

**CAN IMS LEARNING DESIGN BE USED TO MODEL
COMPUTER-BASED EDUCATIONAL GAMES?**

**¿PUEDE UTILIZARSE IMS LEARNING DESIGN PARA MODELAR
JUEGOS EDUCATIVOS DE ORDENADOR?**

Daniel Burgos, Colin Tattersall and Rob Koper
Educational Technology Expertise Centre (OTEC)
Open University of the Netherlands
Valkenburgerweg 177; PO Box 2960
6401 DL Heerlen The Netherlands
dbu;cta;rkp@ou.nl

Abstract

IMS Learning Design (IMS LD, 2003) is a pedagogically expressive specification to model Units of Learning (UoL). IMS LD's generality implies that it should also be capable of modelling computer-based educational games. Issues on synchronization, grouping, multi-role playing, timing, collaborative working, dynamic feedback, adaptive learning and more are implicit or explicit in educational gaming and can be approached with it. Also, a framework must be defined to describe what is considered as an educational game. Many games, both commercial and free, are described as educational ones yet the educational character of a game depends on the objectives, the context and the moment when it is run. In order to support this definition, we will establish a classification of educational games that we will use to define their relationship with IMS LD, as well. Furthermore, this article will provide some answers to general questions about IMS LD coming from a broad study of a concrete field, as educational games is.

Keywords: IMS Learning Design, educational game, Unit of Learning

Resumen

IMS Learning Design es una especificación para modelar pedagógicamente Unidades de Aprendizaje. La teoría general sobre IMS LD insinúa la posibilidad de ser utilizada para modelar juegos educativos de ordenador. Temas como sincronización, agrupamiento, ejecución multirol, control del tiempo, trabajo colaborativo, retroalimentación dinámica, aprendizaje adaptativo, entre otros, se encuentran implícita o explícitamente en la realización de juegos educativos y pueden ser tenidos en cuenta dentro de IMS LD.

Palabras clave: IMS Learning Design, juego educativo, Unidad de Aprendizaje

1. Short introduction to IMS Learning Design

IMS Learning Design [1] is structured in three levels: A, B and C. With a detailed description of roles, activities, environments, methods (Level A), properties, global elements, monitoring services and conditions (Level B), and notifications (Level C), IMS LD is able to translate regular lesson plans into standardized interoperable Units of Learning (UoLs) (Koper & Tattersall, 2005; Koper *et al*, 2004; Burgos *et al*, 2005, 2005a). These UoLs can be created with general purpose editors or with specific IMS LD editors, like CopperAuthor (Van der Vegt, 2005) or Reload LD Editor (Bolton, 2004), and they can be run with several tools and engines, like CopperCore (Vogten & Martens, 2004) or Sled (OUUK, 2005). There is a number of existing runnable example UoLs (OUNL, 2004; OUNL, 2002) focused on several educational issues although not too many on games. Furthermore, it has not been tested extensively during the development of IMS LD whether it could model computer-based educational games. IMS LD has several features to suggest that it can be used to do it.

2. What is an educational electronic game?

Although it is not the main drive of this article to derive a detailed taxonomy of games, a basic conceptualisation of a game is needed in order to examine educational games and IMS LD. Thus, it is necessary to draw a basic profile to be used as a conceptual framework for this paper and for further research. Therefore, an enumeration of essential features of a game to be considered as an educational electronic game will be made. This means, to define first the general features that a game must have, secondly, what components should be added to use it as an educative one and, thirdly, the characteristics of an electronic game.

Reading the classical descriptions on the topic (Huizinga, 1938; Callois, 1967) the essential features of a game are:

- free action, begun and finished by the user
- imaginary, parallel to the real world, replicating a universe or an activity without any consequence on actual issues
- limited, in time and space
- following a set of rules, a specific and private framework
- with an uncertain solution and development, since every run, every play, is different and depends on unpredictable behaviour of users

For a game to be an educational one, some additional features are required (Sutton-Smith, 2001; Salen & Zimmerman, 2003):

- it has to start with a problem to be solved
- it has to be unproductive itself; it doesn't generate any property or wealth. The drive is the gaming activity itself
- it has a correct solution, at least
- it should have something to be learnt by the user/player, while introducing new knowledge, fixing previous acquired knowledge, training skills, sharing experiences, discovering new concepts, developing outcomes

Finally, to be an electronic game (Wolf, 2003) the educational game must be run on an electronic platform, such as a computer, an online terminal, a video-player, a PDA, a mobile phone etc. We will focus on computer-based games, since IMS LD is also based on this platform.

