic selection, retrieval and delivery by the system [12].

In other terms, in order to distinguish each user data, instance-level data flow specification (see Fig. 1a) within the learning design is necessary [12]. Then, data flow design cannot be specified at declarative-level (see Fig. 1b) and therefore it cannot be used several times for different instances of users and CSCL situations [16]. Therefore, the instance-level collaborative data flow specification affects the reusability of the whole learning design, which is also a relevant issue [12].

In order to address the issues illustrated in the above case study, while keeping interoperability with IMS-LD, LeadFlow4LD is presented in the next section, which is a composition-based solution in which the learning flow is specified in IMS-LD while the data flow is specified in a standard workflow language.

3 The Proposed Solution: LeadFlow4LD

3.1 Overview

An overview of the proposed solution is illustrated in Fig. 2. As we can see, LeadFlow4LD consists of two main approaches. The first approach is related to the learning design definition which conforms to LeadFlow4LD, while the second one is related to the enactment of such learning designs. On the other hand the learning design definition consists of a declarative-level definition and an instance-level one for both learning and data flow. This separation allows the learning design definition be reused for different contexts and situations [16], so as the learning and data flow definitions can be reused for different users, groups and shared data logic. In addition, a coordination definition is also proposed in order to enact coordinately both streams.

Although this paper focuses on the approach of learning design definition, enactment and validation, it is also considered for completeness. Moreover, we pay attention to a technology independent approach which describes how the distinct abstract components interact with the learning design (see Fig. 3). Even though the validation of the approach through a prototype is illustrated through concrete technologies, the detailed specifications for each technology are out of scope of this paper.

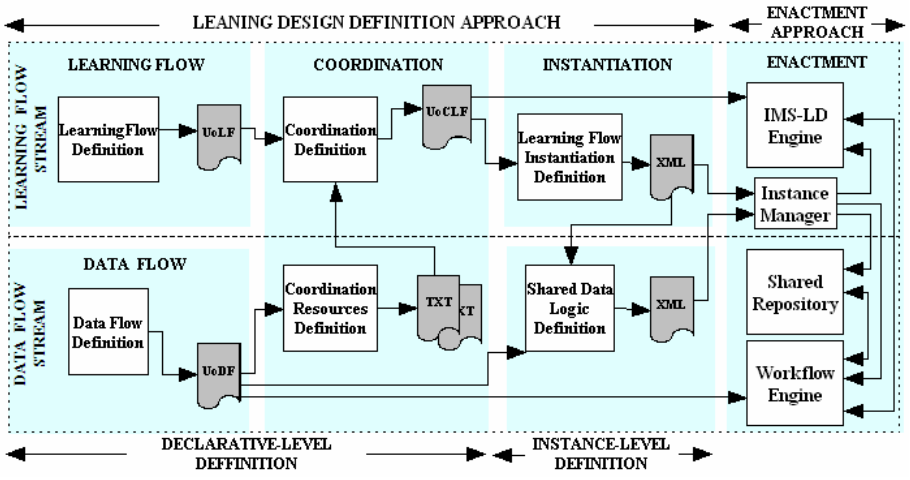


Fig. 2. Overview of LeadFlow4LD

3.2 Learning Design Definition Approach

Overview. A learning design which conforms to LeaDFlow4LD consists of different definitions as we will describe throughout the next subsections. For a better understanding of the approach, an overview of the interactions among the involved elements is illustrated in Fig. 3, which are also referenced throughout these subsections.

Defining a list of tools. The learning flow defines the sequence of learning activities, meanwhile the data flow defines the sequence of tool invocations and the data flow among them (see Fig. 3). Nevertheless, in order to keep interoperability with IMS-LD, LeadFlow4LD does not demand that all tools must be called upon in the data flow. Instead, tools can also be defined as resources in the learning flow, or even not be defined at all, which means that the instructional designer does not provide tools support to the activity, but the learners employs their own tools. In such cases, output data from these tools may require to be imported into the data flow, or data defined in the data flow may require be exported outside it. Therefore, the learning and data flow definitions are not disjoint at all, but it must be previously known whether the tools will be defined in the learning flow or called upon in the data flow in order to define separately both streams. With this purpose, the definition of a list of tools as a previous step for a learning design definition is necessary.

