September 2003

SmartFrame: An Integrated Environment for XML-Coded Learning Material

Torsten Reiners
University of Hamburg, t.reiners@tu-bs.de

Stefan Voß
University of Hamburg

Dirk Reiß
University of Technology Braunschweig

Henrike Schulze
University of Technology Braunschweig

In: Uhr, Wolfgang, Esswein, Werner & Schoop, Eric (Hg.) 2003. *Wirtschaftsinformatik 2003: Medien - Märkte - Mobilität*, 2 Bde. Heidelberg: Physica-Verlag

ISBN: 3-7908-0111-9 (Band 1)
ISBN: 3-7908-0116-X (Band 2)

SMARTFRAME: An Integrated Environment for XML-Coded Learning Material

Torsten Reiners, Stefan Voß
University of Hamburg

Dirk Reiß, Henrike Schulze
University of Technology Braunschweig

Abstract: Universities and lecturers can improve the quality of education by offering virtual course material in addition to the classroom presentation. In most cases, this is either done through downloadable pdf-files or static html-pages even though there is a shift to the application of virtual learning environments. Here, it is important that the added-value of more technology increases the learning experience rather than surrounding the (static) html-pages with some gimmicks. In this paper we present SMARTFRAME, a prototype for a virtual learning environment incorporating a hierarchically structured design of XML-coded learning material including meta-data information. A demarcation to other products is based on the direct application of meta-data during the transformation process of the learning material to the desired output format as well as the integration of innovative concepts and components.

Keywords: e-learning, virtual learning environment, LOM, LMML

1 Introduction

Analyzing actual developments within the sector of virtual learning environments (VLEs), one realizes that a wide-spread variety of different VLEs exist, either freely available or proprietary ones. Some of them offer an integrated learning environment including features for collaborative work like chat- and forum-functionality or an integrated whiteboard (see, e.g., WebCT [WebCT03] and Blackboard [Black03]); others are a loose collection of interactive Java-applets or Macromedia Flash-animations (see, e.g., [SnBy03] and [Bal+03]). These VLEs have in common that they predominantly aggregate the learning material in a more or less static form, i.e., it is only available in standard html-format and lacks the ability to be (re-)used in other contexts. More differentiated approaches use XML for encoding and meta-data for describing the learning objects (see, e.g., [Met03], [Orw03], and [ReVo03]).

Being interested in developing learning material for different courses at university level that are enriched by modern and innovative didactical concepts to improve the learning experience, we develop a concept for a web-based VLE. Main aspects of our architectural concept SMARTFRAME (Smart Technology for Research and Modern Education; see [Smar03]) are its flexibility and configurability for the learner, the structured design of the XML-based learning material, and the exclusive usage of modern (standardized) technologies that are available without charges, i.e. due to open-source projects. In contrast to most VLEs, the content is stored in hierarchically structured semantic units, so-called learning objects, which are directly composed and transformed according to user-specific configurations and meta-data descriptions to the requested output format. Depending on the learning style and goal, the learner can select the form and density of the presentation. That is, if the learner repeats course units to memorize the most important facts, the learning material can be (automatically) condensed to the most relevant parts regarding the learners' needs by hiding "non-relevant" and optional learning objects.

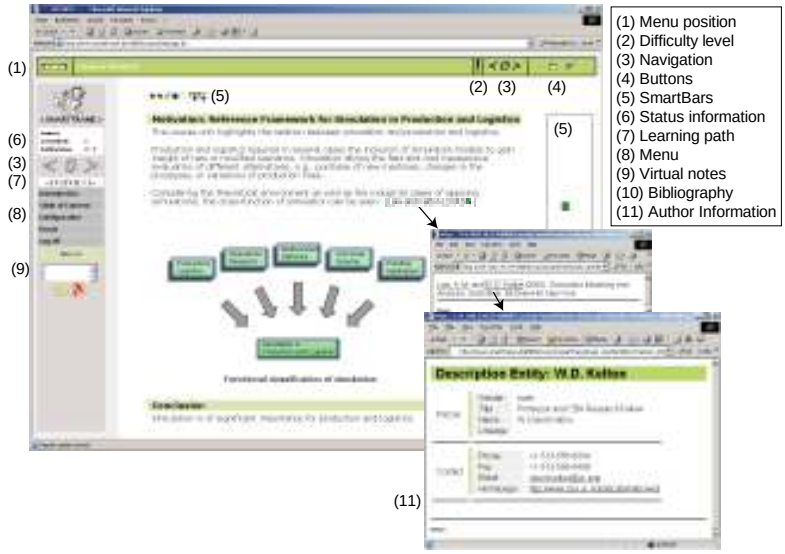

Figure 1: Screenshot of SMARTFRAME

Figure 1 shows a screenshot of the current version, whereas the development is part of an ongoing research project. The VLE consists of a control bar at the upper part containing buttons for superior functionality like navigation, activation of SmartBars, and communication, as well as a menu bar on the left-hand side. The menu bar includes status information, navigation, learning path, menu, and virtual notes. Note that the position of components is not fixed but can be on either side of the browser or completely turned off. In the following this and the underlying technology is called the graphical user interface. Figure 1 shows also the Smart-

Bars on the right-hand side, which is an innovative technology to visualize aggregated information from the learning material; see [Rei+02] for a detailed description of SmartBars and other concepts for improving the presentation of learning material. The learning material itself is displayed in the main area of the browser. The two small additional windows show a bibliographic reference and author information, which can be opened by the learner clicking on the bibliographic reference in the learning material.

