Sharing and Re-using Scientific and Educational Resources in the Monist Learning Environment

Sören Lorenz, Markus Oesker, Wolfram Horstmann

Department of Neurobiology
Bielefeld University
Universitätsstraße 25
33619 Bielefeld
{soeren.lorenz; markus.oesker; wolfram.horstmann}@uni-bielefeld.de

Abstract: The problem of sharing and re-using existing digital scientific and educational resources within learning environments has vital importance for building an effective e-learning resource repository. Most systems lack integrative mechanisms to connect arbitrary resources to learning units. This article describes such a mechanism and opens perspectives for its use in higher education. It is implemented in the monist system, a learning and instruction environment supporting the use of simulations in higher education. Monist provides a unified educational context and supports sharing and recombination of stored learning objects. By integration of external resources the re-use of existing scientific and educational resources is fostered. Originally designed for education in neural and cognitive sciences, monist offers a general solution to the problem of sharing and re-using content.

1 Introduction

The problem of sharing and re-using existing digital scientific and educational resources within learning environments is of vital importance for building an effective e-learning resource repository. In this article, an integration mechanism is introduced to associate arbitrary resources, such as applets, applications, source code or primary data, into learning units. As an example learning environment, the freely available *monist console* [1] is used. The monist system has originally been developed to support the integration of simulations in higher education of neural and cognitive sciences. It is a java based learning and instruction system for simulations [HLE03] [LHO+04] [Die03]. Based on a client-server architecture, it provides several tools for learning, teaching and authoring. Its main focus is to support the design and use of educational simulations in a broader course context. Because of its embedded communication system, monist supports almost every educational scenario, like group-based learning, blended learning or online-learning. Most parts of the monist-console can be used online *and* offline.

The neural and cognitive sciences provide an appropriate basis for testing the practical

[1] monist - simulations for brains, has been supported by the German Federal Ministry for Education and Research - BMBF, 2001-2004, and is maintained by the monist group. For details see: http://www.monist.de.

impact of the proposed mechanism. The role of models and simulations in explaining brain functions has significantly increased in the last years [CS92] [OM00] [Hor03] and the functional context of neural and cognitive models is highly complex. There are several established modelling tools, and many simulations of established scientific models are available. Since it is very cost-intensive and time-consuming to built models and simulations, a fortiori if they are designed for education[2], multiple implementations of the same model should be avoided. To increase the re-use of existing scientific models and to facilitate the integration of existing simulations into courses, the monist system offers an innovative and effective solution, resulting in an instructional system for the educational use of simulations, simulation tools and other complex contents.

2 Sharing and Re-using Content

Since monist was designed for the neural and cognitive sciences, it was a crucial goal to enable the system to share and re-use several types of simulation resources and tools, that are commonly used in computational neuroscience and cognitive sciences as well as in neuroinformatics (e.g. GENESIS [BB98], PDP++ [OM00], NEO/NST [Rit05], to name only a few). The solution was to design monist as an *instructional system* that focuses on unifying the educational context rather than rebuilding the available scientific and educational material into single educational objects. The result, the monist console, is a kind of desktop in a desktop. Locally installed or server based applications can be executed out of a learning unit in a specific stage of the learning path and their use can be guided by instructional information in the unit. Obviously, this does not only work with simulations or simulations tools, but with any resource. The following paragraphs depict the structure of learning units, their organisation and how external resources can be embedded.

2.1 The structure of monist learning units

A learning unit is structured in pages. Every single page consists of four boxes organised in *topical text*, *instructions*, *tasks*, and *personal notes*. The boxes appear as simple formatted hypertext that may contain references to other resources by means of links and embedded objects (typically graphics, or animations). Monist provides several link types to distinct between bibliographic references, glossary entries, learning units, and external resources. Equipped with this features, monist learning units meet requirements for supporting learning with simulations, as they have already proposed by others (see [DeJ91] [Cro01], for instance).

[2] A lot of scientific simulation tools and simulations exist, that are very well suited for scientific use (graduate level and higher). But these resources often lack a well documented and ready-to-use structure that is applicable to novices. Therefore, additional information and sometimes a redesign of user interface are necessary, to transform a scientific simulation into an educational one. Hence, it may be much more cost and time intensive to design simulations for education.