3. Main components of a computer-based educational game

Every game and every activity can incorporate an educational element defining a related scenario that seeks educational objectives. Thus, and depending on the final goal, the original drive and the environment, any game can be considered educational, beyond its original objectives. The focus of this paper, though, is on games designed to be educational.

Following a brief taxonomy of educational games we find several broad categories: chance, action, builders/creators, board, strategy, sports, management/planning, intelligence, platforms, replicates of scenarios or universes and simulators. Mixed models are typical, joining together several features, as simulators and action games, universe replicates and builders or strategy and platforms (Goldsmith, 1999; Klabbers, 1989).

Bearing all these categories in mind, a broad variety of games has been taken as a sample to extract usual components, in order to the current market of sellers, producers and users. Several catalogues and magazines have been used as a base, matching their content with the taxonomy described previously (Catalogues, 2005). Table 1 shows a list of technical and didactical components in the games.

Components	Remarks
1.1 Didactical components	
Single or multiple solution	There is only one chance or several feasible solutions to complete the game satisfactorily
Opened or closed solution	The solution is internally chosen as a result of a selection between several possible solutions or the user can generate his own solution not previously stored inside the application
Individual or collaborative solution	One or several players are needed for a happy result
Collaborative or competitive execution	The solution must be reached as a result of a collaborative work or a competition is established to get it
With dynamic feedback	A valuation of the user action is provided and it is conditioning the flow in the game
With adaptive learning	Personalizing contents, itineraries, assessments, depending on the user profile, on his rhythm and on his playing
With incremental or isolated learning	There is a number of progressive learning levels or they are isolated independent cells

1.2 Technical components	
Single or multiple player	Just one single user at the same time or several users can be playing
With or without users grouping	Sharing resources, goals and communication
Local or distributed running	A connection to a network or to Internet is needed for a multi-terminal/multi-user execution
Synchronous or asynchronous running	Communication at run-time with broadcasting
With or without multimedia resources (video/audio)	Multimedia files are integrated and used, such as video or audio files
With strong graphical support	Graphical illustration is a bottleneck for responding time and storing capacity because of the amount of information
With 2D or 3D graphical design	Illustration comes from two dimensional and flat sources or from three dimensions and incorporates depth as a key factor. This means more resources consumption
With vector or bitmap images	Illustration is math-scalable-based without deformation or it has a photographic quality
With a run-time engine or with engine not adaptable at run-time	a) There is an AI engine that makes calculations at run-time and that modifies structures and the game itself; b) or everything is defined at design-time
With editable or static fields	Fields can change their content during the play
With personalized features or static profiles	Specific features of the play and the user can be modified during the running
Saving and reading external files	Retrieving and storing some user, play or behaviour information, for instance
With communication from and to other applications/tools	Sending information and interchanging variables during the play
With or without interaction in the real world. Blended-gaming or b-gaming	It is an electronic stand-alone game or an interaction with the real world is needed

Table 1. Main components of educational electronic games

Few of the above components can be selected as a stand-alone feature, always requiring others and combining sources and outcomes. Furthermore, the implementation in IMS LD of any of them entails the nested joint implementation of a set of them. For instance, a technical component such as “Local or distributed running” comes with “users grouping”; also the didactical component “dynamic feedback” goes with “adaptive learning”. Likewise, and merging both lists, the didactical component “incremental learning” comes with the technical one “saving and loading external files” and the didactical one “collaborative solving” relates to the technical component “single or multiple player”, just to mention a few.

4. What components can be modelled with IMS LD

Taking into account the previous list of technical and didactical components a correspondence can be established between them and their ability to be modelled with IMS LD. The next table (see Table 2) has four columns: 1) analysed component, 2) whether it can be modelled with IMS LD or not, 3) what level of the IMS LD specification is required (A, B and C, as aforementioned) and 4) additional remarks.