The design of a course can be obtained either by a technically driven or by a didactically driven approach; see [ReVo03]. The first approach is (usually) limited regarding the incorporation of new didactical methods. Therefore, we decided to follow the second one. Subsequent to a preliminary evaluation and, therewith related definitions of the most relevant didactical methods—for the learners as well as the authors—that have to be at least part of the VLE, we designed a specific concept being realized within SMARTFRAME; see [Rei+03a] as well as [Fra+02] for the didactical background of this paper. In this paper we focus on the technical view, which allows a high degree of modularity, flexibility, and reusability due to its modular and open design.

The following technical description of the system architecture as well as the learning material will be explained in such a way that we show how the example in Figure 1 is produced. That is, we describe the structure of the learning material (Section 2.1) and give a brief overview of the content encoding (Section 2.2). In Section 2.3 we describe the meta-data concept being a further specification of the content and learning objects, respectively. In our concept we focus on the Learning Material Markup Language (LMML; see [Sues03]) and Learning Object Metadata (LOM; see [IEEE03]) and, therefore, show how these specifications can be combined and modified with respect to insufficiencies and redundancies. The technical realization of how the data is stored and the author can be supported using an authoring tool is discussed in Section 2.4. The section about the learning material is concluded by a description of the content aggregation and transformation into the user-specific presentation (Section 2.5).

In Section 3 the technical architecture of the graphical user interface of SMARTFRAME is described including the configuration and navigation. Besides presenting a concept, the technical realization is explained, i.e. how Servlets, JavaBeans, and JSPs are used to compose a VLE. The paper is concluded in Section 4 by presenting further components being integrated in SMARTFRAME as well as an outlook on future developments.

2 Learning Material

There are several choices of how to encode learning material. In this paper, we focus on different factors that allow a later reusability in various contexts, incorporation of modern technologies as well as practicability. That is, XML-based learning material is chosen due to its popularity, future perspective, and readability but also from a technological point of view, i.e. simple transformation to various output formats by using XSLT (eXtensible Stylesheet Language Transformations; see [W3C03b]), processing by freely available tools, and standard operations like full textual search. Furthermore, several organizations work on standards specifying the encoding of learning material aiming at interchangeability between different VLEs. Our concept is closely related to LMML, whereas several components had to be adopted, especially in terms of refining the underlying didactical model as well as combining LMML with the meta-data specification LOM. This section describes the learning objects and their encoding in LMML, sketches the LOM specification, our modifications as well as the concept for an authoring tool, and outlines the transformation.

2.1 Learning Objects

The reusability of learning material is improved by defining a hierarchical and modular structure of different learning objects. Each learning object is encoded as an XSP-file (eXtensible Server Pages) using the following structure:

```
(a1)  <xsp:page language="java"
(a2)    xmlns:xsp="http://apache.org/xsp"
(a3)    xmlns:smartframe="http://www.smartframe.de/xsl"
(a4)    xmlns:cinclude="http://apache.org/cocoon/include/1.0">
(a5)
(a6)    <smartframe:header/>
(a7)
(a8)    <smartframe:me identifier="id_1">
(a9)      <!-- CONTENT -->
(a10)   </smartframe:me>
(a11) </xsp:page>
```

Lines (a1)-(a4) define the type of the file as well as the used namespaces, line (a6) will force the inclusion of code, e.g., to resolve links to other learning objects, and line (a8) identifies the learning object by its granularity level (*smartframe:me* corresponds to a media element) and a unique identifier (*id_1*).

Learning objects are composed to larger units using four different types of links. The tag <*smartframe:link type="type" ref="id"*> is used with *type* being chosen from the following list:

- **referenceinternal/-external**: Another learning object, either internal (part of the local learning material) or external (all other material like web-sites in the Internet), is only referenced. That is, the object is not completely integrated but

displayed as a link using descriptive information from the object, e.g., the caption of a figure is shown as a reference instead of the figure itself.

- **includeinternal/-external**: The learning object is completely integrated, e.g., the figure is included and shown.

Depending on the content, size, and relation to other objects, we distinguish different levels of granularity of learning objects with the following meaning. Note that variants of the learning material in terms of the language are stored within the same learning object file.

- **media element (ME)** is a small unit without further partitioning, whereas it is important that other learning objects are not included but referenced as described above. Examples are text fragments, tables, figures, glossary entries, formula systems, or media objects (e.g., sounds, videos, or photos). The following example defines an object of the type image with the title (or caption, respectively) specified in line (b3) and a link to the image itself (b4). Different versions of the learning object regarding the language are distinguished using the attribute *xml:lang* identifying English (b2-b6) and German (b7-b9).

```
(b1)  <smartframe:me identifier="id_2">
(b2)   <lmml xml:lang="en">
(b3)    <image title="Areas of Simulation">
(b4)     <smartframe:link type="referenceinternal" ref="id_3"/>
(b5)    </image>
(b6)   </lmml>
(b7)   <lmml xml:lang=”de”>
(b8)    <!-- content in german -->
(b9)   </lmml>
(b10) </smartframe:me>
```

- **learning element (LE)** is a composition of objects (media elements as well as learning elements) to produce a semantic unit. That is, the content of a ”webpage” or section in a printed document. So-called pagebreaks can be used to split up the LE in smaller parts being displayed separately using standard navigation.

```
(c1)  <smartframe:le identifier="id_le">
(c2)   <lmml xml:lang="en" >
(c3)    <section title="Course in Simulation">
(c4)     <motivation title="A Motivation for the Course">
(c5)      <smartframe:link type="includeinternal" ref="id_1"/>
(c6)     </motivation>
(c7)     <smartframe:pagebreak/>
(c8)     <!-- text component not shown -->
(c9)     <image title=”Functional Classification of Simulation”>
(c10)     <smartframe:link type="includeinternal" ref="id_2"/>
(c11)    </image>
(c12)    <!-- etc -->
(c13)   </section>
(c14)  </lmml>
(c15) </smartframe:me>
```

This example shows the inclusion of two other objects as well as its adaptation. The attribute *title* of image (c9) is used instead of the originally defined title of the included object (see previous listing) and, therefore, supports the reusability in terms of adaptation to the context.