Learning flow definition. Since IMS-LD fails in data flow specification in collaborative learning [1], LeadFlow4LD proposes that it should only be used for the learning flow definition, but not for the data flow one, which will be specified separately in a standard workflow language. Nevertheless, the aim of the learning flow definition approach is not to describe how this sequence of learning activities in collaborative learning should be specified with IMS-LD, since it has already been worked in literature [2], but to describe the implications in the learning flow definition due to the

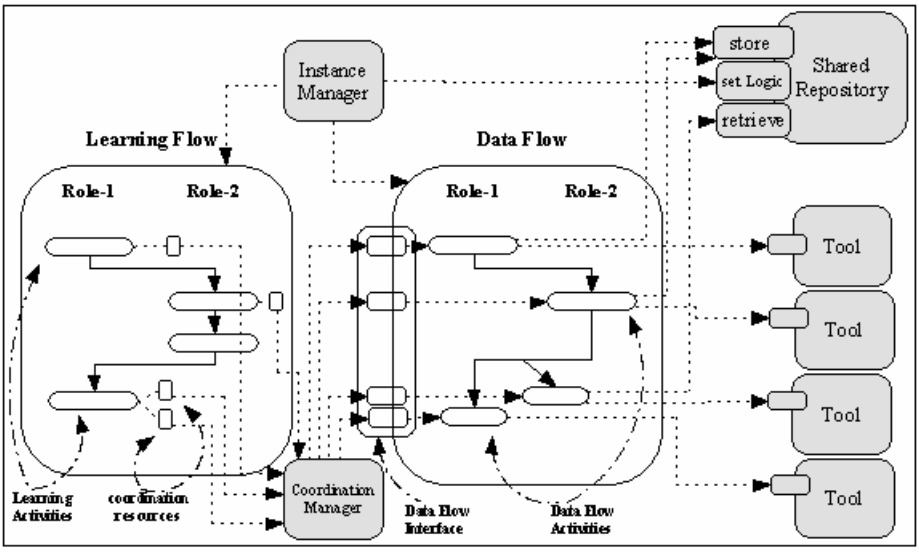


Fig. 3. Overview of a learning design which conforms to LeadFlow4LD. In the background white the learning design-related components are represented, while in background gray the enactment-related ones can be found.

separation of both streams. With this purpose, LeadFlow4LD demands that IMS-LD properties are not used for artifacts and users specification, since the learning flow definition should focus on the flow of activities at declarative-level but not at the instance-level. Moreover, it should also focus on defining activities-related resources including tool definitions, as well as activity descriptions. Finally, as shown in Fig. 2, the resulting learning design definition must be packaged as unit of learning (UoL) according to the standard IMS-CP [3]. In our case, it is denominated unit of learning flow (UoLF) in order to emphasize the fact that it is just focused on the flow of learning activities.

Data flow definition. The aim of the data flow definition is to sequence the invocation of tools and to manage automatically the data flow among them. Although the data flow in collaborative learning requires an instance-level specification [1], since users need to share their data with other users belonging to the same or different roles (groups), the data flow must be defined first at a declarative-level in the same way as the learning flow is defined. Therefore, the approach to define the data flow considers aspects related to the sequence of invocations of tools (known here as sequence of data flow activities); the data flow interface; and the roles in which such data flow activities are associated. First, the sequence of data flow activities is related to the sequence of learning activities in which the tools have been called upon (see Fig. 3). Nevertheless, both of them do not have to match necessarily the same sequence, since some learning activities may not be associated with tools or more than one, in which case these tools should be sequenced concurrently in the data flow. Secondly, the data flow actually performs as an indirection layer, since while users perform learning activities, they are supported by tools which could be invoked through the data flow

interface. Thirdly, LeadFlow4LD demands that the data flow activities must be associated to the same roles than to the learning activities in which the tools are called upon. As it is shown in Fig. 2, the resulting workflow definition of the data flow must be packaged as what we call unit of data flow (UoDF) in order to emphasize the data flow issue. Finally, in order to define the instance-level data flow specification, LeadFlow4LD proposes a mechanism to define in instantiation time what we call the shared data logic definition, which will be treated later in this paper.

