Sessink ODT, Beeftink HH, Tramper J, Hartog RJM (2003) Author-Defined Storage in the Next Generation Learning Management Systems. Proceedings of the Third IEEE International Conference on Advanced Learning Technologies (ICALT), 2003, pp. 57-61

Author-Defined Storage in the Next Generation Learning Management Systems

Olivier Sessink, Rik Beeftink, Johannes Tramper, Rob Hartog

Olivier Sessink, Rik Beeftink, Johannes Tramper
Food and Bioprocess Engineering Group
P.O. Box 8129 6700 EV
Wageningen University, The Netherlands
Olivier.Sessink@wur.nl, Rik.Beeftink@wur.nl, Hans.Tramper@wur.nl
++31 (317) 483229, ++31 (317) 482237 (fax)

Rob Hartog
School of Technology and Nutrition
P.O. Box 8128 6700 ET
Wageningen University, The Netherlands
Rob.Hartog@wur.nl
++31 (317) 483408 / ++31 (317) 483158 (fax)

Abstract

One of the current trends in E-learning is the development of student-activating learning material. In four research projects aiming at the design of high quality learning material, a large body of student-activating learning material is being developed. During the development of this learning material, the limitations of the current generation learning management systems became obvious. The forthcoming SCORM 1.3 standard will resolve some of these limitations, but we have identified six additional functional requirements. The learning management system should enable adaptivity, the retrieval of history and state, comparison of results, tracking for pedagogical research, shared reference databases, and problem scenario databases. Each requirement will be illustrated with examples from learning material developed in one of the research projects. Also an overview is given of temporary workarounds we have developed to deploy this learning material in the current generation of learning management systems. However, we argue that future learning management systems with an author-defined storage facility will satisfy all six requirements.

Introduction

One of the current trends in E-learning is the development of student-activating learning material. In the Food and Biotechnology (FBT) program at Wageningen University, a large body of digital learning material is being developed. Most courses are supported by simple static objects, but in four research projects more advanced learning material is being developed [7]. One of the results of these projects is the articulation of new requirements for a next generation of learning objects as well as learning management systems. These projects aim to exploit the pedagogical possibilities of digital learning material, resulting in the development of activating learning material [2]. This learning material has been used for the last years and is appreciated both by students and lecturers [4][5][6]. Most of the student-activating learning objects, developed in these four research projects, process data, and these data are often related to a user action (e.g. a mouse click in a specific region or text submitted in a form). We will call these objects active objects.

One of the results of these research projects is a highly increased awareness of the limitations of the current generation learning management systems (LMSs). Future learning management systems are likely to implement the forthcoming SCORM 1.3 specification with the Simple Sequencing Specification from IMS [1]. Once learning management systems support SCORM 1.3, some of the most pressing shortcomings will be resolved, but six requirements will remain. The LMS should enable adaptivity, retrieval of history and state, comparison of results, tracking for pedagogical research, a shared reference database, and a problem scenario database. This paper will give an overview of these requirements, and why each of the requirements is neither met by the current generation LMSs nor by the SCORM 1.3 standard. Each requirement will be illustrated with examples from learning material developed in one of the research projects. Next, the paper will give an overview of workarounds we have developed to deploy this learning material in the current generation of learning management systems. Finally, the paper will describe a single solution for all requirements, the author-defined storage. This is a database where the data structure is to be defined by the author. We will argue that future standards should include this feature to overcome the mentioned problems when deploying advanced learning material.

Adaptivity

"Adaptive systems cater information to the user and may guide the user in the information space to present the most relevant material, taking into account a model of the users goals, interests and preferences" [3]. In an educational context, the users competence levels are also part of the model and the model is called a student model. In demo site [7d], a student model for the process engineering knowledge domain has been developed. The learning material can raise or lower the levels of competences in the student model based on the interaction with the student. In the navigation overview, the data in the student model are used to inform the student if a document fits his competences. This helps students to quickly find their way through the material, thus limiting the time spent on searching, and maximizing the time spent working with the learning material.