Components	IMS LD	Level	Remark
1.1 Didactical components			
Single or multiple solution	Yes	A	
Opened or closed solution	Yes	A	
Individual or collaborative solution	Yes	B	XML(1)
Collaborative or competitive execution	Yes	B	XML
With dynamic feedback	Yes	B	XML
With adaptive learning	Yes	B	XML
With incremental or isolated learning	Yes	B	XML
1.2 Technical components			
Single or multiple player	Yes	B	XML
With or without users grouping	Yes	A	
Local or distributed running	Yes	A	
Synchronous or asynchronous running	Yes	A	
With or without multimedia resources (video/audio)	Yes	A	
With strong graphical support	Yes	A	
With 2D or 3D graphical design	Yes	A	
With vector or bitmap images	Yes	A	
With a run-time engine or with engine not adaptable at run-time	Partial	-	Depends on the executable module itself, and not on IMS LD

With editable or static fields	Yes	B	XML
With personalized features or static profiles	Yes	B	XML
Saving and reading external files	Partial	-	Depends on the executable module itself, and not on IMS LD
With communication from and to other applications/tools	Yes	B	XML
With or without interaction in the real world. Blended-gaming or b-gaming	Yes	A	
(1) <i>Additional XML component needed</i>			

Table 3. Main components of the games to be modelled in IMS LD

The first reading of Table 2 shows that, all the components but two, both didactical and technical, can be modelled in IMS LD. Concerning Level A there are two reasons: a) IMS LD has integrated the feature itself: grouping, single or multiple solution, open or closed solution, synchronous or asynchronous solution; and b) the feature doesn't depend directly on IMS LD but can be properly supported: 2D/3D design, strong graphical support, multimedia resources. Concerning Level B, the same two reasons come up once more: a) IMS LD has built-in the component or can be programmed with some additional help of global elements, properties, conditions and monitoring services in XML, like editable fields, personalized features, individual or multiple running, collaborative or competitive running; and b) it doesn't depend on IMS LD but it is supported: communication from and to other applications.

This previous analysis leaves apart two main components: 1) The first one is the interaction with an engine at run-time, which is not possible with IMS LD so far. For instance, modification of the structure or dynamic management of users on-the-fly, at run-time. 2) The second component is the facility of reading and writing external files. There is a difference between loading or retrieving a file and reading a file. Although an external file can be uploaded using the FILE property in IMS LD, the use of data stored in the file in the instance of the Unit of Learning is not possible. For instance, in the set-up of variables or providing dynamic contents for properties or fields. Furthermore, IMS LD doesn't allow a direct process to save data in an external file. Although the specification manages global properties to store contents outside an executed run, this information cannot be exported to any known and usual format.

Saving and loading information becomes important concerning the interoperability between some outcomes already created inside and outside an IMS LD platform. The outsourcing of data generated in IMS LD as well as the import of data generated outside IMS LD into a learning structure ensures the proper integration of any Unit of Learning with the usual tools and documents needed in the daily costume of an user, such as managing students dossiers, valuating assessments, adding attachments, merging notes taken in different platforms and so on.

These two issues, interaction at run-time and external files, are fully related to the own nature of any IMS specification and something hard difficult to approach right now. Tools are

evolving quite fast and some key expected improvements from engines/tools/applications are coming, chasing more flexibility in the creation and running of more powerful Units of Learning. On the other hand, another solution is a revision of IMS LD, since it is a fresh specification that could need some fine tuning in forthcoming versions. [1]

5. Three models of implementation for an example Unit of Learning

In order to illustrate the previous analysis, a Unit of Learning has been developed to support the storing and retrieving information process in a external file inside IMS LD and the use of its content if the learning flow. The chosen game to support our argument is called *Caminatas* and it was included for the first time in a multi-rom providing support for language acquisition in Primary School (OUP, 2004)(see Figure 1).

Caminatas consists of several stand-alone modules with basic data communication, grouped around a central main module to grant access and share tasks. It has some features to be personalized (user name), some setting-up for the full multi-rom (ambient audio) and some adaptive configuration (visibility of intros just in the first running of every instance). It was scripted and programmed in *Flash/Action Script* by the first author of this paper, together with the publishing company Oxford University Press and furtheron it has been adapted to XML and IMS LD in the Open University of the Netherlands to be used as a base on educational gaming research and LD (Burgos, 2005).