Coordination definition. Once that learning and data flow have been defined separately, the coordinated execution of both streams is necessary. For this purpose, LeadFlow4LD proposes a synchronous master-slave coordination model: while the learning flow is the master stream, the data flow is the slave. That is, while the data flow invokes one or more tools, the learning flow remains blocked, so users cannot move on to the next learning activity, until the tasks carried out by the invoked tools have been finished. With this aim, the so-called coordination resources are defined, and these are added to the learning activities as IMS-LD resources in order to invoke the data flow interface, either to invoke tools, import or export data from/to the data flow (see Fig. 3). Finally, the resulting coordination definition must be packaged as a UoL according again to the standard IMS-CP. In our case, it is denominated unit of coordinated learning flow (UoCLF) in order to emphasize the fact that it is actually a UoLF which has been completed with the so-called coordination resources.

Defining instantiation. Once the learning design which conforms to LeadFlow4LD has been defined at a declarative-level, the instantiation defines the context in which it will be carried out. This context spans both the learning flow and the data flow. On one hand, the instantiation of the learning flow defines groups, users, assignment of users to groups and users to roles [4]. On the other hand, LeadFlow4LD proposes that the instantiation of the data flow defines the shared-data logic among users belonging to the same or distinct roles (groups). Through this approach to define the instantiation, the learning design definition can be reused for different users, groups, group's size and what is our contribution: for different shared data logic. Finally, both instantiations are defined using XML schemas so as to be interpreted by an instance manager, which in turn will interact with the proper enactment engine (see Fig. 2). Next, a description of our proposal for the shared data logic definition is presented.

Shared data logic definition. In order to separate the declarative-level data flow definition from its instantiation, it is necessary to have a shared data logic definition into an external component. For this purpose, a shared repository could be used to store users' data in order to be retrieved later for the same or other users according to the shared data logic defined at instantiation time.

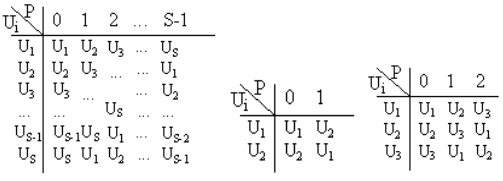
Suppose that S users are part of the same collaborative group (role): $\{u1,u2,..us\}$. In order to define the shared data logic between them, LeadFlow4LD defines the tuple:

$$L\ (D,\ S,\ P)$$

Where:
L- The shared data logic
D- The shared (declarative) data
S- The group's size
P- The peer, who data is accessible for the user ui, $0<=P<S$.

For example, the shared data logic: *L(D,S=2,P=1)* defines a typical peer sharing logic. That is, working in pairs, each user gets access to the data belonging to his unique peer. Now consider the shared data logic defined by *L(D, S=3,P=1)*, which means that in a group of three people, each user has access to the data belongs to his first peer (in a triplet a user has two peers). Finally, consider the case when the shared data logic is defined by a null peer: *L(D,S,P=0)*, which means that there are not shared data at all, because each user only has access to his own data, which is the case of individual learning.

Nevertheless, data within the shared repository are stored and therefore retrieved regarding each user, not to the number of the peer. So, in order to retrieve data regarding to some user *ui* according to his peer P defined in the shared data logic, an evaluation of the matrix function *getUser(ui,S,P)* shown in Fig. 4 is necessary. Note that particular cases for *S=2* and *S=3* are shown in Fig. 4b and 4c respectively.



(a) General case (b) S=2 (c) S=3

Fig. 4. Matrix function *getUser(ui,S,P)* used to get the user who shares his data with the user *ui* according to the defined shared data logic *L(D,S,P)*

For example, the shared data logic definition given by *L(D1,S=3, P=2)*, means that the user *u1* must get access to the retrieved data from the shared repository according to the tuple *(u3, D1)*, since *u3* is the result of evaluating the function *getUser(u1,S=3,P=2)* shown in Fig. 4c. Although LeadFlow4LD requires a shared data logic definition among users from the same or different roles, the current approach only covers the first, so an extension of the current approach is necessary.