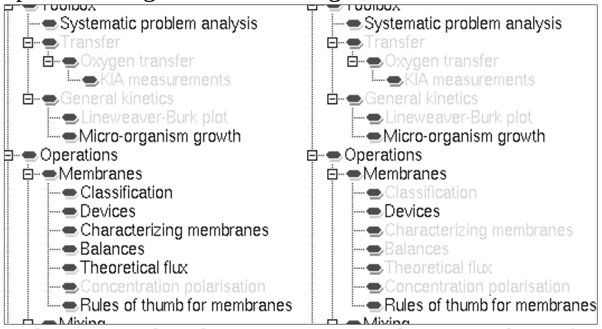
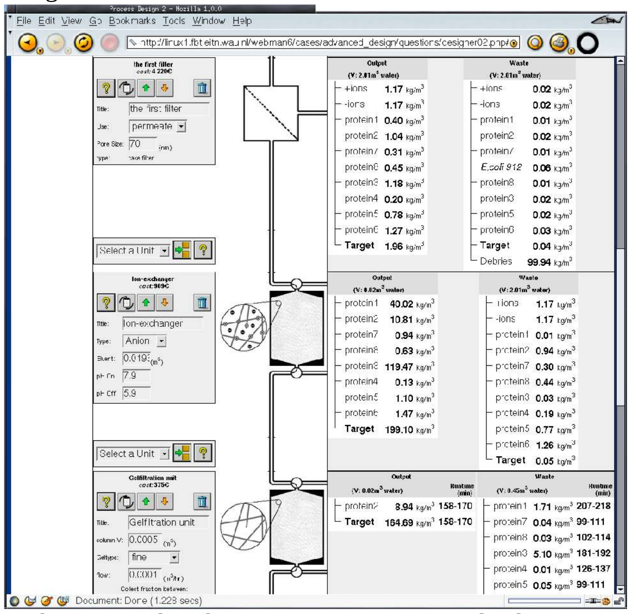


Figure 1. The document overview; on the left for a student educated in membrane technology, on the right for a starting student.

The combination of Simple Sequencing and the CMI model in SCORM 1.3 does facilitate limited adaptivity. However, the only available data for the user model are the data in the fixed CMI model. The data that can be set by the active learning objects are also limited to the CMI model. Alternative or more complex approaches to adaptivity such as described above are therefore not possible within SCORM 1.3.

Retrieving history and state

We have developed learning objects that need to retrieve their state history, or the state history of other objects, in order to initialize themselves. In demo site [7a], a tool has been developed with which students should design a downstream processing chain. Downstream processing is a series of steps in which a product is purified. Students work several days with this tool to carry out several assignments. It is essential that they can compare designs with other designs from previous days or previous assignments [4]. This tool therefore needs a storage facility, so the student can retrieve previous designs.


Process Design 2 - Mozilla 1.0.0
File
Edit
View
Go
Bookmarks
Tools
Window
Help
http://linux1.fbt.eitn.wau.nl/webman6/cases/advanced_design/questions/designer02.php
Select a Unit
Select a Unit
Document: Done (1.228 secs)

the first filter
cost: 4 220€

Title: the first filter
Use: permeate
Pore Size: 70 (nm)
type: case filter

Output (V: 2.01m^3 water)		Waste (V: 2.01m^3 water)	
+ions	1.17 kg/m^3	+ions	0.02 kg/m^3
-ions	1.17 kg/m^3	-ions	0.02 kg/m^3
protein1	0.40 kg/m^3	protein1	0.01 kg/m^3
protein2	1.04 kg/m^3	protein2	0.02 kg/m^3
protein7	0.31 kg/m^3	protein7	0.01 kg/m^3
protein8	0.45 kg/m^3	*E.coli 912*	0.06 kg/m^3
protein3	1.18 kg/m^3	protein8	0.01 kg/m^3
protein4	0.20 kg/m^3	protein3	0.02 kg/m^3
protein5	0.78 kg/m^3	protein5	0.02 kg/m^3
protein6	1.27 kg/m^3	protein6	0.03 kg/m^3
Target	1.96 kg/m^3	**Target**	0.04 kg/m^3
		Debries	99.94 kg/m^3

Ion-exchanger
cost: 909€

Title: Ion-exchanger
Type: Anion
Eluent: 0.0193 (m^3)
pH On: 7.9
pH Off: 5.9

Output (V: 0.02m^3 water)		Waste (V: 2.01m^3 water)	
protein1	40.02 kg/m^3	+ions	1.17 kg/m^3
protein2	10.81 kg/m^3	-ions	1.17 kg/m^3
protein7	0.94 kg/m^3	protein1	0.01 kg/m^3
protein8	0.63 kg/m^3	protein2	0.94 kg/m^3
protein3	119.47 kg/m^3	protein7	0.30 kg/m^3
protein4	0.13 kg/m^3	protein8	0.44 kg/m^3
protein5	1.10 kg/m^3	protein3	0.03 kg/m^3
protein6	1.47 kg/m^3	protein4	0.19 kg/m^3
Target	199.10 kg/m^3	protein5	0.77 kg/m^3
		protein6	1.26 kg/m^3
		Target	0.05 kg/m^3