Any resource used in monist - this includes learning units, and hyperlinks - is maintained using a so called *virtual learning object (VLO)* [Die03] [LHO+04], a set of customised standard metadata stored in the *monist database* (see 2.2 for details). A major part of the metadata is generated automatically. The remainder, mainly descriptive information, has to be specified explicitly by the author of any object. Any relationship between resources (containment, dependency) is determined using VLOs. From a user's point of view, learning units and other resources can be retrieved via a file-browser like tree navigation or a search dialog. VLOs come in different flavours: *internal objects* fully described by their metadata, and objects that refer to external resources, like files, applets, applications, and websites. A special type of link, called *external link*, has been defined, to trigger actions (e.g.: view, execute) on any kind of resource out of the units (see 2.3 for details). Learning units (internally managed as XML) can be exported into HTML and therefore be re-used independently from monist. This applies to individual task solutions and notes, too. However, external links will not work out of the box in a standard web browser.

2.2 Retrieving, sharing, and re-using VLOs

The variable form of VLOs allows for easy sharing and recombination of content. Once stored, existing VLOs can be linked into new units and be recombined with new VLOs. Existing units can be collected and combined to courses and may appear in different courses at the same time. The visibility of any object, from VLO to complete courses, can be assigned to different user groups, like course members, and therefore be restricted in their access. This allows for individual access to specific courses, enables authors to restrict access to some individuals, i.e. for reviewing contents, and prevents copyright violation of licensed resources, that are licensed for one group of courses or institute members, but not for all registered monist users.

The opportunities of sharing and re-using heavily depend on the possibilities of retrieving VLOs once stored in the monist database. Therefore, appropriate annotations have to be made, labelling learning objects according to their intended use. This is guaranteed by the provided monist metadata set for each single VLO as well as for compositions of VLOs to learning units. The related search function allows a full metadata search even in descriptive entries. The metadata set is based on Dublin Core (DC)[3] supplemented by a subset of educational metadata, adapted from the IMS Learning Resource Meta-Data Information Model[4] as well as by some simulation metadata[5], developed especially for monist.

The educational metadata set provides information on didactics and instructional use. Hence, it reflects the author's intention and experiences concerning the related VLO. This

[3] See http://dublincore.org/ for details; last visit: August 11, 2005.

[4] See http://www.imsglobal.org/metadata/ for details; last visit: August 11, 2005.

[5] A specialized set of metadata for simulations is under development, as proposed in [Ho02]. It allows a detailed description of the underlying model as well as of the intervention types, number of parameters, visualization types and a complexity measure. Since this work is still in progress, a detailed description of this type of metadata is not part of this paper.

set shall ensure that items stored in monist can easily be categorized by educational means. To prevent authors from being stressed by to many metadata, the set is restricted to six entries, shown in Tab. 1.

ENTRY	ATTRIBUTES	COMMENT
learning objective	[description]	Brief description of what should be learned with VLO or unit or unit
educational context	[Higher Education |University First Cycle |University Second Cycle | Bachelor |Master |University Postgrade |Professional Formation, Continuous Formation]	Level of education; implies the degree of difficulty
educational setting	[lecture |talk |seminar |tutorial | exercise |practical training]	Types of arrangements a a VLO or unit is designed for; best match first, max 5 items
instructional use	[description]	Description of how a VLO or unit may be used, i.e. its didactical use and opportunities for integration into existing courses or lectures
learning time	[single value] + [minutes |hours |days]	Approximation of duration needed to learn with VLO or unit
learning resource type	[Exercise |Simulation |Questionnaire | Diagram |Figure |Video |Graph | Index |Slide |Table |Narrative Text | Exam |Virtual Experiment |Problem Statement |Self Assessment]	Elements VLO consists of or are contained in unit, best match first, max 10 items

Table 1: Set of educational metadata provided by monist for every VLO, both, as single item or as composite unit. These metadata entries are adapted from the IMS Learning Resource Meta-Data Information Model. For details, see text.

The *learning objective* is a descriptive entry that shall give an overview of what can be learned with a particular VLO. It reflects the author's aim in creating this object. Applied to units, it additionally may serve as a topical summary. The entries *educational context* and *educational setting* classify the level of education (and implicitly the degree of difficulty) and its possible use within the specified context.

The most variable and in the context of sharing and re-using very important entry is *instructional use*. Its purpose is to describe the didactical background and the didactical context within the constraints of the entries *educational context* and *educational setting*. Additionally, a description can be added, how a particular VLO or unit may be embedded

into classical educational settings or how to combine it with other units and objects on the didactical level. Since the term *instructional use* is in a way fuzzy, and several instructional settings can be imagined for the same VLO or unit, a free text field is given instead of preselection. Appropriate guidelines are providing some examples and commonly used terms for this metadatum.