3.3 Architecture for the LeadFlow4LD Run-Time Environment

The enactment of a learning design which conforms to LeadFlow4LD requires the integration of different components and specifications (see Fig. 3). Taking advantage from service-orientation, we used services as the basic components of service oriented architectures (SOA) [5]. In this context, we have proposed a three-tier logical architecture for the enactment of such learning designs as illustrated in Fig. 5. So, once a UoCLF and its corresponding UoDF have been defined and stored in the proper repository, the learning flow and data flow instance manager and their clients are responsible for deploying, instantiating, monitoring and terminating the enactment of learning design instances (both learning flow and data flow instances). Beside this, the data flow instance manager client is also responsible for setting the shared data logic through the shared repository service. Then, the learning flow engine is responsible for playing the active learning flow, while its client, the so-called enactment client indicates the proper activity that should be carried out by each user, as well as

the corresponding resources. Finally, when a user invokes a service (either an importing or exporting data service), the client manager provides the client with the proper service to invoke operations regarding to the data flow service. In the next section a prototype based on this architecture is implemented.

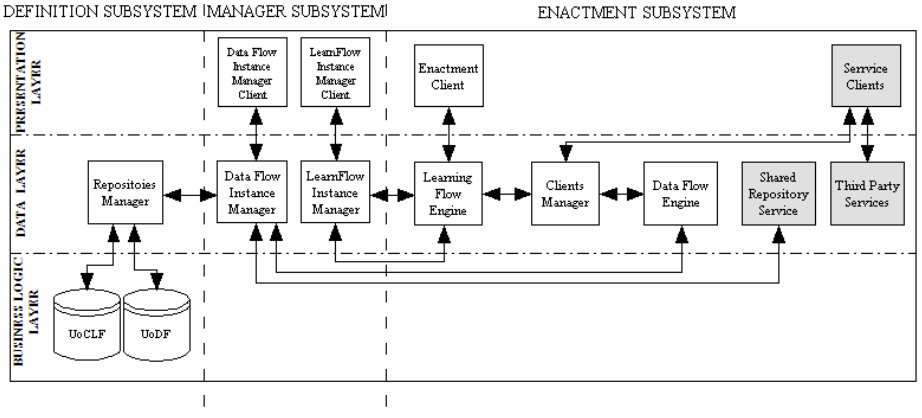


Fig. 5. Logical architecture for the LeadFlow4LD run-time environment. The external elements are represented in gray background.

3.4 Prototype Implementation

One of the main decisions related to the implementation of the prototype refers to the selection of BPEL [6] as the workflow language for the data flow specification. Although this is not the only choice, BPEL seems to be an adequate one for several reasons; It is a commonly accepted standard widely used by the industry and especially for the composition, orchestration and coordination of web services, a popular technology based on the service orientation, thus fostering interoperability and integration [7]. Furthermore, specifying the sequence of service invocations and the data flow among them can be easily implemented with BPEL through appropriate mechanisms.

Current prototype implements the following elements that correspond to the logical architecture shown in Fig. 5: the enactment client, the learning flow engine, the data flow engine and the client manager. On one side, *Coppercore* [8] has been used to implement the learning flow engine, whereas *WebPlayer*, distributed together with *Coppercore*, has been used to implement the enactment client. On the other side, *ActiveBPEL* [9] has been used to implement the data flow engine, whereas the *JNLP (Java Network Launch Protocol)* application manager of *Java Web Start* has been used to implement the client manager. *JNLP* [10] is a standard specification that allows Java applications to be automatically launched from an application server in order to be executed locally in the client machine. Therefore, during the coordination definition, the so-called coordination resources are specified conform to the *JNLP* specification. The files that correspond to the coordination resources contain the address of the application server where the resources required for the execution of the proper application are located. Then, these files are added as resources to the activities into the learning flow and as result a *UoCLF* is defined. Finally, the shared repository

service was implemented as a web service using *MySQL* data base management system to storage the state information.