Gelfiltration unit
cost: 375€

Title: Gelfiltration unit
column V: 0.0005 (m^3)
Geltype: fine
flow: 0.0001 (m^3/hr)
Collect fraction between:

Output (V: 0.02m^3 water)		Runtime (min)	Waste (V: 0.45m^3 water)		Runtime (min)
protein2	8.94 kg/m^3	158-170	protein1	1.71 kg/m^3	207-218
Target	164.69 kg/m^3	158-170	protein7	0.04 kg/m^3	99-111
			protein8	0.03 kg/m^3	102-114
			protein3	5.10 kg/m^3	181-192
			protein4	0.01 kg/m^3	126-137
			protein5	0.05 kg/m^3	99-111

Figure 2. The downstream process designer, a learning object that stores and retrieves much state information.

The same kind of functionality is used in demo sites [7b] and [7c] where previous decisions or previous actions may have consequences for what is presented to the student [5].

The CMI model used in SCORM does specify the *core_lesson* property to temporary store the learning object state. However, this property is limited to 256 bytes of ASCII and its use is limited to a single learning object.

This property will not meet the requirements when a compound state (e.g. a state that consists of several attributes and their values) or a state history is required for one or more learning objects.

Enabling students to compare (aggregate) results

In the previous section, a downstream processing design tool is described. This tool has another interesting feature. It stores several scores for all designs, like best recovery, best purity, least number of units, least waste and best price. Every assignment that uses this tool can show which student scored the best result for each category. This introduces a competitive aspect, which is a great motivator [4]. Also this clear indication that other students are working on the same learning material will motivate the students if the material is used in a distance-learning setting. The SCORM specification does not feature a way to store and retrieve this type of scores.

Tracking for pedagogical research

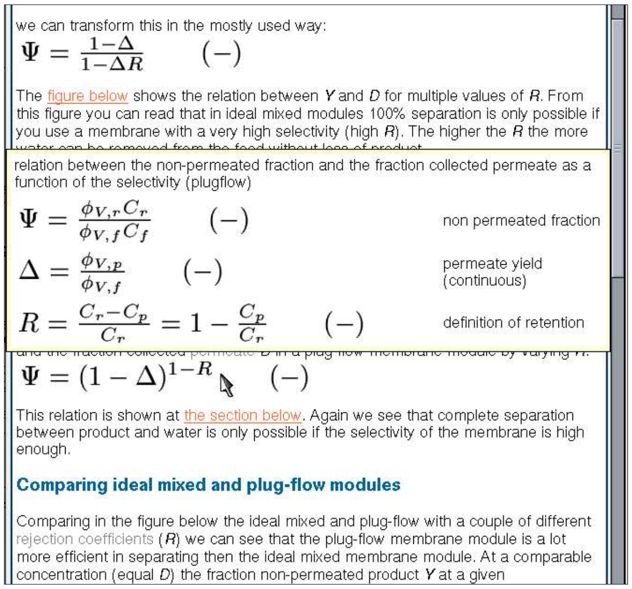
In demo site [7b], learning material is developed that tracks all interaction aspects [6]. The internal navigation within the learning objects and the navigation outside the learning objects are stored. It is possible to follow every activity from a student. A typical tracking log for some student could look like:

1. student starts learning object I (a question is presented)
2. student answers possibility B in this learning object (feedback on possibility B is presented)
3. student starts learning object II in the library (information is presented)
4. students continues with learning object I and answers possibility A (feedback on possibility A is presented)
5. etc.

After interpretation this gives much information to the lecturer about the understanding of different parts of the learning material, the difficulties students have, and the effect of the feedback presented by the learning objects to the students.

Shared reference database

In demo site [7d], there is a shared definition list and a shared equation list, which are used throughout all learning objects from the department of Process Engineering. The definition list is simply a list of keywords and their definition. The equation list is a little more complex. The base is a list of equations with a description. Every equation has a list of associated equations describing the symbols in the pertinent equation. Both the equations and the definitions are server-side added to the learning objects.



we can transform this in the mostly used way:

$$\Psi = \frac{1-\Delta}{1-\Delta R} \qquad (-)$$

The figure below shows the relation between *Y* and *D* for multiple values of *R*. From this figure you can read that in ideal mixed modules 100% separation is only possible if you use a membrane with a very high selectivity (high *R*). The higher the *R* the more water can be removed from the food without loss of product.

relation between the non-permeated fraction and the fraction collected permeate as a function of the selectivity (plugflow)

$$\Psi = \frac{\phi_{V,r} C_r}{\phi_{V,f} C_f} \qquad (-)$$ non permeated fraction

$$\Delta = \frac{\phi_{V,p}}{\phi_{V,f}} \qquad (-)$$ permeate yield (continuous)

$$R = \frac{C_r - C_p}{C_r} = 1 - \frac{C_p}{C_r} \qquad (-)$$ definition of retention

and the fraction collected permeate *D* in a plug flow membrane module by varying *R*.

$$\Psi = (1-\Delta)^{1-R} \qquad (-)$$

This relation is shown at the section below. Again we see that complete separation between product and water is only possible if the selectivity of the membrane is high enough.