- **content module (CM)** is superior to learning elements grouping these to larger units. The content module should cover a certain content of the lecture or a special field, or include the slides of a course or seminar. The presentation to the learner should be realized as a table of contents or hyperbolic tree or graph, respectively.
- **thematic metastructure (TM)** covers a whole thematic field in the VLE and could either be the work of a department in a specific field or the courses a learner has to take in order to obtain a certain degree.

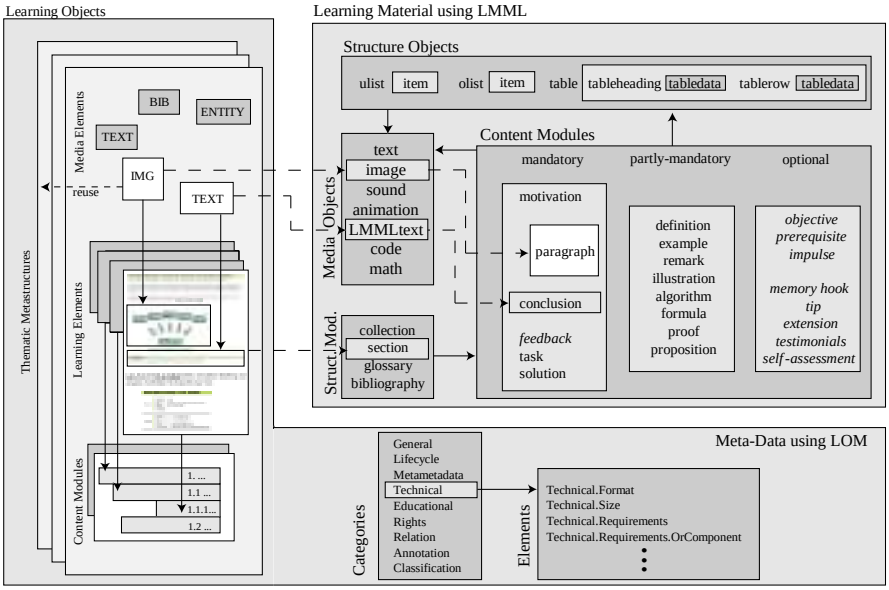

Figure 2: Structure of learning material

Figure 2 gives an overview of the learning material. On the left-hand side, the different kinds of learning objects are shown, which are used to construct the sample course unit. Note that the media element IMG is reused within another thematic metastructure. The MEs are combined within a LE, whereas the bibliographic entry is part of the LE but only shown to the user by the SMARTBARS. The learning objects are mainly built using LMML as shown for the media elements IMG and TEXT. In addition to LMML, objects like BIB and ENTITY use further specifications, e.g., DocBook [WaMu99], QTI (Question and Test Interoperability,

[IMS03b]), and SVG (Scalable Vector Graphics [W3C03a]). For a detailed description on how to encode learning objects see [Rei+03b].

2.2 Standards for the Encoding

The majority of the learning objects is coded using LMML; only non-supported types like bibliographic entries have to be based on other approaches. LMML introduces further semantical classification of the learning material by distinguishing four categories. Figure 2 shows on the upper right-hand side structure modules (used to enclose and structure the content), content modules (to compose the learning material), structure objects (to build tables and lists), and media objects to include and classify the content itself.

Our sample course unit first has to motivate the learner (*motivation*) before presenting the content in form of a textual and pictorial block (*paragraph*) that uses the media objects *LMMLtext* and *image* to include the respective content. The course unit is concluded by a content module *conclusion*. The course can be enriched by using further elements, whereas we also define new elements (presented in italics) not considered in the LMML-specification. The modularization allows a user-specific presentation according to his own configuration as well as difficulty level by selecting required objects. A learner of a beginner level might require the learning objects *motivation*, *objective*, and *prerequisite* to follow the subject, whereas the expert might only want the main definitions or theses.

In the following we describe the transformation process on the learning object *motivation*, whereas the textual content is shortened and the header of the XSP-file has been left out; see also the examples in the previous section.

```
(d1)   <motivation title="A Motivation for the Course">
(d2)    <paragraph>
(d3)     <LMMLtext>
(d4)      Production and logistics requires …
(d5)     </LMMLtext>
(d6)    </paragraph>
(d7)   </motivation>
```

In addition to text-based content in Unicode, MathML, or SVG, binary learning objects have to be used for, e.g., sounds in mp3 or pictures in gif. These objects are wrapped, specified, and referenced by another object coded in LMML using the tag <*smartframe:link*>. During the transformation process the reference is resolved into the real location of the object (LOM-database) and inserted into the output according to the mime type.

```
(e1)   <image title="Functional classification of simulation">
(e2)    <smartframe:link type=”includeinternal” ref=”id_3”/>
(e3)   </image>
```

2.3 Meta-Data

Learning objects have to provide mechanisms for authors as well as learners, allowing reusability, classification with respect to several criteria, and recovery in a large pool of objects. Therefore, each learning object is specified by meta-data as shown in Figure 2 on the lower right-hand side. Currently, several organizations develop specifications for meta-data to be used within VLEs; see, e.g., ARIADNE [Aria03], IMS [IMS03a], LOM [IEEE03], or SCORM [ADL03]. In our approach we decided to use LOM, which basically arose from the other standards and is on the verge of being published as an IEEE-standard.

