An advanced interactive language teaching platform

Sven Radde[*,1], **Liubov Gordienko**[2] and **Burkhard Freitag**[1]

[1] Institute for Information Systems and Software Technology (IFIS), University of Passau, Innstraße 43, 94032 Passau, Germany
[2] Language Centre, University of Passau, Innstraße 40, 94032 Passau, Germany

Russian Online is an innovative e-learning platform, jointly designed and implemented by the University of Passau's IFIS institute and its Language Centre. It extends the traditional classroom language courses by allowing students to train their reading comprehension skills in private study at the PC. Our approach combines advanced teaching techniques for autonomous learning with a unique technology to deliver the learning content to the students. We give a well-rounded overview of the whole project, highlighting the technical and didactical key concepts, sketching organizational issues and sharing our experiences during real use of the platform for language courses. Other similar projects will be able to use our ideas and experiences to benefit during their planning or development stages.

Keywords e-learning; teaching/learning strategies; web-based learning; authoring tools

1. Introduction

Russian Online is a joint development of the Institute for Information Systems and Software Technology and the Language Centre of the University of Passau. It is designed to stimulate and promote the reading comprehension skills of students in advanced Russian language courses. Students are presented eleven chapters ("learning units") of authentic Russian text material about study- and job-relevant topics intermixed with various kinds of exercises, designed to deepen their involvement in the content and to provide immediate feedback about their learning success. Numerous suggestions for effective autonomous learning are embedded in the learning units, such as reading strategy hints, vocabulary exercises and intercultural footnotes. We selected the material to continually convey knowledge about Russian culture and applied geography. The chapters are arranged in the fashion of a "Reader's virtual journey across Russia" to provide a general theme to the whole content.

The application uses Java as the underlying programming language and relies on a central relational database server to store the learning units and all other persistent data. Using JSP/Servlet technologies, we provide a web-based portal as the central access-point to the platform. From there, students can launch the learning units, while tutors can edit and organize the material and have access to comprehensive statistics. Both the students' "player" application and the authoring tool are dedicated Java programs, downloaded and executed directly on a user's PC via the Java Webstart technology. Using this two-layered approach, we preserve an easy and intuitive entry into the application, while availing ourselves with the full power of the Java programming language and the Swing user-interface for the actual display and execution/authoring of the learning units. The authoring tool closely resembles the player application, providing convenient WYSIWYG editing of both texts and exercises.

A current trend in e-learning is the increasing usage of Learning Management Systems (LMS) such as Moodle and Clix Campus. These systems mainly provide a download-platform for a variety of externally created files, most commonly slideshows or ready-to-print lecture scripts, and, to a varying degree, other community-functions. However they do not directly integrate authoring tools for the internal creation of interactive content. In contrast, there are tools like Macromedia Authorware that excel at creating web-based courseware but do not focus on providing an access infrastructure such as a central portal to serve the content and aggregate statistics. Our approach combines a web-based portal including user-/course-

[*] Corresponding author: e-mail: Sven.Radde@uni-passau.de, Phone: +49 851 509 3188

© FORMATEX 2006

management and statistics functions with integrated authoring and playback applications that are particularly designed following didactical concepts for reading comprehension and private study.

The rest of this paper is organized as follows: Section 2 details the didactical concepts used in creating the material and section 3 discusses various technical aspects of the application. We conclude with a short evaluation and summary in section 4, including an outlook to current developments.

2. Didactical concepts

2.1 General concept

The main goal of *Russian Online* is the systematic improvement of reading abilities in advanced Russian language classes. Relying on available basic language skills, we strive to accelerate the development of reading competence in addition to the regular language courses. Reading is a complicated task in itself that is influenced by many different kinds of knowledge, abilities and psychological conditions [1]. A human's reading behaviour is strongly influenced by a number of tightly interconnected factors, e.g. text-types, reading-goals and reading-styles. To train these different styles of reading in a foreign language, it is therefore important to use many different and, if possible, authentic text samples. In *Russian Online*, we offer a great collection of exercises with a variety of goals to the reader. This allows a student to select and improve on the learning strategies that are individually best suited to him or her.