Comparing ideal mixed and plug-flow modules

Comparing in the figure below the ideal mixed and plug-flow with a couple of different rejection coefficients (*R*) we can see that the plug-flow membrane module is a lot more efficient in separating then the ideal mixed membrane module. At a comparable concentration (equal *D*) the fraction non-permeated product *Y* at a given

Figure 3. The mouse is over an equation and a popup is shown.

If the student moves the mouse over a keyword, a popup with the definition is shown. If the student moves the mouse over an equation, a popup is shown with the description of that equation and all related equations. This functionality is highly rewarded by the students.

Also in demo site [7c], there are six reference databases used. One of these databases has additional features and is described in the next section.

In terms of sharable content objects every definition and every equation should be a separate object. However, automatically adding this separate object to other learning objects would be out of the question, and equation objects could not refer to related equation objects. Another difference is the different nature of the information. The list is not to be studied by the student, but is typical reference information.

Larger knowledge bases, especially if they are also used outside the educational context, should not be in the author-defined storage. A link to an external server would be more appropriate.

Problem scenario database

In demo site [7c], one of the reference databases as described above is extended to a problem scenario database with microbiological hazard problems. The database contains both problems (for example *infection with Salmonella*), and reference information with solutions for all these problems (for example *detecting a Salmonella infection*). A learning object with an assignment can select random or specific problems from the database and introduce these to the student. A learning object in the library from this course allows the student to search in the same database for solutions. Since these objects share the same database, the solution is guaranteed to be available. The other advantage of the problem scenario database as illustrated above, is that it can be used to produce a range of different but equivalent assignments. The microbiological hazard database is accessed by several learning objects.

Workarounds for current generation LMSs

There is no standard for the described learning material yet. Nonetheless, we have developed such learning material and have deployed it in a well-known LMS. At Wageningen University these workarounds are used for several years now, with full appreciation of students and authors. Two workarounds are described, but of course a mixed approach would be possible as well.

In the first workaround the functionality of the LMS server is extended. Many LMSs internally consist of a webserver, a database management system (DBMS) for administration storage, and a filesystem for storage of learning objects. In this situation the webserver can be configured to run server-side languages. In our setup, the PHP scripting language was chosen. When there is no public application programming interface (API) to communicate with the LMS, the scripting language has to connect directly to the underlying DBMS. We have developed a PHP library for retrieval of the LMS administration data from the DBMS. The client-side objects (JAVA, FLASH) interface with a PHP script to store and retrieve data.

This situation is far from ideal. Most LMSs are currently not designed for server-side languages and this introduces security problems. There are several possibilities to bypass the LMS security if server side scripting is enabled on the LMS server. The PHP library is also specific for one LMS, and only available on our server, so neither the library nor the learning material is portable to other systems.

The second workaround is to run all active objects on a second server. This will avoid all interference with the LMS server. This server can be built from standard components. Our server is configured to run the PHP scripting language, and the MySQL and InterBase DBMSs. We have also developed a PHP library to retrieve LMS administration data from the LMS DBMS for authentication and authorization. The client-side active objects (JAVA, FLASH) again interface with a PHP script to store and retrieve data.

This situation is also not ideal. The objects outside the LMS are not managed at all by the LMS. The courses in the LMS consist of links to the external objects which are not managed either. Furthermore the learning material and the PHP library are again specific for this configuration, so that this material is not portable either.

Interface requirements for future generation LMSs

To deploy the learning material described in this paper in a future generation of LMSs there is a need for a standard extension to LMSs. This extension should enable the *author-defined data storage* on the LMS. Essentially, this facility adds database functionality to the LMS. This database is to be defined by the author instead of the LMS manufacturer. Therefore the extension should include a user interface to define and edit the database. Since the learning objects need to access the database, the extension should also include an API. The extension should also provide some security (e.g. which learning objects, and which authors have access to the database), and it should prevent name-clashes.

The first requirement for the user interface of the LMS is an interface to define the data model. An obvious way to enable this is to add an SQL data definition upload facility to the LMS. Besides SQL upload, a web interface as found in phpMyAdmin, or a database upload system as found in Frontpage would certainly improve this user interface. For interoperability reasons it is important that the data structure definition is portable. It should be possible to transfer the data structure from one LMS to another LMS.