Even though LMML itself also incorporates meta-data for the learning objects, it is to be preferred to use external specifications. That is, the meta-data in LMML is designed as a part of the content and, therefore, can only be used if a full textual search over all objects is initiated. Hence, we modified LMML in a way that meta-data elements are either eliminated in case of redundancy or moved to LOM by being added as new elements in the appropriate category; see the following list for some examples and [Rei+03b] for a detailed description.

Group	LMML-meta-data	LOM element
general	author	lifecycle.contribute
time	creationTime	lifecycle.contribute.date
content	language	general.language
pedagogical	difficulty	educational.difficulty
structure	type	educational.learning resource type

Table 1: Examples

Additionally, further attributes in LMML, e.g., for linking learning objects or referencing bibliographic entries, were changed due to our design of the XML-based learning material. The linkage of other learning objects using direct URLs (*target_module)* does not allow the incorporation of the meta-data during the transformation process and bibliographic entries are coded as separate learning objects and, therefore, have to be referenced in the same way as other objects (<*smartframe:link ref=”identifier”*> instead of *bibkey*).

Even though the current version of the LOM-specification reached a final status, there are still required adaptations with respect to allowing an appropriate application. Especially a major error in the category *technical* has to be corrected. In the specification, several requirements are conjunct with *and*, whereas in the requirements itself further *or*-conjunctions are allowed. Unfortunately, this does not al-

low the definition of terms following the Boolean algebra and, therefore, had to be extended by further elements that allow the adequate definition of technical requirements for learning object. The category *technical*, as we amended it, has the following elements:

Requirements

Req.{Or,And,Not}Composite

Req.{Or,And,Not}Comp.Type

Req.{Or,And,Not}Comp.Name

Req.{Or,And,Not}Comp.Min Version

Req.{Or,And,Not}Comp.Max Version

This allows the definition of requirements T with the following properties:

$$t_i=(r_{i1} \wedge \ldots \wedge r_{ik}) \wedge (r_{i(k+1)} \vee \ldots \vee r_{il}) \wedge (\neg r_{i(l+1)} \wedge \ldots \wedge \neg r_{im}) \quad i=1,\ldots,n$$

$$T = t_1 \vee \ldots \vee t_n$$

Furthermore, the meta-data should correspond with the type of the learning objects. That is, not all meta-data seems to be useful to be set—if possible at all—to a default value. For example, the duration is necessary for movies but can and should be left empty with pictures, the interactivity of bitmap-graphics is generally low, and the semantic density of bibliographic entries is high. This can be considered by changing the cardinality of elements in respect of the learning object type; see [Rei+03b] for an extensive description.

2.4 Authoring Tool and Storage in Databases

Besides the great advantages of modularity, reusability, and the superior specification of learning objects with meta-data, the necessary labor for an author who has to determine and input all the required information exceeds the tasks for composing simple html-pages by far. Therefore, we implemented an advanced authoring tool that can be used to enter both the learning objects as well as meta-data in an intelligent way. That is, the authoring tool—configured by XML-schema-specifications of the LOM-categories and the learning objects as well as further XML-specifications about relations of elements—automatically sets all elements to a default value according to the type of each learning object that can also be based on the input of other elements. This allows a flexible and effective development of learning material that includes the necessary information for an improved learning experience; see Figure 3 for a screenshot showing the current authoring of meta-elements in the category *technical*.

Figure 3: Screenshoot of the authoring tool

Most of the data is encoded in XML suggesting the usage of XML-databases. Nevertheless, there are currently major disadvantages: XML-database-systems are either very expensive (and, therefore, contradict our concept of using freely available software) or not comparable to relational database systems regarding speed and stability. Therefore, we store the meta-data in a relational database using a data model based on a transformation from the XML-structures; see, e.g., [Schö03]. Due to the time consuming SQL-queries that have to be done for each learning object involved in the transformation process we adapted the resulting model with respect to efficiency. The learning objects themselves are not stored in the database but as files using the native file system. The location of the files is part of the meta-data such that every access to learning objects includes the meta-data information. Basically, the learning material can be distributed on the network, but in terms of speed a local storage from the learners' point of view is preferred and also supported by replication mechanisms.

2.5 Transformation

The XML-based learning material is transformed to, e.g., traditional printed scripts or web-pages. The transformation process of XML- and, especially, XSP-files is done with Cocoon [Apa03a], a specialized software running as a web-application under the Tomcat Servlet Engine [Apa03b], using XSLT (eXtensible Stylesheet Language for Transformation). In this section we demonstrate the transformation of the described learning elements into a web-based presentation; see Figure 4, especially the numbers in brackets. Based on the users' request, e.g., *http://www.smartframe.de:8080/cocoon/main.xsp?identifier=id_1*, the XSP-page *main.xsp* (see the following listing) is called with the identifier for the requested learning object as a parameter (1). The file *main.xsp* is used to initiate the trans-

formation to a html-page by executing the link-tags <*smartframe:link …*/> with the identifier as a parameter, which then will be further processed by logicsheets.

```
<!-- XSP-header -->
<smartframe:header/>

<html>
  <body onload="initialize()">
     <smartframe:link>
        <smartframe:identifier>
          <xsp-request:get-parameter name="identifier"/>
        </smartframe:identifier>
     </smartframe:link>
  </body>
</html>
```

Logicsheets are used to transform specific tags (e.g., <*smartframe:header/*>) into markup language code-embedding directives, thus allowing the execution of Java-code while keeping the logic and the content separated. The following extract from the logicsheet shows the template that is matched with and used instead of the tag <*smartframe:header/*>.

```
<xsl:template match="smartframe:header">
  <xsp:structure>
     <xsp:include>smartframe.DBInterface</xsp:include>
  </xsp:structure>

  <!-- code to read meta-data information from DB like location-->

  <xsp:logic>
    public String elementName;
  ...
```

Another template is applied for the tag <*smartframe:link ...*>, which reads the required meta-data from the database to perform at least a transformation, i.e. the location as well as mime-type of the referenced object. Depending on the type and context, further information has to be requested from different categories, e.g., the requirements from *technical*, the difficulty from *educational*, or the kind of relation from *relation*. Herewith, we are capable to directly react on the user and, therefore, influence the learning experience, e.g., by presenting learning material according to a specific learning level and qualification.

