Interaction Concepts for Learning Objects in Codewitz

Wladimir Bodrow, Irina Bodrow
University of Applied Sciences Berlin
Treskowallee 8, 10318 Berlin, GERMANY
+49 30 5019 2478
w.bodrow@fhtw-berlin.de

ABSTRACT
In this paper we present an empirical investigation of selected e-Leaning examples for learning objects (LO) developed within the Codewitz Project. We consider the specific features of different interaction concepts with students when learning programming languages and technology. Several rules and proposals important for the design of leaning objects in Codewitz in particular as well as for analogous e-Learning solutions are derived and described in this paper.

Keywords
Programming education, e-Learning, distance education, learning object.

1. INTRODUCTION
The interaction concept is a significant part in every e-Learning application. It is responsible for the representation, dissemination, and assimilation of knowledge to be submitted to the recipients. The applied interaction concept supports the understanding of the content and makes up the fundament of an efficient e-Learning process. This is where the acceptance of computer supported education begins and ends. During the last couple of years different partner universities and institutions have dealt with this topic. As a result of these dealings a number of concepts and solutions for the development of teaching units have come up. The current state of the results in the project is reported in the proceedings of MMT2006 [1] or at the projects site [2].

2. EXAMPLES OF IMPLEMENTED LEARNING OBJECTS
In this chapter we focus on some of the 180 LOs developed in the Codewitz project and discuss the corresponding interaction concepts. In order to stay in an academic context, they will be analyzed without (!) any link to the developer – strictly concentrating on their scientific value.

Example 1 The first LO concept is shown in Figure 1. The screen is subdivided into four areas: code, execution, memory and conditions. As part of the code area an explanation area is integrated. The navigation is implemented by three buttons situated between the areas in the middle of the screen. An additional button for answering the questions about the implemented navigation concept is also positioned there.

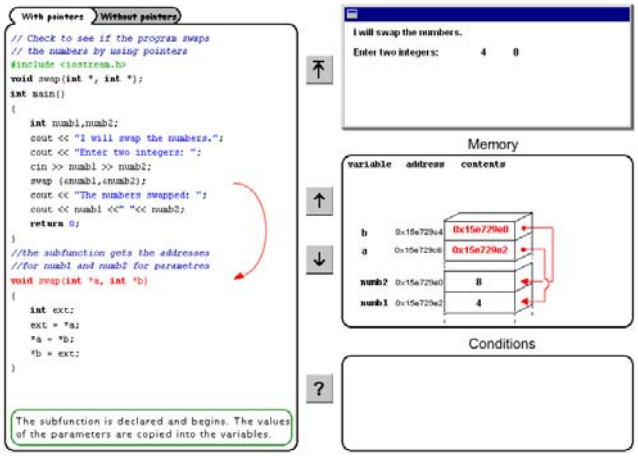

Figure 1. Example 1.

Pros and cons
This solution is implemented according to the debugger concept and step-by-step program execution. The advantages of the interaction concept in example 1 are:

- Clear subdivision of the screen into areas to highlight the interdependence of the different resources used in the exercise.
- Possibility to navigate to the specific line of code i.e. to locate the exact line where the previous session ended.
- Integrated option for input in the execution area.
- Representation of output in the execution area.
- In memory area numerous graphics used to explain the work with storage resources. This provides a very good basis especially for learners of pointer arithmetic in C++.

The following features should be considered as disadvantages of this solution:

- Integration of the explanation into the code area. This placing of explanation will lead to confusion between source code and what it stands for.
- Generally it is not clear: which kind of interaction is in use - instruction or explanation.
- Each time the student wants to navigate to the special line of code – implemented step-by-step – he will be confronted with all the screens that have appeared so far. This leads to a loss of concentration and time.
- Arrows appearing simultaneously in different areas need additional explanation. There is no advice on the screen to understand the priority of all these arrows.
- Pieces of text within the areas have no common frames: some of them are left- other right- or centre- justified. A table frame would support better concentration and efficient learning of the textual content.

- At different levels the LOs request the input of some numbers. Without the student's input the learning procedure will be stopped. This is not self-explanatory and causes problems especially for beginners.

Example 2 The typical screen for this example is presented in the Figure 2.