4 Validation of the Proposed Solution through the Case Study

In this section we present preliminary results regarding the validation of the proposed approach that are illustrated through the case study *CNS2* presented in section 2. Table 1 shows the specific information provided by the designer with regard to the specific tools used in this CSCL scenario. Thus, it can be seen that e.g. the *Tcl/Tk* editor tool has been specified as an IMS-LD resource to be employed in the simulation edition activity. On the contrary, the *ns-2* tool will be invoked as a third-party web service, while the network animator and *x-y* plotter have not been prescribed by the designer, thus it allows the users to choose the most appropriate ones.

Table 1. Tools, data and activities for the case study *CNS2*

Tool	Specified as	Input data	Output data	Activity
Tcl-Tk Editor	Resource in *IMS-LD*	-	Sim. script	Edit simulation
ns-2	Third-party service	Sim. script	*nam, graph*	Simulation
Network Animator	-	*nam*	-	Sim. analysis
x-y Plotter	-	*graph*	-	Sim. analysis

An overview of the learning design which conforms to LeadFlow4LD for the case study *CNS2* is shown in Fig. 6. The prototype has been tested in several configurations, related to number and size of groups, as well as data sharing logic. Therefore, we could determine the flexibility of the proposed approach and the current prototype, especially with regard to the complex characteristics of collaborative learning.

The snapshot shown in Fig. 7 presents an activity description from the user perspective. Note that, even in this simple peer sharing case, the user's cognitive load is reduced notably, since he is not concerned in handling adequately data flow according to instructions. Instead, a click in the corresponding input of the system environment window is sufficient.

On the other hand, in Fig. 8, we can see what error-prone situations are avoided, due to the automatic delivery of the artifacts, when the user *u1* invokes the service for downloading (exporting) the *nam* file regarding to his partner defined in the shared data logic *(u2)*.

Reusability of the whole learning design at the definition level has been shown through the evaluation of several runs in different situations, corresponding to different users, number of groups, group's size or even the shared data logic. In all runs, except one, no changes were necessary with regard to the learning design definition (both learning and data flow), which shows that LeadFlow4LD fosters reusability. However, in the case of learners who have to access the data of two peers at a time, instead of one, a change in the learning and data flow definitions were necessary, since current shared data logic definition approach is limited to users having access to the data of one peer at a time. Even in this case, such change could be easily made in the learning and data flow definitions but an extension of the shared data logic mechanism is necessary.

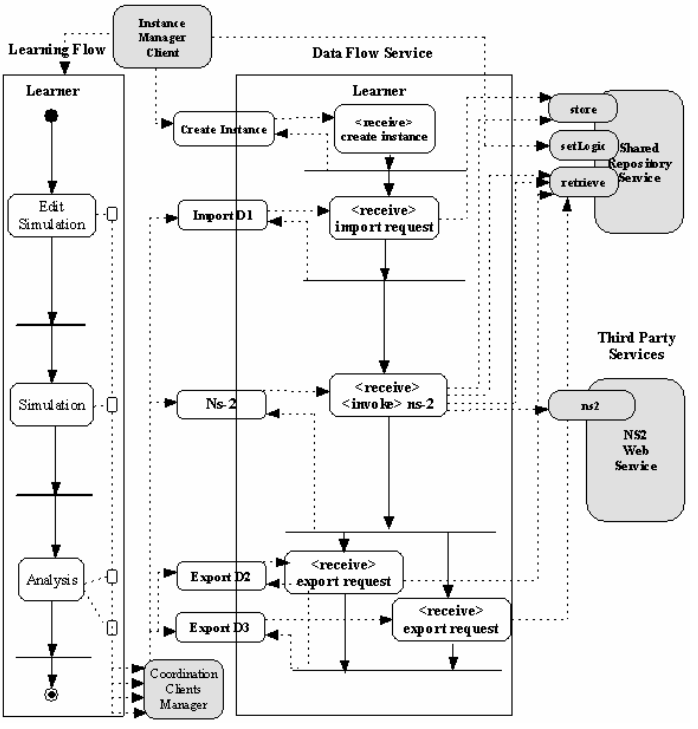


Fig. 6. Overview of the learning design which conforms to *LeadFlow4LD* of the case study *CNS2*. (White background and gray background denote design components, and enactment components respectively.)