The corresponding XML-files as specified by the location are included; see the pipeline in Figure 4 on the upper right-hand side. Further links to other objects are processed recursively (2, 9) subsequent to the extraction of the sub-documents in the user-specific language (7, e.g., the content of the <*lmml xml:lang=”en”*>-tags); if not available a default language is used. The extracted sub-documents are serialized (10), i.e. converted to XML, and combined to a single document containing all requested learning objects of the same language.

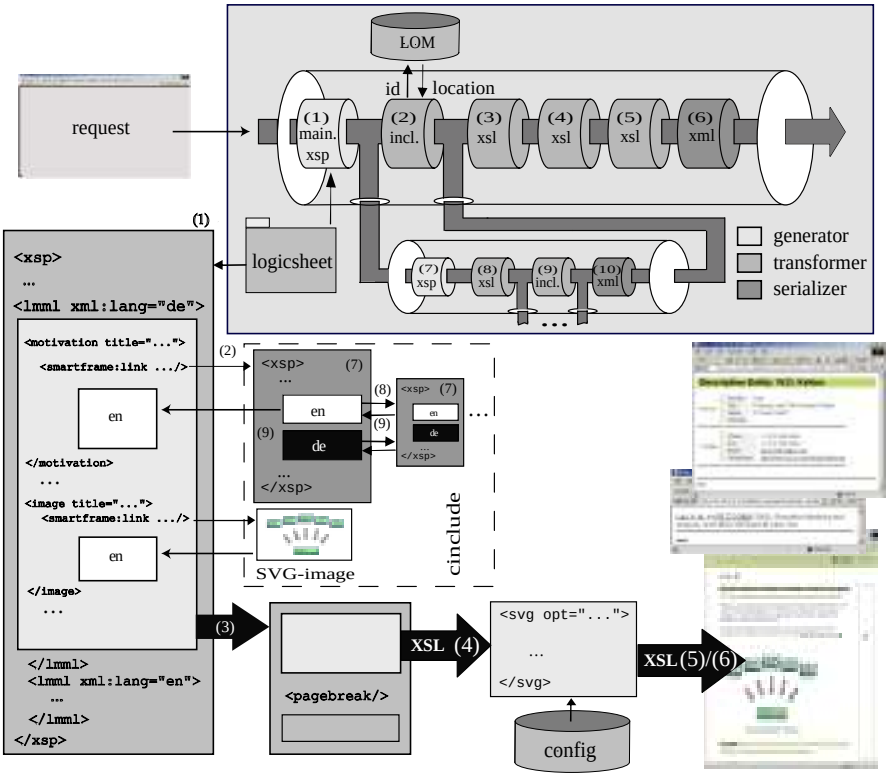

Figure 4: Transformation process of learning material

In the next step the document is analyzed for pagebreaks, and, if needed, decomposed in sub-documents—part of the semantic unit to be displayed on one page—whereas the corresponding part is specified with another parameter (3). Within two further steps the XML-document is transformed to the output-format. The predesign- (4) and design-phase (5) are used for a two-step process, whereas the first one prepares certain structures for the next phase, e.g., inserting Javascripts to be used by the learning objects. In the second phase, the settings of the user are accounted. That is, the user decides, which learning object is shown and in which ways. For example, the object of the type bibliographic entry can either be displayed within a table or as a list of references. The result of the transformation pipeline is serialized according to the requested and from Cocoon supported output format, e.g., for a pdf-document or a web-page.

The learning object *motivation* of our example is transformed to the following html-code:

```
<h3 class="me_text_motivation_titlebar">
```

```
    Motivation: Reference Framework for Simulation in Production
    and Logistics
</h3>
<p class="me_text_motivation_content">
    This course unit highlights the relation between
    simulation and production and logistics.
</p>
```

The design regarding font-type, color of the text, or formatting is defined in user-specific cascading stylesheets (css) that may be edited by the user through an interface and referenced in the html-page by

```
<link rel="stylesheet" type="text/css"
      href="http://www.smartframe.de:8080/cocoon/smartframe/
            static_content/css/treiners.css"/>
```

The following code shows an extract for the title of the motivation being shown with a greenish background and bold black-colored verdana font slightly larger than standard. Further design features are realizable, e.g., a colored frame if a table for the motivation with the corresponding coloring of fields is used.

```
h3.me_text_motivation_titlebar
{
   background-color:#dbe8ac;
   color:#000000;
   font-size:110%;
   font-family:verdana;
   font-weight:bolder;
}
```

Referenced learning objects that are not completely included in the web-page require a special processing. At first, the displayed name for the reference is constructed from the learning object depending on its type as well as the user-specific configuration. In our example (see Figure 1) the bibliographic entry is shown on the web-page as *[Law, A.M. and W.D. Kelton (2000)]* with a small picture of a book. To increase the flexibility of such references, <*span*>-tags with different attributes are used. Depending on the stylefile the display can be changed from showing to hiding the references. Furthermore, these tags can be read by the SmartBars to aggregate the information. For simplification, the Javascript-code for *showBibliography* is not quoted in the following example.

```
<span id="bib" class="hideBib">
   [Law, A.M. and W.D. Kelton (2000)<img
      src="/cocoon/static_content/images/bib.gif"
      class="hideBib"
      onClick="showBibliography('main.xsp?identifier=ident_4')"/>]
</span>
```