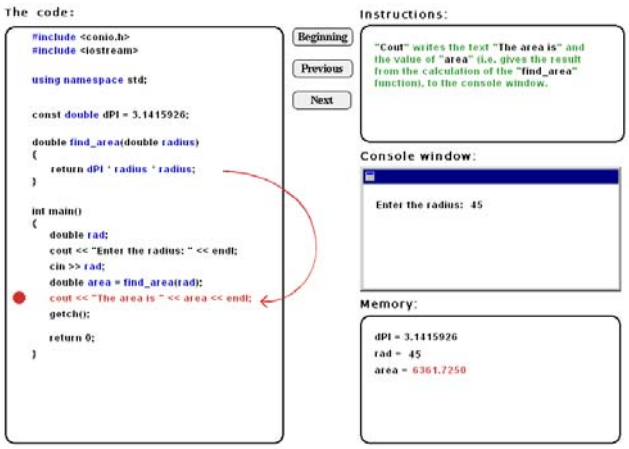

Figure 2. Example 2.

This concept represents a variation of the previous example. The advantages and the disadvantages described before are valid for this example as well. Some additional features could be mentioned:

- Based on praxis experience the instruction/explanation area is separated from the source code on the left. All instructions are posted there.
- The red dot in the code area points to the line currently under explanation.
- The condition area is cut out.
- Outgoing from the beginners' responds the application of the non-interactive exercises (without the input of numbers etc.) will be preferred.
- The textual content is consequently presented in table frame.
- The number of simultaneously appearing arrows is significantly reduced in comparison to example 1.
- The application of graphics for explanatory purposes is also reduced.

Example 3 A typical screen which represents this third kind of LOs is shown in the Figure 3. The user interface is subdivided into four areas: Exercise, Source code, Feedback and Dialog. The buttons: "PREV" and "NEXT" are used to switch the task presented on the screen. The "OK" button is responsible for dialog with the program. Several possibilities to interact with the LO are integrated in the dialog area. In most cases there is a field for input or selection of one of the predefined answers to the task.

Pros and cons

The presented solution does not follow the debugger concept or step-by-step program execution. The advantages of the implemented interaction concept are:

- Clear subdivision of the screen into areas to support the understanding of complexity of programming. This complexity is represented through different structures which the programmer has to control simultaneously.
- The simple avatar (the dog in the Feedback area in Figure 3) supports motivation of the students.
- Integrated possibility for input in the dialog area.
- This example follows the constructivist educational concept (and not mainly applied drills in step-by-step solutions).

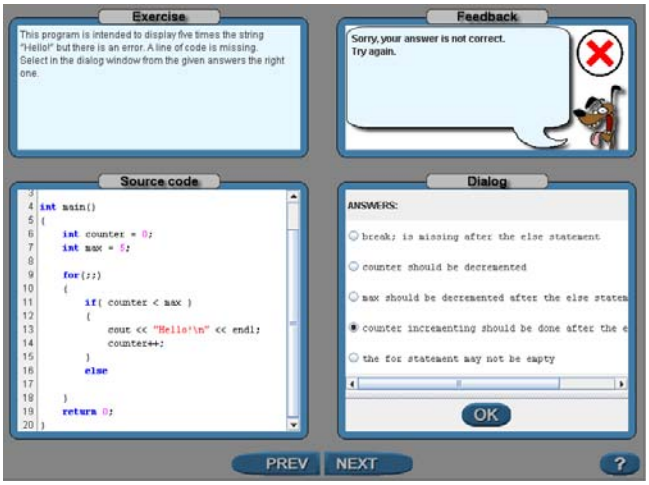

Figure 3. Example 3.

The features below should be considered as disadvantageous in this solution:

- There is no explicit explanation area on the screen. This can result in problems especially for beginners.
- All three areas (except the code area) support interaction with the user but the concept of this interaction and the role of the chosen instruments within the learning process are not clear.
- The same applies to the position of these areas. Where exactly is the important message for the student located? – On the left, on the right or above the code?
- There is no detailed explanation of code (as opposed to step-by-step concepts). Such explanation is very useful not only for beginners.
- Without detailed explanation to the code students will learn some kind of "code blocks" consisting of many lines. Only one single description in the exercise area is given for the whole block (mainly small program).
- The size of the dialog area does not fit the implemented input activity.
- The students have to answer a special question concerning the presented program code. But the explanatory knowledge necessary for answering they have to gather elsewhere.

Example 4 A typical screen design for the fourth kind of example is presented in Figure 4. The user interface is subdivided into three areas: Source code, Feedback and Memory. The buttons are responsible for navigation, similarly as in the Example 1 or 2. The circle labeled by "i" in the source code area provides an explanation to the line of code being under consideration.