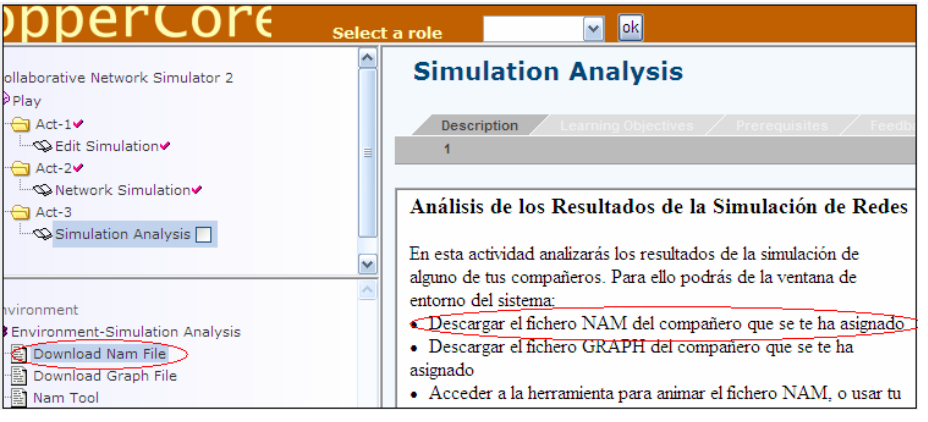


Fig. 7. Snapshot of the *WebPlayer* showing the activity description (center frame) and the system environment window (bottom left frame)

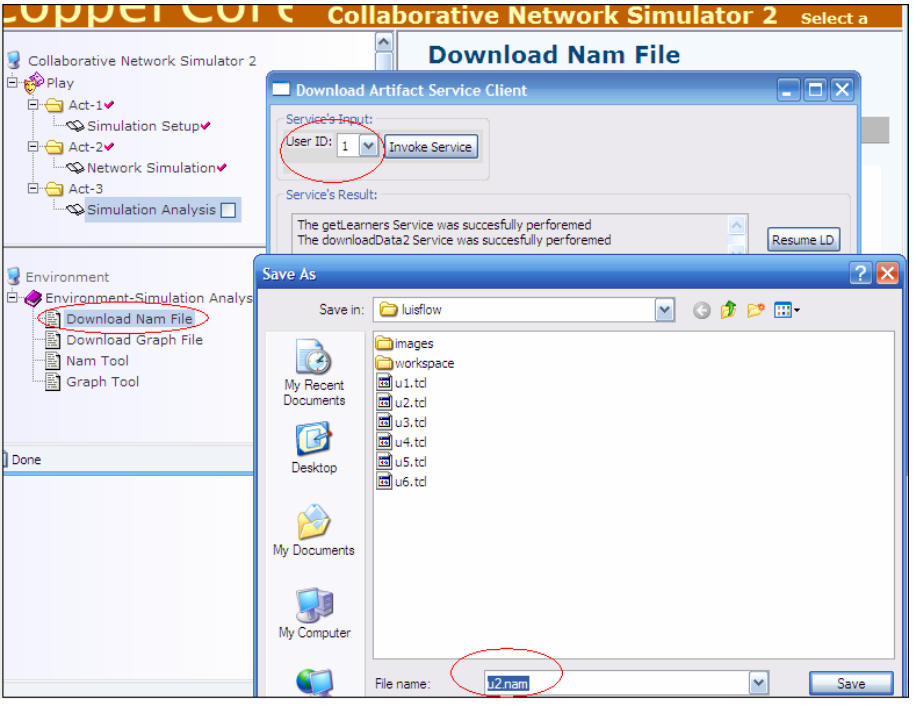


Fig. 8. Snapshot of *WebPlayer* for showing that the *nam* artifact regarding to the assigned peer is retrieved automatically by the workflow engine