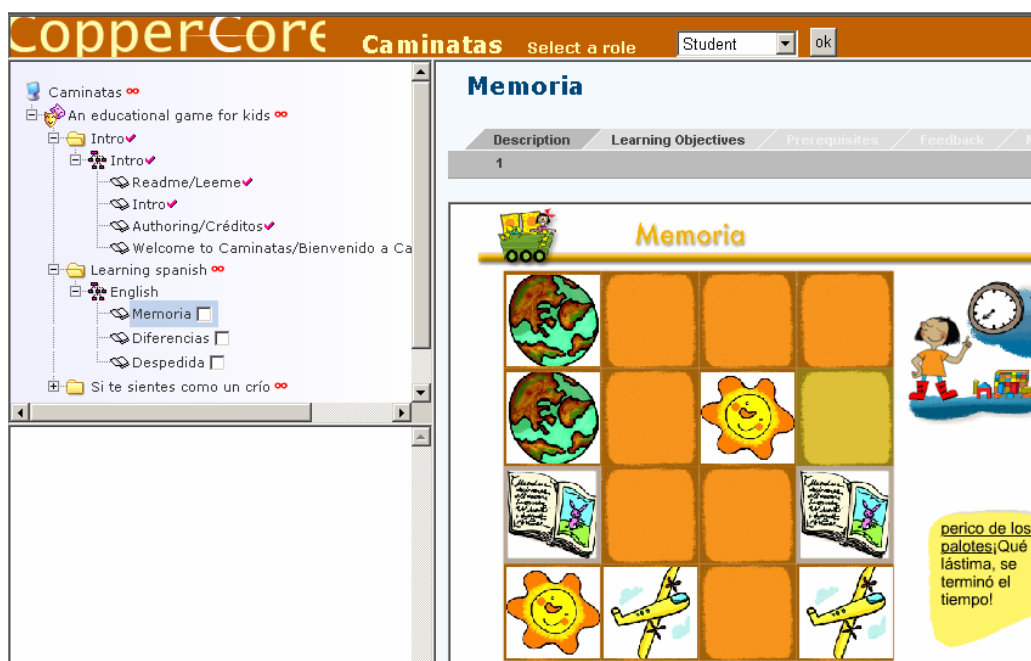


Figure 1. Computer-based educational game *Caminatas* in IMS LD

Concerning IMS LD version of *Caminatas* there are three possibilities of implementation: a) a mixed model Action Script – IMS LD; b) a mixed model IMS LD – Action Script; and c) a pure model IMS LD.

The game consists of three very different moments or sections: *Introduction*, *Game* and *Goodbye*. *Introduction* involves setting-up the basic requirements of a run, shows the front page and requests the initial personal information of the player. In the *Game* section, a play of the game is run using the previous data. In the last section, *Goodbye*, these previously

input data are used again, to personalize the closing message. The three sections are single files, stand-alone programmes, which send and receive the input data between them.

To solve this challenge of intercommunication between the modules there are three feasible models of implementation:

- Model A: mixed model Action Script - IMS LD, the component *SharedObject* of Action Script is used as the built-in mechanism to store-retrieve information provided by Macromedia Flash in a proprietary extension called .SOL. In this case, IMS LD is just a container of modules, a structural support for the internal execution of every module in Flash (Richards, 2005). The consequence is a one hundred per cent compatible modelling with specific insertions of code, regardless of their nature or original language. Java, Javascript, Action Script, Lingo an a large etcetera are valuable, therefore. This is the option taken in the provided example (see Figure 2).