Pros and cons

The presented solution follows the debugger concept and step-by-step program execution. The advantages of the implemented interaction concept are:

- Clear subdivision of the screen in areas like in previously presented examples.
- Implemented explanation allows for the user to switch it

off or on if needed. The implementation of explanations is superior to the examples before.
- Graphics on the right part of the screen support a better understanding of memory use.
- The solution can be used in classes (without explanation) as well as at home (with explanations).

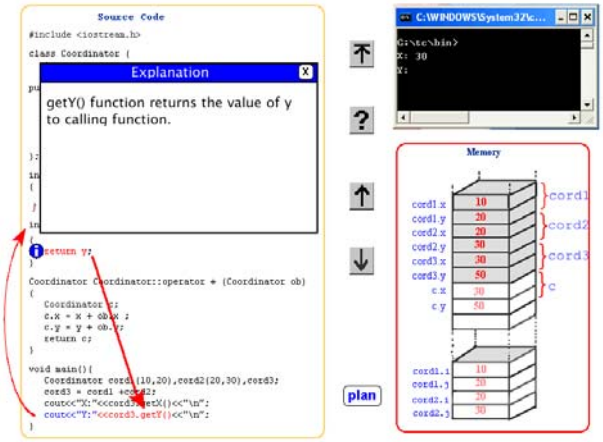

Figure 4. Example 4.

The following features should be considered as disadvantages in this solution:

- Students have to understand the navigation concept first before they start with the example. This provides some problems especially for beginners.
- The use of arrows like in Figure 4 is not stringent and causes some misunderstanding for students.
- The given explanations are very compact and have to be complemented by the teacher in class or by other knowledge sources (i.e. books) at home.

Example 5 The next example for the implemented LO is presented in the following Figure 5.

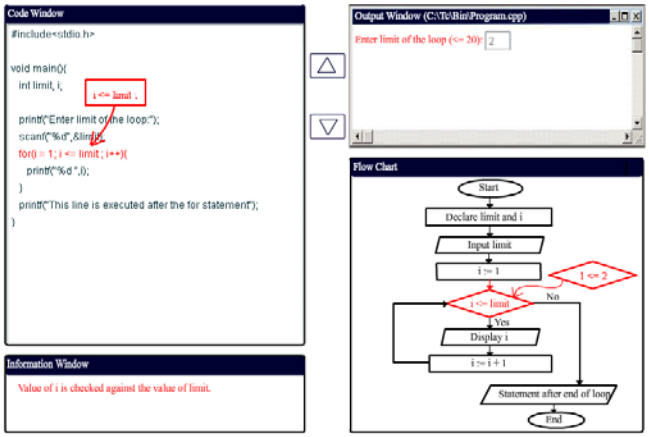

Figure 5. Example 5.

This example is also similar to Examples 1 or 2. Therefore the only differences to them are listed below.

- The exceptional feature of this interaction concept is the explanation of the program progress with the help of a flow chart in a separate area.
- The output area allows the input of data by the students during the session.
- The explanation to each single line given in the information window is very short.
- Additional explanation graphics in the source code as well as in the flow chart area are less helpful because they disturb the whole concept of the screen usage.

Example 6 The interaction concept developed in this example is presented in Figure 6. The user interface is subdivided into two parts: source code and explanation. The slider between both parts is used as a border between them.

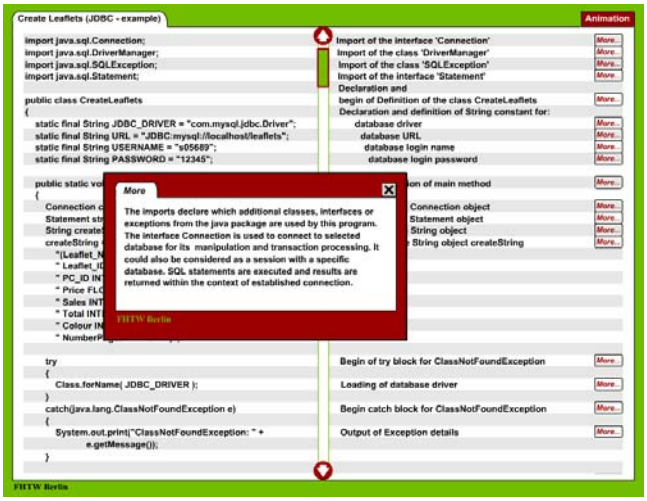

Figure 6. Example 6.