5 Related Work

The data flow problem of IMS-LD has been dealt in many ways for different authors in literature. Peter and Vantroys [11] state that IMS-LD lacks of data flow management, but do not go further in defining the associated defects. Wilson criticizes in [12] IMS-LD since it does not consider whether the results of a service are going to be exported to other services; however, he does not provide any solution to this issue. Dalziel states in [13] that IMS-LD requires mechanisms in order to pass information between tools with a possible information processing between them, but the proper mechanism for data flow between tools is not proposed. Furthermore, Miao et al. mention in [14] that IMS-LD has no means to specify the relation between data and tools, and therefore they propose a new scripting language. Nevertheless, this is not an interoperable solution as LeadFlow4LD is. Moreover, they do not consider the global data flow problem in collaborative learning [1] since they mainly address a data flow automation issue. Vantroys and Peters, propose in [15] a mapping approach between IMS-LD and the standard workflow XPDL, but according to [16] both of them are complementary approaches, and therefore a mapping mechanism between them is not enough in order to address the data flow problem. According to [16], a solution space is described in order to resolve the data flow problem of IMS-LD, that considers the substitution, mapping and composition approaches. However, in [16],

no concrete solutions and results are provided with regard to the composition-based approach based on learning design and workflow standards. On the other hand, the initial proposal for the coordination mechanism is based on a third independent stream, i.e. a Petri Net stream. However, the two-streams master-slave coordination mechanism proposed in LeadFlow4LD is simpler to implement than the 3-streams coordination mechanism proposed in [16]. Finally, in [1] a case study in order to evaluate the data flow problem of IMS-LD in collaborative learning is presented but a concrete solution to the problem is missing.

6 Conclusions and Future Work

IMS-LD, as the de facto standard for learning design supports a user-oriented mechanism for data management and especially handling input and output data related to tools. Nevertheless, this approach has special serious drawbacks in CSCL situations. Due to the need of sharing data among users, learners suffer a high cognitive load as well as error-prone situations when they locate and handle data from other users that may potentially affect the accomplishment of the learning objectives Furthermore, another relevant issue refers to reusability of the whole learning design since the shared data logic cannot be specified at declarative level, but instance-level data flow have to be specified.

In this paper we propose an interoperable solution to this problem, called LeadFlow4LD, which consists of approaches for the specification of both learning and data flows, using respectively IMS-LD and a standard data flow workflow language. Furthermore, an approach for coordination of both streams has been also presented. Moreover, we have presented an architecture related to the run-time environment, necessary to enact learning designs which conforms to LeadFlow4LD. We have also implemented a prototype based on this architecture and several runs of the same case study have been carried out. Results indicate that LeadFlow4LD solves all the issues mentioned in this paper about the data flow problem of IMS-LD in CSCL situations. On one side, by delegating the responsibility for shared data management to the workflow engine, a learner is not responsible for locating and handling the shared data, therefore reducing error-prone situations and the associated cognitive load. On the other side, through the separation of the declarative-level learning design definition from its instance-level definition the learning design can be reused for distinct contexts including users, groups, groups'size or even the shared data logic. For this purpose, we have proposed a mechanism to define the shared data logic in an independent way. This approach fosters reusability of the data flow and consequently of the whole learning design. Finally, it is interesting to see LeaDFlow4LD as the in between approach of a future integration of learning design and workflow streams.

Future work covers several issues. Firstly, an extension of the shared data logic mechanism is necessary so as to include shared data among users belonging to different roles as well as a user having to access the data of more than one peer at a time, and therefore it should be explored and incorporated in LeadFlow4LD. Evaluation needs to go further, than the significant *CNS2* case study presented in this paper. More complex case studies have to be carried out and evaluated from a technological and educational perspective in authentic environments, in order to identify advantages and drawbacks

of the proposed approach. Of course a complete adoption of LeadFlow4LD requires the development of authoring tools that support its use by educational practitioners in the same direction as Collage [17]. Also, integration of LeadFlow4LD with tailorable service-oriented educative systems, such as Gridcole [18], is currently under development, as well as the generation of specific documents for the guidelines of the distinct approaches of LeadFlow4LD regarding to the selected technologies.

Acknowledgments. This work has been partially funded by the EU Kaleidoscope NoE FP6-2002-IST-507838, Spanish Ministry of Education and Science project TSI2005-08225-C07-04, Autonomous Government of Castilla y León, Spain (projects VA009A05, UV46/04 and UV31/04), Tecnológico de Monterrey Campus Guadalajara, and Fundación Carolina. The authors would also like to thank Henar Muñoz Frutos as well as the rest of EMIC/GSIC research group at the University of Valladolid for their support and ideas to this work.