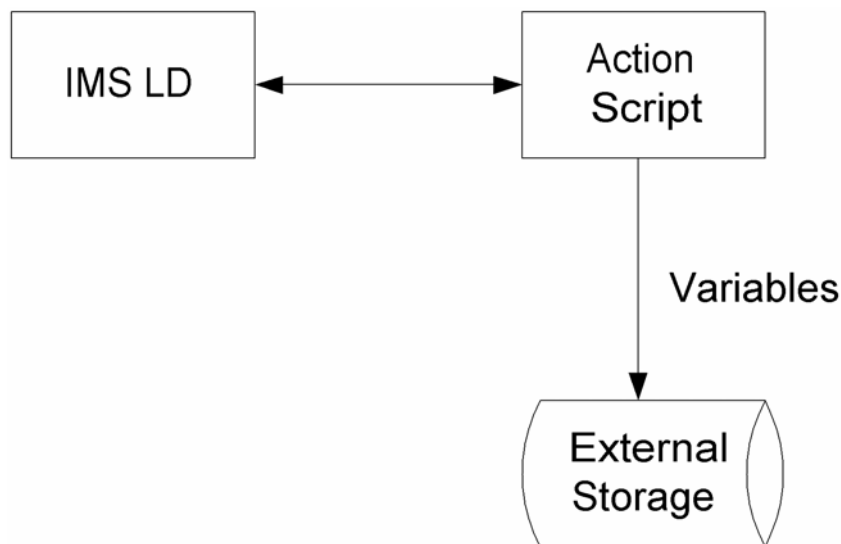


Figure 2. Model A

- Model B: mixed model IMS LD - Action Script, the information is sent from one system to another one using the variable interchanging through the heading of the modules and the methods GET and POST, usual ways in URL's communication and information linking with XML and CGI's. XML or any other Mark-up Language supporting these methods becomes the actual bridge between platforms. Afterwards, saving data could be done inside IMS LD, using global variables and local variables and keeping the access in the Unit of Learning, or in Action Script, using the procedure explained in a) (see Figure 3).

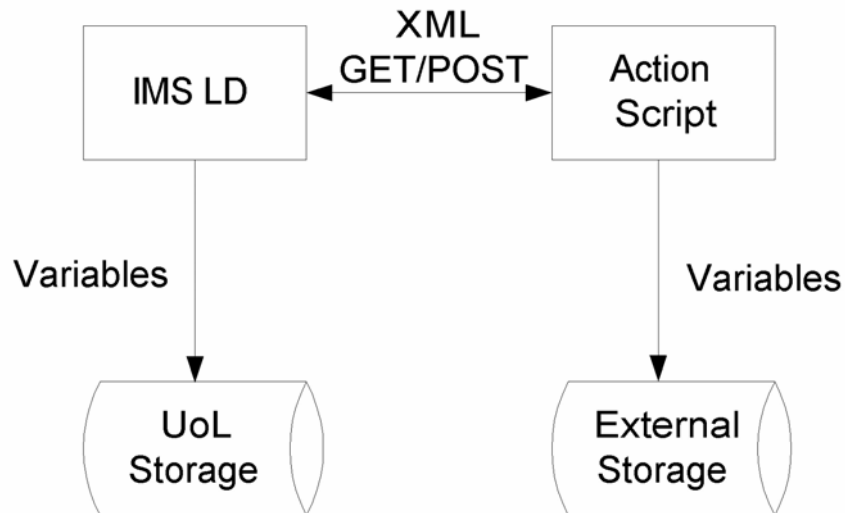


Figure 3. Model B

- Model C: pure model IMS LD, the game is completely developed in IMS LD, replicating in the specification the script already existing in Flash and using Action Script only to draw cartoons, if needed, and IMS LD to store, retrieve and use variables and content. The IMS LD core elements in Level B are main parts in this structure but the GET/POST mechanism is passed by. Again, the information is still stored in the IMS LD system but with no access as an external file (see Figure 4).