Pros and cons

The presented solution does not follow the step-by-step program execution concept. The advantages of the implemented interaction concept are:

- The concept implemented in this example strictly follows the subdivision of the screen into two parts: source code and explanation. The explanation area supports the knowledge acquisition during the session.
- There are two versions of explanation implemented to each line of code. The short version is presented in the same line (similar to Figure 6). Activating the ***>More<*** button the user gets detailed explanation to the particular part of program as presented in Figure 6.
- The navigation through the source code is realized by two buttons in the middle of the screen.
- Both the optional detailed explanation and the navigation according to the individual knowledge and experience provide the support of different user models.
- Additional media, such as graphics and animation, is implemented in a separate window (Figure 6A).
- The described solution can be used both in classes (with short explanations) and at home (with extended explanations).
- The arrows like those in Figure 6A connect different parts of resources and support the understanding of complexity by the students this way.

The following features should be considered as disadvantages in this solution:

- There are no graphics integrated into the extended explanation.
- In this example the students' influence on the procession

of the program is limited to the navigation buttons.
- Testing the knowledge acquired is also not implemented in this example.

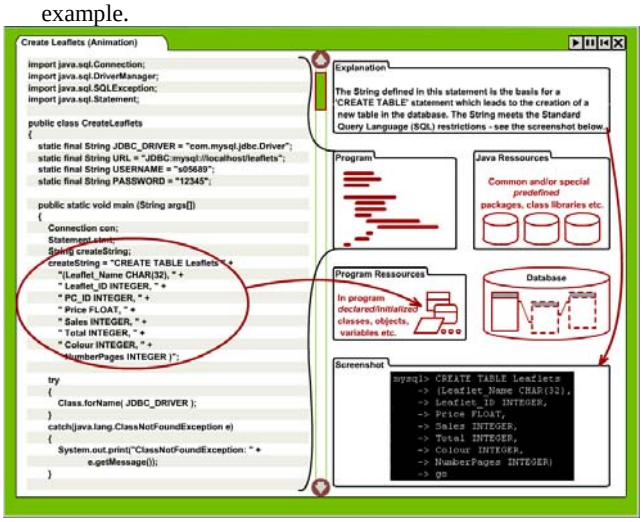

Figure 6A. Example 6.

Example 7 Several LOs were implemented in the Codewitz project based on a special development environment. Those environments were also produced by the project partners. An example for such learning objects is presented in Figure 7.

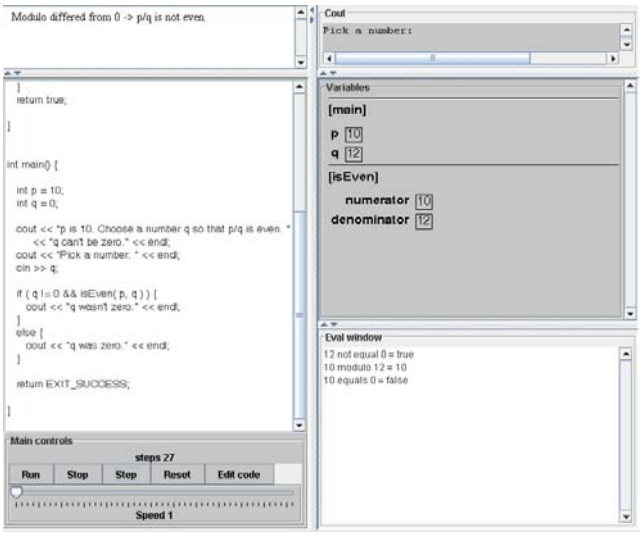

Figure 7. Example 7.

Some of the common features used in such a LO are well designed and realized. A good example for it is the Main controls area (shown in Figure 7). Students can navigate individually through the learning process according to their plans. Otherwise the generalized representation of source code or other components on the screen does not allow their appropriate appearance there. So the currently discussed line of source code can be outside of the visible area. Nevertheless those developments are very interesting and will allow the production of LO with small resources.

Examples 8 and 9 Some solutions like shown in Figures 8 or 9 concentrate on the application of special development tools or presentation techniques.

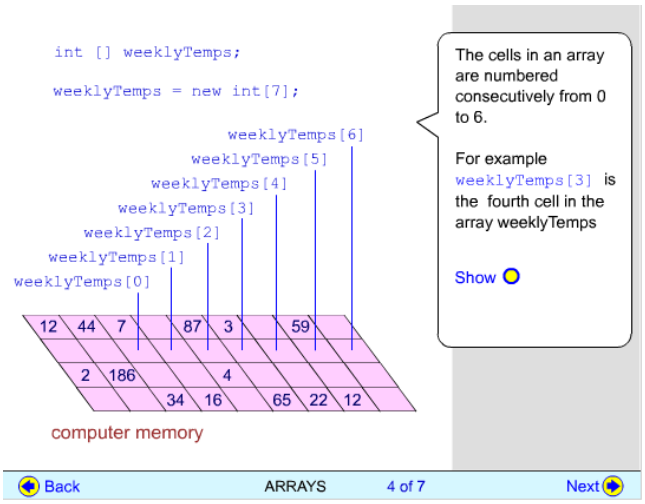

Figure 8. Example 8.