The user interface should also feature some authorization options. Which learning objects are allowed to retrieve data from this datastore, and which learning objects are allowed to store data in this datastore. Possible options should include a selection 'objects from author' and 'objects in course'.

Some of the data in the author-defined data storage will be of interest to the lecturer. Other data will be provided by the author. In both cases the LMS should have a user interface to search, view and edit the data in the author-defined data store. Such a user interface could be similar to the phpMyAdmin web-interface. Another possibility would be to enable upload and download of the data in a specified format (e.g. Access format or tab delimited text file) so third party software can be used to search, view and edit the data. It should also be possible to export and import the data itself from one LMS to another.

Last but not least, the active learning objects need to access the author-defined data store. New answers from students have to be inserted or updated in the datastore, or definitions have to be selected from the datastore.

For client-side active learning objects (e.g. JAVA applets or FLASH movies), there is an interface described in SCORM. The SCORM 1.3 runtime API defines a JavaScript interface for communications initiated by a client-side learning object to the LMS. This API is used to get and set values from the CMI model [1]. The interface to the author-defined storage could be an extension of this API, for example a method named LMSRunSql().

Example usage would then look like:

```
var Database_ID = "5432";
var SqlStatus = LMSRunSql(Database_ID, "update scores set purity='99' where assignment='DSPD';");
if (SqlStatus == 0) {
                // Succeeded
} else {
                // Error condition; handle appropriately:
}
```

Enabling server-side active objects (e.g. JSP or PHP objects) is a completely new area for LMSs. Whether or not to support server-side active objects in an LMS is a discussion beyond the scope of this article. Three of our four research projects referred to in this article have developed their learning material using server-side technology. If the LMS would support server-side active objects, the interface would be very simple. The objects should simply call a method from the LMS to access the author-defined data storage. An example in PHP would look like:

```
$Database_ID = "5432";
$SqlStatus = LMSRunSql($Database_ID, "update scores set purity='99' where assignment='DSPD';");
if ($SqlStatus == 0) {
                // Succeeded
} else {
                // Error condition; handle appropriately:
}
```

Conclusion

In a number of research projects the limitations of the current generation of learning management systems became apparent. Apart from tangible results such as digital learning material these projects resulted in an articulation of the shortcomings of current learning objects and learning management systems and a proposal to alleviate these shortcomings. In this paper we have described six functional requirements for learning management systems. This learning material is neither supported by the current generation of learning management systems nor does it fit in the forthcoming SCORM 1.3 standard.

Having an author-defined storage facility will satisfy all six requirements, and will enable the deployment of this learning material. We think there is need for a standard that will include the author-defined storage facility. This standard should specify 1) what methods from the LMS can be called by a learning object, 2) how data structure and the data itself can be transferred from one LMS to another LMS.

References

[1] Advanced distributed learning (ADL), (2002). *SCORM 1.3 working draft 0.9.* http://www.adlnet.org/

[2] Anderson, J.R. (1995). *Learning and memory. An integrated approach.*, John Wiley & Sons, Inc.

[3] Brusilovsky, P., Kobsa, A., Vassileva, J., editors (1998). *Adaptive Hypertext and Hypermedia.* Kluwer, Dordrecht. http://www.wkap.nl/prod/b/0-7923-4843-5

[4] Schaaf, H. van der, Vermuë, M., Tramper, J., Hartog, R., (2003). *A Design Environment for Downstream Processes for Bioprocess-engineering students*. submitted to the European Journal of Engineering Education.

[5] Wilmsen, T., Bisseling, T., Hartog, R., (2002). *Web based learning support for experimental design in molecular biology*. Proceedings of the Edmedia conference, 2063-2068.

[6] Wilmsen, T., Hartog, R., Bisseling, T., (2003) *Web based learning support for experimental design in molecular biology: a Top-Down approach*. Accepted for publication in the Journal of Interactive Learning Research.

Demo sites:

[7a] downstream process designer, http://www.fbt.eitn.wau.nl/ follow content showcase, Downstream Processing Design Case.

[7b] molecular biology cases, http://www.fbt.eitn.wau.nl/ follow content showcase, Molecular Biology site.

[7c] food safety management case, contact Marc.Boncz@wur.nl for access information.

[7d] process engineering cases, http://www.fbt.eitn.wau.nl/ follow content showcase, Mixing and Membranes cases.

Acknowledgements

The authors would like to thank Hylke van der Schaaf, Tinri Wilmsen and Marc Boncz for providing access to their learning material, and discussing the requirements for LMSs.