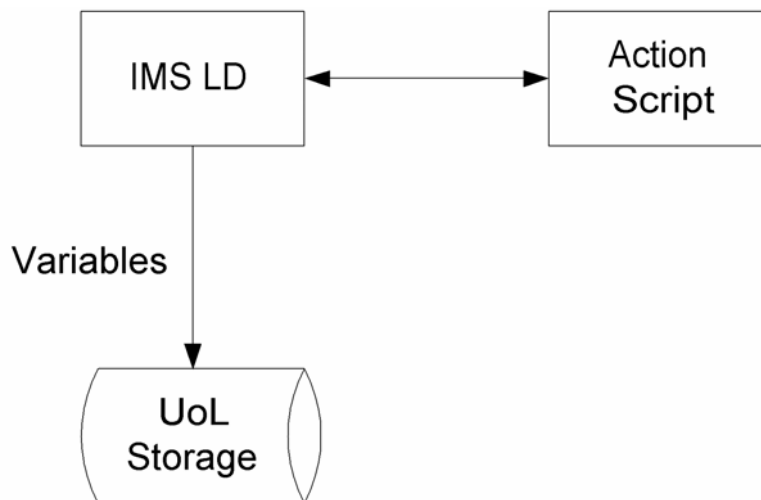


Figure 4. Model C

6. Conclusion

Every component in a computer-based educational game can be modelled in both ways, mixed or pure. In the pure option, only with IMS LD, several components of Levels A and B can be used to model the structure of the game. Activities, roles, environments, method, acts, resources, for instance, and the communication among all of them, joining properties and additional conditions are available in order to get a right flow that represents the original script properly. On the other hand, in the mixed option, with IMS LD and some additional XML, a communication can be established between both using specific features of this language and the system of global properties, local properties and global elements. In this

option, therefore, Level B is basic to manage all these elements. Following the list of main components of any educational game enumerated in this paper, 20 of 22 features can be modelled this way. Level A and Level B provide resources enough to sort it out.

The only two components, both technical, not completely affordable by now are 1) an engine at real time and 2) saving and reading external files. In terms of 1) an engine could be developed with any other language (Java, Action Script...) and to link the related information created with them to IMS LD through XML code or using IMS LD as a container of the main action and provider of learning and support activities. Because of the actual nature of the specification that clearly divides design time from run-time, IMS LD is not able to provide this feature by now. Maybe any of the current tools under development can support it in the upcoming releases.

In terms of 2) saving and reading external files, IMS LD allows to upload an external file using the FILE property but it is not able to use the content of this file neither than save any property nor content in an external one. This feature can be developed complementing the IMS LD action through an external language or program that saves and reads the information in disk to send the result to the Unit of Learning managed by specification. Albeit IMS LD cannot solve it directly it is possible to be done through a walk-around, taking advantage of the components and functions of others.

Regardless these two important issues, it is largely proved that IMS LD can modelled 20 of the 22 analysed components. Just two of them, the engine, can not be directly affordable and just one of them has no obvious and quick solution. The other one, the external files, can use a combination of IMS LD and other additional languages to find the closest solution to the challenge. This means a very clear outcome of the flexibility of IMS LD to develop educational electronic games.

7. References

- Bolton University (2004) "Reload". Retrieved March 29th, 2005 [<http://www.reload.co.uk/>]
- Burgos, D. (2005) "Caminatas example in IMS LD". Retrieved March 19th, 2006. [<http://hdl.handle.net/1820/357>]
- Burgos, D., Berbegal, N., Griffiths, D., Tattersall, C., Koper, Rob. (2005) "IMS Learning Design: How specifications could change the current e-learning landscape", e-Learning World, issue 2, March-April 2005. ISSN: 1811-069X. Moscow: Moscow State University of Economics, Statistics and Informatics – MESI. Retrieved October 1st, 2005. [<http://hdl.handle.net/1820/354>]
- Burgos, D., Berbegal, N., Griffiths, D. (2005a) "IMS Learning Design Level 0". Retrieved December 7th, 2005 [<http://moodle.learningnetworks.org/mod/resource/view.php?id=174>]
- Callois, R. (1967) "Man, play and games". University of Illinois Press
- Catalogues (2005) Amazon [<http://www.amazon.com>], Computer Gamer [<http://cgw.1up.com>], Gamers [<http://www.gamers.com>], PCGamer [<http://www.pcgamer.com>], Soft32 [<http://buy.soft32.com>]. Retrieved April 19th, 2006
- IMS (2001) *IMS Content Package Specification*, Retrieved March 1st, 2005 [<http://www.imsglobal.org/contentpackage/index.html>]
- Educational Technology Expertise Centre (OTEC), Open University of The Netherlands. Retrieved March 22nd, 2006 [www.ou.nl]
- Goldsmith, J. (1999) "A taxonomy of games". California, USA: Caltech. Retrieved March 9th, 2006 [<http://www.gg.caltech.edu/~jeff/games/taxonomy.html>]
- Huizinga, J. (1938) "Homo Ludens". Beacon Press