The two remaining entries assist users in estimating the effort needed to use a particular VLO or unit, both, in time and media complexity. The specification of *learning time* helps users to accurately estimate the effort for learning with the corresponding VLO. The *learning resource type* gives insights to the included (interactive) elements and media. It might be more than one in the case of a complete learning unit and implicitly gives hints to the complexity of assumed user background in media skills.

2.3 Embedding external resources

External links in Monist can point to public available resources via standard URL protocols like *http*, and *file*. Invoking this kind of link usually means to open its target using the default application registered at the respective OS for the mime type of the URL, e.g. to open a web-page in a web-browser. Plain html and text files can as well be displayed inside of the monist console.

Links pointing to resources covered by other VLOs exhibit a more complex behaviour: Files can be uploaded to the monist console. In doing so a new VLO is created, whose metadata contain entries for the *name*, and the *mime-type* of the uploaded file. External links only accept target objects of archival type, i.e. mime-types corresponding to zip- and tar-archives, and jar-files.

The creation of a link pointing to another VLO automatically declares the referenced VLO as its child object, thereby reflecting the dependency-relation. Activation of an external link triggers the following steps:

1. For each child object, its content gets extracted to a well defined location, if not already done before. Jar-files just get copied to the destination directory without extraction of contents.
2. If the extracted content contains a specially named batch-file/shell-script, this file is executed after checking execute permissions for this file. (The major goal here is to check execute/access permissions.)
3. The link target - a metadatum - is parsed into parts. The first part is taken as path to a file relative to the directory wherein the archives have been extracted.
4. If such a file exists, it is executed. Any further parts of the parsed target are appended to the execution call as runtime arguments, allowing the re-use of one VLO for several links that differ only in runtime arguments (e.g. simulations that belong to a specific tool or application and therefore require a run-time engine, like NEO/NST [Rit05], SNNAP [ZBB94] or RUBIN [LH99]). Jar-files and batch files are handled

specially: they get parsed as arguments to a java virtual machine or to a DOS-console, respectively.

Care is taken to provide an appropriate set of environment variables for execution calls. Authors can select a special call type named ”batch” to enforce the environment to be the same as the one the monist console sees itself. The latter variant hinders monist to augment the environment with some monist-specific variables, but enables applications to run, that depend on settings invisible to the java virtual machine monist runs in.

An external link in monist contains three targets: One common target, and two platform specific ones for MS-Windows, and Linux. The monist console chooses the most specific target that is not empty, depending on the operating system it is running on. The Linux-target is chosen for Linux and Unix. OSX platforms work with the common target. Some minor differences between Microsoft platforms are handled internally. This strategy of link invocation enables authors to define links to applications that work well on all major platforms, especially as the archives attached to a link can be defined in a platform specific manner, too. End users working with monist will neither notice the OS-specific differences nor will they be confronted with the details of the mechanism. For learners and teachers, external links appear and work like standard hyperlinks.

Additionally, external links can refer to archive-files not present on the local machine, but on the monist server. This is interesting for big resources that are not expected to be used by every user. Those archives are automatically downloaded on demand. End-users have the option to cancel downloads, and interrupt link invocation.

3 Practical implications for authoring and teaching

The feasibility of including external resources opens innovative perspectives on the application of learning environments and the design of course material. Four main advantages can be described for the use in higher education:

- *Single dynamic media, like Applets.* By providing metadata information for the topical and instructional context of integrated applets, these can not only be used within courses but can be shared by third party in a reliable form. This may foster their re-use by other educators.
- *Educational resources, like educational simulation repositories.* Existing resources, based on applications, like interactive electronic textbooks (e.g. RUBIN[6]) or ready-to-use simulations, based on simulation tools (e.g. NEO/NST, SNNAP) can be

[6]RUBIN - Rechnergestützer Unterricht zur biologischen Informationsverarbeitung in neuronalen Netzwerken (*Computer based Education of Biological Information Processing in Neural Networks*). An interactive electronic textbook providing about 60 educational simulations for the respective topic divided into 6 chapters. Simulations are constructed by an integrated kit, operating on an object-oriented class library. Each educational simulation can additionally be use as a standalone application, provided all classes and graphics are delivered. RUBIN was funded by the Universitätsverbund MultiMedia, NRW, Germany from 1998-1999. Its development has been maintained within the context of the monist project.

embedded in parts and in this way be re-used in different topical and educational contexts, if the appropriate run-time engine is once installed.