3 Graphical User Interface

The graphical user interface (GUI) is built using JavaServer Pages (JSP) for the presentation of the user interface in a browser and Servlets, simple Beans (no Enterprise JavaBeans), and Java classes to implement the application logic. Furthermore, we use a JDBC interface to a relational database (here, we use the free of charge database MySQL), a connection pool as an interface between the database and the user, and mechanisms to handle XML-files. These XML-files are on the one hand used to configure the GUI as well as the underlying logic, on the other hand they contain user-specific settings, e.g., for the transformation process of the learning material.

In the following sections the architecture of the GUI, the configuration of all components, the interface to the learning material as well as the navigation within the selected learning units are described.

3.1 Composition of SMARTFRAME

SMARTFRAME is designed in a 3-tier-architecture (see Figure 5) with a presentation tier, a business tier with business objects (mostly beans) running on a Tomcat Servlet-engine, and the data storage tier, currently a relational database and XML-files using the native file system. The concept considers thoughts about separating business logic from presentation logic, but also extensibility and scalability were of particular importance. Furthermore, SMARTFRAME is executed in the context of Cocoon such that a communication with the transformation process of the learning material is enabled; see Section 3.2.

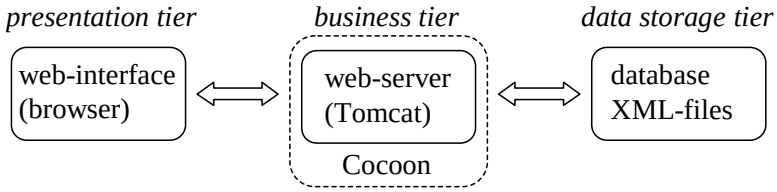

Figure 5: Architecture of the web application

The visualization is done by JSPs, and to guarantee a consistent visualization of SMARTFRAME, they all use the same user-specific stylesheet and are composed using a frameset. That is, the four components as shown in Figure 6 (i.e. title, control bar, menu bar, and content definition) are generated by a specific JSP. This JSP is configured by an XML-file containing *views*, i.e. mappings between the four components and corresponding JSP-files. For example, if the content-page is requested, a menu with all items, a control bar, and a content page with the selected learning unit are shown. The advantage of this design lies in the independ-

ence between the JSPs such that each can easily be replaced or renamed by modifying the configuration file; which can also be easily changed to create new mappings.

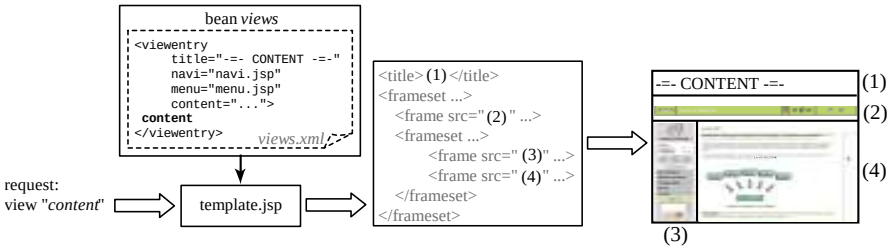

Figure 6: Dynamic frameset generation using *template.jsp*

Within SMARTFRAME certain actions, e.g., identifying the user during the login process (*CheckLoginAction*), initialization of user-specific settings (*InitAction*), or navigation on the learning path (*next* or *previous)*, have to be performed. This is done by encoding a URL with the specific action (i.e. the action name with an appended *.do*, e.g. *next.do)* calling a Servlet where the action is mapped to the corresponding method; see also the Jakarta Struts Framework for a similar concept [Apa03c]. Each action either returns a view being displayed as the result of the action within the browser or returns a further action to be executed. The latter one is repeated until the return value is a view. Like views, actions are mapped to Java-classes within an XML-configuration file. Thus, new actions can easily be integrated into SMARTFRAME by adding a combination of a unique name for the action and class reference into the configuration file.

Several beans are used to store data for the period of a session, e.g., user-specific data required to generate the GUI. This concept allows reacting on the user. That is, a bean is triggered to transfer the current data to a database or XML-file whenever the user applies a modified setting of SMARTFRAME.

3.2 Configuration

We have to distinguish several configuration settings. First, global settings for the GUI include the previously mentioned mappings for action-classes and views, color schemes, language, menu structure as well as information about the database access and location of files. Second, user-specific configurations for the GUI cover the preferred language, the position of the menu, the chosen color scheme, and user-data (email-address, username, and password). Furthermore, the user specific configuration concerning the learning material has to be specified and stored in an XML-file being accessible by the transformation process described in Section 2.5. This involves information about, e.g., the incorporation of the Smart-

Bars, the kind of display for bibliographic entries, or how SVG-images are displayed—either directly or being converted into static binary image like gif.

A Servlet initiates beans with "*scope=application*" (i.e. the bean is available application-wide) after the Tomcat Servlet-engine is started and the first request submitted. That is, XML-based configurations for the SMARTFRAME are read and applied to the interface. In Figure 7 the configuration and communication process from the login to the presentation of the learning unit is shown. The user gets the login view being asked for the login name and password. After providing the information, user-specific settings are read either from the database or XML-files and verified with the password (1-3). During the initialization process the temporary file *userconfigfile* to pass configuration information to the transformation process within the Cocoon process is generated. Furthermore, two attributes are stored in the session object: the location of the *userconfigfile* and the CSS-file for the final presentation of the learning material (4). Afterwards, the welcome screen is shown including the menu and control bar (5).