- IMS (2003) "IMS Learning Design specification". Boston, USA: IMS Global consortium. Retrieved March 8th, 2006 [<http://www.imsglobal.org>]
- Jeffery, A., Currier, S.(2004) "What Is IMS Learning Design?" *Cetis standards briefings series*. Retrieved November 8th, 2005 [http://www.cetis.ac.uk/lib/media/WhatIsLD_web.pdf]
- Klabbers, J. (1989) A User-oriented taxonomy games and simulations. In: Oosthoek, H., and Vroeijenstijn (eds.), "Higher Education and New Technologies". Oxford Press
- Koper, R., Spoelstra, H., Burgos, D. (Eds)(2004). "Learning Networks using Learning Design. A first collection of papers". Educational Technology Expertise Centre, The Open University of the Netherlands. Retrieved January 20th, 2006 [<http://dspace.learningnetworks.org/handle/1820/291>]
- Koper, R., Tattersall, C. (Eds) (2005) "Learning Design: A Handbook on Modelling and Delivering Networked Education and Training". Germany: Springer Verlag
- OUNL (2004) "Learning Network for Learning Design (LN4LD)". Open University of The Netherlands, OTEC. Retrieved March 14, 2006 [<http://moodle.learningnetworks.org/>]
- OUNL (2002). "DSpace". Open University of The Netherlands, OTEC. Retrieved March 14, 2006 [<http://dspace.learningnetworks.org>]
- OOUK (2005) "Sled player". Open University of United Kingdom. Retrieved March 14, 2006 [<http://sled.open.ac.uk>]
- OUP (2004) Caminatas. Oxford University Press. Retrieved May 15th, 2005 [<http://www.oup.es/>]
- Richards, G. (2005) "Designing Educational Games", chapter 13. In Koper and Tattersall (2005)
- Salen, K., Zimmerman, E. (2003) "Rules of play: game design fundamentals". The MIT Press
- Sutton-Smith, B. (2001) "The ambiguity of play". Harvard University Press
- Tattersall, C., Koper, R. (2003) "EML and IMS Learning Design: from LO to LA". Educational Technology Expertise Centre, The Open University of the Netherlands. [<http://dspace.learningnetworks.org/retrieve/186/LTSN+Presentation+-+EML+and+LD+20030238.pdf>]
- Tattersall, C., Manderveld, J. Hummel, H., Sloep, P., Koper, R. (2003) *IMS Learning Design Frequently Asked Questions*. The Open University of The Netherlands. [<http://dspace.learningnetworks.org/handle/1820/116>]
- UNFOLD (2004) "UNFOLD Project". Retrieved March 10th, 2006 [<http://www.unfold-project.net/>]
- Van der Vegt, Wim (2005) "CopperAuthor". Retrieved March 29th, 2005 [<http://www.copperauthor.org/>]
- Vogten, H., Martens. H (2004). "CopperCore". Retrieved on March 14, 2006 [<http://www.coppercore.org/>]
- Wolf, M.J.P, Perron, B. (Eds) (2003) "The video game theory reader". Routledge, Inc.

8. Notes

[1] The IMS Learning Design specification supports the use of a wide range of pedagogies in online learning. Rather than attempting to capture the specifics of many pedagogies, it does this by providing a generic and flexible language. This language is designed to enable many different pedagogies to be expressed. The approach has the advantage over alternatives in that only one set of learning design and runtime tools then need to be implemented in order to support the desired wide range of pedagogies. The language was originally developed at the Open University of the Netherlands (OUNL), after extensive examination and comparison of a wide range of pedagogical approaches and their associated learning activities, and several

iterations of the developing language to obtain a good balance between generality and pedagogic expressiveness. Full specifications are available at: <http://www.imsglobal.org/learningdesign/index.html>

[2]With the current version 1.0 of IMS LD, though, the problem about storage can be solved using already existing languages commonly used to create resources, such a web page. For example, with HTML and JavaScript, the *Fscomand* function can be used to import, export and save information. With *Action Script*, of Macromedia Flash, the component *SharedObject* can save and retrieve information inside environmental variables, local or global ones. Also the function *MMSave* in the authoring system of Flash provides the possibility of storing an external ASCII-text file with the content of internal fields and variables in a CSV format.

Besides, if a higher level editor, language or programming environment is used to create resources, like Java or Lingo of Macromedia Director, for instance, there are some specific functions on storing/retrieving issues to do these jobs very easily.

Using these languages, the storing problem is solved and some information can be imported or exported inside the learning structure of an IMS LD Unit of Learning, although with some extra help from external languages.