- *Complex scientific applications, like simulation tools.* Scientific applications are often very powerful research tools, demanding an intensive treatment of parameters and features. The design of guided tours for concrete examples can help to focus on specific tasks solving and to reduce the overall amount of time needed to gain some practice in using complex tools.

- *Connecting laboratory hardware, like measurement systems.* For example, a hardware measurement system can be controlled via monist and the recorded data can be added to the individual notes and be processed by a statistical software tool, also referred to by an external link[7].

In 2004, at Bielefeld University, the Bachelor Module "Neurobiology and Behaviour" was completely supported by the monist console. The course involves 15 days of six hours lecture and tutorial, including real world experiments and simulations, four participating departments and 40 students. More than 10 different external applications were connected to the monist console, like complex simulations and virtual behavioural experiments, videos and animations not included. Problems in using many different resources could be decreased by offering the unifying learning environment supported by monist (unpublished internal evaluation).

4 Conclusion

The proposed mechanism for sharing and re-using existing scientific and educational resources in the monist learning environment [LHO+04] opens innovative perspectives for learning environments in general. Monist is designed as an instructional system that focuses on unifying the educational context rather than rebuilding the available scientific and educational material into single educational objects. Instead, it provides features for the integration of arbitrary external resources into learning units. This is done by a special type of link, called external link, defined to trigger actions on any kind of resource out of the units. Learning units, that are associated to resources, like applets, applications or primary data, for instance, can be equipped with metadata, instructional and topical text and may be added to the educational repository. Since the repository is based on virtual learning objects (VLO) [Die03], any object can be shared, re-used and re-combined to new learning units. This process is supported by the provided monist metadata set and the related search function, to guarantee an easy retrieval of stored units. The experiences with the use of monist in bachelor courses have shown that the instructional character and the unified look of monist units help to guide through different kind of resources and facilitate computer aided learning.

[7] At the Bielefeld University, a respective learning unit was designed for undergraduate students. This unit as well as the measurement system have been developed in the context of the "Notebook University Project", funded by the German Federal Ministry of Education and Research.

References

[BB98] J.M. Bower and D. Beeman. *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* Springer, New York, 2nd edition, 1998.

[Cro01] D. Crookall. State of the art and science of simulation/gaming. *Simulation & Gaming*, 32:449, 2001.

[CS92] P.S. Churchland and T.J. Sejnowski. *The Computational Brain.* MIT-Press, Cambridge, MA, 1992.

[DeJ91] T. DeJong. (Ed). Computer simulations in an instructional context. *Education & Computing (Special Issue)*, 6, 1991.

[Die03] A. Dieckmann. *Generische E-Learning-Plattform für interaktive Lehrsimulationen zum Einsatz in Selbststudium und Präsenzlehre online und offline*. PhD thesis, Bielefeld University, 2003.

[HLE03] W. Horstmann, S. Lorenz, and M. Egelhaaf. Monist - Educational Simulations for Brains. In *Proceedings of the 29th Göttingen Neurobiology Conference and the 5th Meeting of the German Neuroscience Society 2003*, page 1052. Thieme Verlag, 2003.

[Hor03] W. Horstmann. *Explaining Brains by Simulation*. PhD thesis, Bielefeld University, 2003.

[LH99] S. Lorenz and W. Horstmannn. Das RUBIN Projekt. Technical report, Lehrstuhl für Neurobiologie, University Bielefeld, 1999.

[LHO+04] S. Lorenz, W. Horstmann, M. Oesker, A. Dieckmann H. Gorczytza, and M. Egelhaaf. The monist project - A system for learning and teaching with educational simulations. In *Proceedings of ED-MEDIA 2004, World Conference on Educational Media, Hypermedia & Telecommunications*. AACE, 2004.

[OM00] R.C. O'Reilly and Y. Manukata. *Computational Explorations in Cognitive Neuroscience: Understanding the Mind by Simulating the Brain*. Bradford Book, 2000.

[Rit05] H. Ritter. Neo/NST - a graphical simulation tool kit for artificial neural networks; http://www.techfak.uni-bielefeld.de/ags/ni/, 1994-2005.

[ZBB94] I. Ziv, D.A. Baxter, and J.H. Byrne. Simulator for Neural Networks and Action Potentials: Description and Application. *J. Neurophysiol.*, 71:294, 1994.