The course material as a whole is designed in the fashion of a virtual journey across Russia. The entry-page shows a map of Russia with eleven cities, each of which represents a chapter that may be entered by clicking on the city icon. Each chapter has its relative difficulty indicated by a varying number of stars next to the icon and focuses on a particular topic. We subdivided each of the eleven learning units into 4-5 subchapters that treat different aspects of the topic. To give an example, the chapter "Accommodation" consists of the subchapters "Living with a host family", "Living in a students dorm", "How to find an apartment to rent" and "Residential market in Russia". The texts follow a progression in difficulty of language, methods and content and we chose them in a way that they convey knowledge of applied geography and culture at the same time. When working on a learning unit, students can follow the intended linear sequence but may also freely choose the next text or exercise using a tree-based menu.

2.2 Design of an individual learning unit

Each learning unit contains a number of text passages along with preparatory ("*pre-reading*") and reinforcing ("*post-reading*") tasks. For smaller text passages such as signs, placards or statistical diagrams, we usually omit the preparatory tasks and limit the post-reading tasks to one or two exercises to train word-spotting within the text ("scanning"). For larger and more complicated text passages, we demand the alternating use of different reading styles, with the work being done in several steps [1]:

We use *pre-reading tasks* like "think about the caption" to activate the reader's world knowledge and to encourage building first hypotheses regarding the text's content (see Fig. 1a). These first considerations create curiousness and facilitate further progress with the text if confirmed. In the other case, further reading corrects the assumptions and adapts them to the new information. Exercises to deduce a word's meaning followed by verification exercises are also used in this preparatory phase. Here we offer the student several strategies during different text passages, using e.g. synonyms/antonyms, rephrasing or context. The goal of this systematic practicing of deduction techniques is to teach the student to use these techniques autonomously, enabling him to resort to knowledge in his mother tongue or other already known foreign languages to master unknown Russian vocabulary.

Now follows the actual reading of the text, guided by special *while-reading tasks*. Reading-hints before the text suggest a certain style of reading, e.g. selectively, intensely or extensively. Our underlying goal with the reading-hints is to demonstrate to the student that the purpose of reading requires a certain reading style. The application offers additional visual guidance to the student by labelling each exercise with a difficulty level and colouring the navigation tree items depending on their type (e.g. to differentiate between vocabulary and grammar exercises etc.).

© FORMATEX 2006

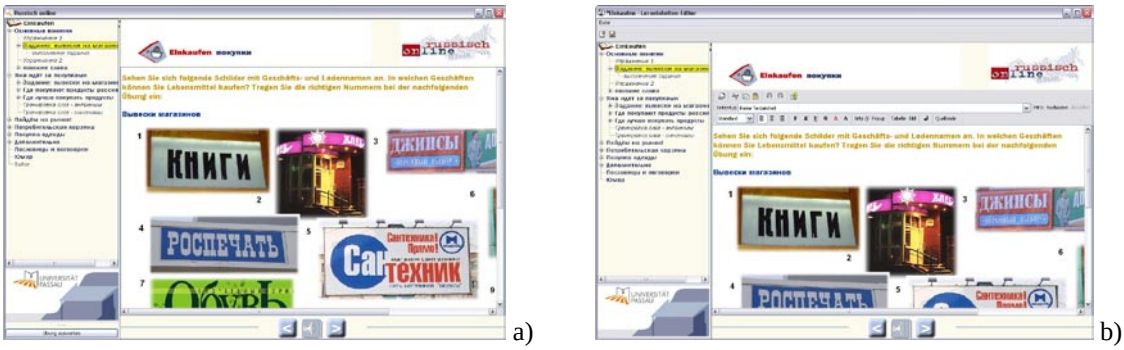

Fig. 1 a) Pre-reading exercise in which the student is supposed to identify signs related to food stores in preparation of a learning unit dealing with shopping. **b)** The same exercise in the authoring tool.

During the *post-reading tasks*, students verify their correct understanding of the text, e.g. by summarizing the most important points of the text, restoring text parts into their logical ordering or answering questions about the content. This requires active dealing with the text and trains different learning styles such as global, analytical and selective understanding. We use these techniques in an alternating and flexible way. For example, only a cursory reading may be required at first, followed by an in-depth or even analytical reading of certain particularly relevant text parts. In this case the reading is not done linearly but the student rather 'jumps' to different points in the text to find the searched information.