Selecting the menu item *Table of Content* presents a list of available course units (6) from which the user can select the desired course (7). Based on the identifier associated with the first learning element in the learning path, the transformation process as described in Section 2.5 is initiated (8). The resulting web-page—generated according to the user-specific preferences in the configuration file *userconfigfile* and incorporating a reference to the individual CSS-file—is shown in the content frame (9).

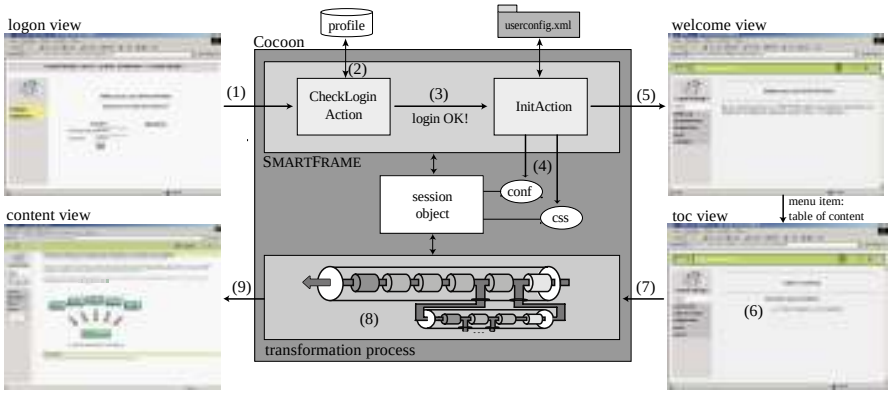

Figure 7: Configuration and communication between the GUI and the learning material transformation within Cocoon

3.3 Navigation and Browsing

The navigation within the learning material is insofar crucial that most VLEs only support linear learning paths and no explorative browsing. Our approach incorpo-

rating learning objects allows a larger degree of freedom in designing different views of the learning material. Besides the inclusion of different types of learning objects, the user can flexibly decide on the learning path, i.e. selecting excursions or changing the learning paths where these intersect. Intersections of learning paths especially highlight the relation between different courses supporting the interdisciplinary teaching at universities.

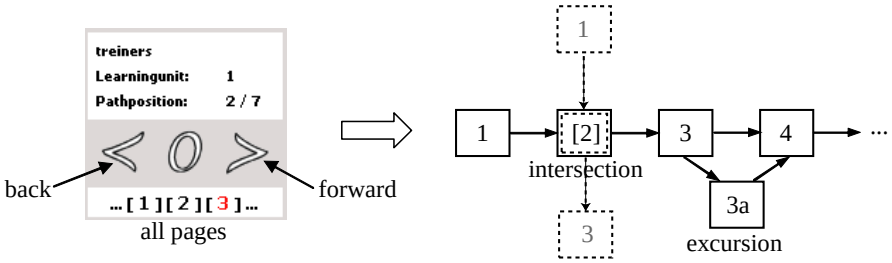

Figure 8:Navigation and learning path

Figure 8 shows the navigation components of SMARTFRAME. Besides going backward and forward on the linear learning path using the arrows, the user can directly select a particular page on the numerical view of the learning path. Visited pages are shown in a different color allowing the user to receive an overview of already visited pages. The button in-between the arrows is used for reload, return from the configuration menu or other views to the current page, and to exit an excursion path to continue on the original learning path.

The navigation is improved using the action mechanism as described before. Depending on the users profile the next button can either show the next learning element on the learning path independent from the status (visited or unknown) or jump to the next so far unvisited page. Furthermore, learning elements might consist of several parts being separated by <*smartframe:pagebreak/*>-tags. In this context, the next button could induce the display of the next part of the learning element. Note that this implies a consideration of the pagebreaks compared to a complete display of the learning element.

4 Conclusions and Outlook

In this paper we have described SMARTFRAME as a prototype of a virtual learning environment, which consists of several components, namely the transformation process of the learning material into different user-specified formats like html and pdf, the graphical user interface as well as an authoring tool. Here, we should emphasize that the transformation process itself could easily be incorporated in various environments like proprietary learning environments. That is, the transforma-

tion process requires information about the user-configuration passed in XML-files but could also use default settings resulting in static learning material presentation. On the other hand, the communication between the user interface and Cocoon could also be established using parameters appended to the invoking URL but would be fairly limited due to restrictions concerning the ability and allowance to adapt a proprietary learning environment.

These environments would have to be capable of managing some kind of linear (or arborescent) learning path structure to invoke the appropriate learning element and display the result within a separate frame. This is, e.g., given in WebCT [WebCT03] where the learning elements generated by Cocoon can be referenced by URLs like *http://www.smartframe.de:8080/cocoon/main.xsp?identifier=id_1*. Besides its more general functionalities, the user interface of SMARTFRAME can also be used to display simple static html-pages. Both alternatives are significantly lacking the advanced features that result from the collaboration of the GUI and Cocoon.

Having implemented both, the GUI as well as the transformation of the learning material, within the same technological context using the Apache Tomcat server allows a much greater flexibility. Both components share the same session object so that the user interface is able to assign attributes to it, which can be read out by Cocoon and vice versa. Having the graphical user interface handling administrative tasks like user management and preferences concerning, e.g., technical and didactical aspects, the learning material can be displayed in a user-specific way. The user is able to choose his favorite color-scheme, select his preferred language, and determine whether and where menu bars and components like SmartBars are displayed. Furthermore, the composition of the displayed learning material with respect to LMML-elements can be adapted.