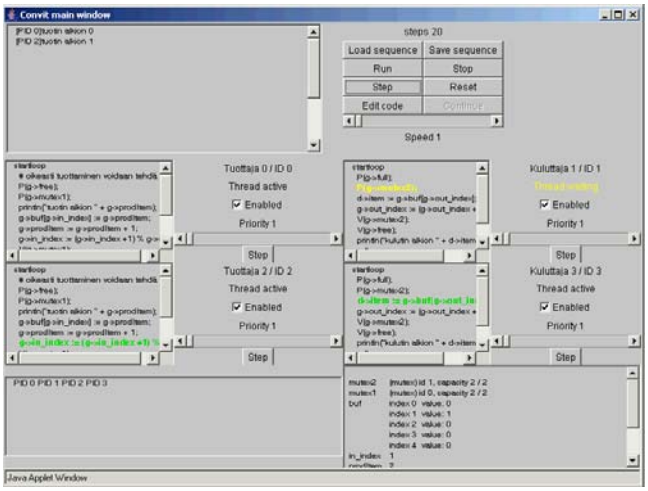

Figure 6A. Example 6.

For an effective usage of such LOs, students at first have to learn and understand the complexity implemented in such interfaces. Obviously those solutions are useful for advanced education in programming. For students in higher semesters the special programming techniques rather than the intuitive concept of interaction with the tool are important.

3. CONCLUSIONS

Based on the analysis made some rules for the (re)design of the Codewitz learning objects can be derived. Those rules are of common interest for analogous e-Learning solutions as well. Different dimensions of consideration build the foundation for classifying them.

Supervised versus non-supervised dimension

The interaction concepts implemented in examples 1, 2, 3, 4, 5, 7, 8 and 9 require additional explanation and therefore could be used in supervised teaching or learning processes. Some of them need more supervision to understand the source code like examples 3, 7, 8 and 9. The application of others (examples 1, 2, 4 and 5) presupposes at least some introductory in class lectures before the learners can start studying on his own. The only example consequently oriented on non-supervised learning is example 6.

Instructions versus explanations

In some examples (3, 7, 9) only instructional concepts of interaction are implemented. In others (examples 1, 2, 5 and 7) a mixed concept with a combination of instruction and explanation is realized. In examples 4 and 6 the explanation concept was used.

Different approaches of teaching/learning

All step-by-step or debugger based implemented interface concepts follow the drilled strategy of teaching/learning among these are the examples 1, 2, 4, 5, 7, 8, and 9. Whereas in the examples 1, 2, 4, 5 and 8 the drilled approach is the only one implemented, in example 7 and 9 additional variants are realized, too. Only example 3 follows the constructivist concept and example 6 offers a flexible navigation and an organization of the learning process.

Interface design

A clear subdivision of the screen into areas is realized in all examples. The source code area is consequently located on the left part of the screen in the examples 2, 4 and 6. Most problems with the screen subdivision appear in example 9. In other examples the problems mainly concern the role of the area and its placement on the screen.

Graphics and media

In the examples 1, 4, 5 and 8 various graphics are used to explain the content directly on the main screen. In example 6 graphics and animation are used in a separate window. The animation is also implemented in the examples 4 and 8. In a number of applications graphics could support the process of content understanding efficiently.

Interaction activities (input)

In most of LOs (examples 1, 2, 3, 5 and 7) the option for input is integrated into one or another area.

Adaptivity

Only in the examples 4 and 6 students can navigate efficiently to the special line of code and start or continue the session there. The step-by-step implementation in other examples provides too much unrequired information which cost time and concentration.

Visualization tools – arrows

Arrows as a special tool for the visualization of knowledge (particularly meta-knowledge) are not used in examples 3, 7 and 9. In most examples there is no stringent concept for using arrows. In order to visualize meta-knowledge the arrows have to connect at least two areas on the screen, to illustrate details of the source code they have to stay within code area. The same problem occurs in the analysis of the learning metaphor and the corresponding template for LOs in general and used colors and text formatting in particular.

4. REFERENCES

[1] MMT2006 Conference Proceedings, Tampere 2006.

[2] http://www.codewitz.com