2.3 Usage of the exercises

Distinctive about the exercise types in *Russian Online* is the fact that, for the most part, they can be corrected automatically by the software. Available types are gap texts, matching/linking exercises, marking exercises, multiple choice tests and text puzzles amongst others. Although the number of exercise types is limited, they allow many variations so that we have effectively very many types of different exercises available.

As the feedback of the automatic correction mechanism is essentially limited to "correct" / "incorrect", we must take this into consideration when phrasing both the exercise task and the solution. Each exercise has a score that is retained in a statistic. Continued evaluation of the statistics displays the learning progress of a student in a compact form and provides clues to teachers as to possible unclear wording of exercises and the like.

3. Technological concepts

3.1 Architecture overview

Russian Online is designed as a Client/Server application, implemented with the Java programming language. Students and tutors use their web-browsers to access a portal website rendered by Java Server Pages and Java Servlets. The learning units themselves are launched by downloading and executing a dedicated "player" application via the *Java Webstart* [2] technology. This application communicates with several Servlets to receive the learning unit's content and to transmit the results. Note that the authoring tool works analogously to the player application. Fig. 2 shows an outline of this scheme, including details of the server-side architecture (cf. 3.2).

© FORMATEX 2006

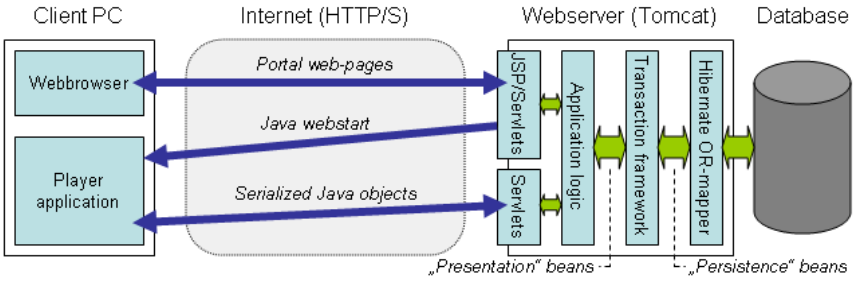

Fig. 2 Architechtural overview of the *Russian Online* application and server-side abstraction-layers.

The communication between the client browser and the web portal is handled via HTTP, while the use of SSL-encryption is possible and encouraged. The player application uses a HTTPS connection to transfer serialized Java objects of the learning units and the results. This greatly simplifies the handling of the learning units, as their representation on both the client- and the server-side is exactly the same and their transmission over the network (including serialization and re-assembly) is completely transparent to the programmer.

3.2 Server-side MVC architecture and layers

The web portal software uses the *Jakarta Struts* framework to create the user interface and therefore follows a strict Model-View-Controller pattern [3]. Java Server Pages, annotated using the Taglibs of the Struts framework, form the View, while the Struts actions represent the Controller. In addition, the player application (cf. 3.3) as a whole forms a part of the View (from the server-side viewpoint), the Servlets it uses for communication being another part of the Controller.

We use so-called "*presentation beans*" to form the model. These beans represent the object-oriented modelling of learning units (organized into text-pages and exercises) and contain the necessary application-logic to e.g. verify a student's answer (in the case of an exercise). To ensure complete independence of the application from the underlying persistent storage technology, the presentation beans do not contain any database-specific code. For storing and retrieving objects from the persistent storage (most commonly a relational database), we developed a specialized transaction framework (cf. Fig. 2).

This framework hides the actual implementation of the persistent storage and is able to guarantee the common ACID properties of transactions independently from the used storage. A central function of the framework, apart from providing ACID transactions, is the transformation of the presentation beans to "*persistence beans*" and vice versa. The latter do merely store the state of a presentation bean but do not contain application logic. Depending on the used storage system, they may be very similar to the presentation beans, as is the case for our current implementation that uses the object-relational mapping framework *Hibernate* [4]. In this case, the transaction framework is able to delegate much of the desired functionality directly to Hibernate and the persistence beans contain the necessary "mappings" for Hibernate to persist the objects.