The concept of SMARTFRAME allows the integration of innovative technologies, especially SmartBars and hyperbolic graphs [Rei+02]. This is due to the modular design of both technologies providing several interfaces for integration. In particular the transformation process, which might have to be adapted according to the requirements of new components, is extensible by integrating new transformation rules regarding the configuration of the user.

SMARTFRAME is part of an ongoing research project and, therefore, has to be seen as a prototype missing several essential components to be a fully featured virtual learning environment according to various notions. Mainly communication features including email, message board, or whiteboard, are currently under development. That is, we do not seek for a rudimental implementation just supporting the basic features but develop innovative functionality that, first, increases the learning experience and, second, enables an individual support of the learner during the course being comparable to traditional classroom courses.

On the other hand, the modular design of the learning material should be used to implement different scenarios like the application in the classroom, the self-paced

study as well as composing of printed material. Here, synchronized blended learning can be seen as a potential application of SMARTFRAME where several didactical methods are combined within one environment; see also [Fra+03].

References

[ADL03] Advanced Distributed Learning Initiative: SCORM. http://www.adlnet.org. Date of last check 2003-05-10.

[Apa03a] Apache Software Foundation. Cocoon software. http://xml.apache.org/cocoon. Date of last check 2003-05-10.

[Apa03b] Apache Software Foundation. The Jakarta Site – Jakarta Tomcat. http://jakarta.apache.org/tomcat/index.html. Date of last check 2003-05-10.

[Apa03c] Apache Software Foundation. The Jakarta Site – Struts. http://jakarta.apache.org/struts. Date of last check 2003-05-10.

[Aria03] ARIADNE Foundation for the European Knowledge Pool: ARIADNE Educational Metadata Recommendation - V3.2. http://www.ariadne-eu.org/en/publications/metadata/ams_v32.html. Date of last check 2003-05-10.

[Bal+03] Balci, O.; Gilley, W.S.; Adams, R.J.; Tunar, E.; Barnette, N.D.: Online Interactive Modules for Teaching Computer Science. http://courses.cs.vt.edu/~csonline. Date of last check 2003-05-10.

[Black03] Blackboard: Blackboard. http://www.blackboard.com. Date of last check 2003-05-10.

[Fra+02] Frank, C.; Kassanke, S.; Suhl, L.: Meeting students expectations and realizing pedagogical goals within the development of a virtual learning environment. In Proceedings of Elearn, World Conference on E-learning in Corporate, Government, Healthcare, & Higher Education, pp. 1124-1129, Montreal, Canada, 2002.

[Fra+03] Frank, C.; Reiners, T.; Suhl. L.; Voß, S.: Conceptual Framework for Synchronized Blended Learning on the Web. Technical Report, University of Paderborn. 2003.

[IEEE03] IEEE Learning Technology Standards Committee: Standard for information technology: Education and training systems – learning objects and metadata. http://ltsc.ieee.org/wg12. Date of last check 2003-05-10.

[IMS03a] IMS Global Learning Consortium: IMS Learning Resource Meta-data Specification. http://www.imsproject.org/metadata/index.cfm. Date of last check 2003-05-10.

[IMS03b] IMS Global Learning Consortium: IMS Question & Test Interoperability Specification. http://www.imsproject.org/question. Date of last check 2003-05-10.

[Met03] Methodenlehre-Baukasten. http://www.izhd.uni-hamburg.de/mlbk/home.html. Date of last check 2003-05-10.

[Orw03] OR-World. http://www.or-world.com. Date of last check 2003-05-10.

[ReVo03] Reiners, T.; Voß, S.: Some Thoughts on how to Teach OR/MS. Technical Report, University of Technology Braunschweig. 2003.

[Rei+02] Reiners, T.; Reiß, D.; Voß, S.: Using Hyperbolic Trees and SmartBars within Virtual Learning Environment Concepts. In Proceedings of the World Congress Networked Learning in a Global Environment, Challenges and Solutions for Virtual Education (NL 2002), Volume #100029-03-TR-026, Millet Alberta, pp.1–7. ICSC-Naiso Academic Press. [ISBN: 3-906454-31-2], 2002.

[Rei+03a] Reiners, T.; Reiß, D.; Sassen, I.; Voß, S.: Simulation und Meta-Heuristiken als Bestandteil quantitativer BWL-Ausbildung: Von der Technik zur Anwendung. Accepted for publication in Proceedings of the Workshop E-Learning in Wirtschaftsinformatik und Operations Research, 2003.

[Rei+03b] Reiners, T.; Reiß, D.; Voß, S.: XML-basierte Kodierung von Lernobjekten. Technical Report, University of Technology Braunschweig. 2003.

[Schö03] Schöning, H.: XML und Datenbanken: Konzepte und Systeme. Hanser, München, 2003.

[Smar03] SMARTFRAME. http://www.smartframe.de. Date of last check 2003-05-10.

[SnBy03] Sniedovich, M.; Byrne, A.: tutOR. http://www.tutor.ms.unimelb.edu.au/frame.html. Date of last check 2003-02-01.

[Sues03] Süß, C.: Learning Material Markup Language. http://www.lmml.de. Date of last check 2003-05-10.

[W3C03a] W3 Consortium: XSL Transformations, http://www.w3.org/TR/xslt. Date of last check 2003-05-10.

[W3C03b] W3 Consortium: Scalable Vector Graphics, http://www.w3.org/TR/SVG. Date of last check 2003-05-10.

[WaMu99] Walsh, N.; Muellner, L.: DocBook: The Definitive Guide. O'Reilly, Beijing, 1999.

[WebCT03] WebCT: WebCT.com. http://www.webct.com. Date of last check 2003-05-10.