References

1. Palomino-Ramírez, L., Bote-Lorenzo, M.L., Asensio-Pérez, J.I., Dimitriadis, Y., de la Fuente-Valentín, L.: The Data Flow Problem in Learning Design: A Case Study. In: 6th International Conference on Networked Learning (NLC 2008), Halkidiki, Greece (2008)
2. Hernández-Leo, D., Asensio-Pérez, J.I., Dimitriadis, Y.: Computational Representation of Collaborative Learning Flow Patterns using IMS Learning Design. Journal of Educational Technology & Society 8, 75–89 (2005)
3. IMS, C.P.: IMS Content Packaging Specification (2003)
4. Hernández-Gonzalo, J.A., Villasclaras-Fernández, E.D., Hernández-Leo, D., Asensio-Pérez, J.I., Dimitriadis, Y.A.: InstanceCollage: a Graphical Tool for the Particularization of Role/group Structures in Pattern-based IMS-LD Collaborative Scripts. In: Proceedings of the 8th International Conference on Advanced Learning Technologies (ICALT 2008), Santander, Spain, pp. 1–5 (2008)
5. Papazoglou, M.P., Georgakopoulos, D.: Service-Oriented Computing. Communications of the ACM 46, 25–28 (2003)
6. Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S.: BPEL4WS Specification: Business Process Execution Language for Web Services Version 1.1 (2003)
7. Juric, M.B., Mathew, B., Sarang, P.: Business Process Execution Language for Web Services. Pakt (2004)
8. Vogten, H., Martens, H.: CopperCore 2.2. 4. Heerlen: Open University of The Netherlands (retrieved on August 9, 2005), `http://www.coppercore.org`
9. ActiveBpel, L.L.C.: ActiveBPEL, the Open Source BPEL Engine, `http://www.activevos.com/community-open-source.php`
10. Schmidt, R.: Java Network Launching Protocol (JNLP) Specification v1.0.1 (2001)
11. Peter, Y., Vantroys, T.: Platform Support for Pedagogical Scenarios. Journal of Educational Technology & Society 8, 122–137 (2005)
12. Wilson, S.: Workflow and web services. CETIS White paper (2005) (Last retrieved January 22, 2007), `http://www.e-framework.org/resources/SOAandWorkflow2.pdf`

13. Dalziel, J.R.: Lessons from LAMS for IMS Learning Design. In: Proceedings of the 6th International Conference on Advanced Learning Technologies (ICALT 2006), Kerkrade, The Netherlands, pp. 1101–1102 (2006)
14. Miao, Y., Hoeksema, K., Hoppe, H.U., Harrer, A.: CSCL scripts: modelling features and potential use. In: Proceedings of the International Conference on Computer Support for Collaborative Learning (CSCL 2005): the next 10 years!, Taipei, Taiwan, pp. 423–432 (2005)
15. Vantroys, T., Peter, Y.: COW, a Flexible Platform for the Enactment of Learning Scenarios. In: Favela, J., Decouchant, D. (eds.) CRIWG 2003. LNCS, vol. 2806, pp. 168–182. Springer, Heidelberg (2003)
16. Palomino-Ramírez, L., Martínez-Monés, A., Bote-Lorenzo, M.L., Asensio-Pérez, J.I., Dimitriadis, Y.A.: Data Flow between Tools: Towards a Composition-Based Solution for Learning Design. In: Proceedings of the 7th International Conference on Advanced Learning Technologies (ICALT 2007), Niigata, Japan, pp. 354–358 (2007)
17. Hernández-Leo, D., Villasclaras-Fernández, E.D., Asensio-Pérez, J.I., Dimitriadis, Y., Jorrín-Abellán, I.M., Ruiz-Requies, I., Rubia-Avi, B.: COLLAGE: A collaborative Learning Design editor based on patterns. Journal of Educational Technology & Society 9, 58–71 (2006)
18. Bote-Lorenzo, M.L., Gómez-Sánchez, E., Vega-Gorgojo, G., Dimitriadis, Y., Asensio-Pérez, J.I., Jorrín-Abellán, I.M.: Gridcole: a tailorable grid service based system that supports scripted collaborative learning. Computers & Education 51(1), 155–172 (2008)