Note that using this paradigm of a strictly separated persistence layer makes a change of the underlying storage technology completely transparent to the application. Hibernate itself would already hide a change of the relational database software, but it is also imaginable to change the object-relational mapping software or to use a completely different approach, e.g. XML files.

3.3 Client-side technologies

We chose to "leave" the web-based portal for the actual display and execution (as well as authoring) of the learning units. The design of a dedicated Java player application provided us with a very expressive and – in comparison to a realization in HTML – simple way to display a learning unit's content and exercises, correct answered exercises automatically and enforce timing-restrictions. All this is possible

© FORMATEX 2006

without the need of a permanent internet connection: Online access is only necessary to download the learning unit and to transmit the results of the exercises back to the server upon completion of the chapter. Another advantage of this approach is the presence of a platform-independent execution environment, unimpeded by browser-incompatibilities and relatively secure against manipulation.

The player application contains the necessary classes to build a GUI from the presentation beans representing the learning unit (cf. 3.2). Again, we took care to avoid any UI-specific code in the presentation beans so that replacing the rendering engine is possible without affecting the rest of the application. This conforms to the point of view that the whole player application may be interpreted as the "view" part of an MVC design, while it is organized as an MVC application in itself (with the presentation beans again forming the Model and Java Swing components for Controller and View).

The authoring tool has been developed as a natural extension of the player application that only adds the necessary user interfaces to allow modification of texts and exercises. This principle is reflected in the object-oriented modelling of the authoring tool: For every Java class responsible for GUI representation (e.g. `GapTextExerciseGUI`) there exists a corresponding subclass that adds the necessary editing functionality (say, `EditableGapTextExerciseGUI`). The result is that the authoring tool closely resembles the player application in appearance and gives intuitive results when editing a learning unit (see Fig. 1a & 1b). A preview function is included to provide true WYSIWYG inspection without launching the full player.

4. Summary and outlook

Before introducing *Russian Online* on a wide basis, we chose to run several pilot courses to evaluate the advantages and disadvantages of the platform. Students of these courses were encouraged to use *Russian Online* on a voluntary basis and their feedback was, for the most part, very positive. Reported learning-success was distinctively better than with earlier courses that were completed without *Russian Online*.

This successful first evaluation encourages us to introduce *Russian Online* as a mandatory part of the Russian language courses at the University of Passau, starting with the current semester. Now a tight integration of the software with our traditional classroom-based language courses has become possible and promises interesting perspectives to advance the teaching at our Language Centre. Key point of its success has been the tight cooperation between the content creators and the programmers. Frequent communication ensured the integration of needed features by the programmers and created understanding for compromises due to technical limitations on behalf of the authors.

Russian Online is under continuous development to further its use as a central platform for language teaching. Although initially developed for Russian language courses, the software can easily be adapted to other languages, as the primary didactical concepts and technical requirements remain the same. We imagine that even other kinds of lectures might also profit from use of the platform with its tight integration of texts and exercises. For current developments, we focus on improving the user-friendliness and add some personalization aspects. For example, students will be able to take vocabulary notes within the platform which will then be available to them for their subsequent learning units. We are also adding an export function which makes the learning units available in PDF, so that students may print and archive the text material for offline study / exam preparation. The need to suspend and resume work on longer learning units is being addressed by adding functionality to temporarily save the current state of work.

Acknowledgements: The programming work of Wolfgang Völkl for this project is gratefully acknowledged.

References

[1] Jung, Udo (ed.), Praktische Handreichung für Fremdsprachenlehrer (Peter Lang, Frankfurt a. M., 1998), pp. 281-287, 352-358.
[2] Sun microsystems, Java Web Start, 2006. http://java.sun.com/j2se/1.5.0/docs/guide/javaws/
[3] Apache Software Foundation, Struts Key Technologies Primer, 2006. http://struts.apache.org/primer.html#mvc
[4] Hibernate, Relational Persistence for Java and .NET, 2006. http://hibernate.org/

© FORMATEX 2006