Diagnosing problem solving strategies of programming novices in secondary education automatically?

Ulrich Kiesmüller, ulrich.kiesmueller@cs.fau.de
(University of Erlangen-Nuremberg, Chair of Informatics 2/Didactics of Informatics, Martensstr. 3, D-91058 Erlangen, Germany)

Torsten Brinda, torsten.brinda@cs.fau.de
(University of Erlangen-Nuremberg, Chair of Informatics 2/Didactics of Informatics, Martensstr. 3, D-91058 Erlangen, Germany)

Abstract
Teaching the fundamentals of algorithms in secondary informatics education is often assisted by special learning and programming environments. Using them to solve algorithmic problems often causes problems in the classroom, although the design of these environments is age-based. In order to give optimal support to pupils, the learner-system interaction of these learning and programming environments should be enhanced and adapted better to the learners’ individual problem solving strategies. The main objective of the research project described here is to find out to what extent the automated diagnosis of a learner’s problem solving strategy is possible, and to what extent this knowledge can be used to enhance the learner-system interaction. To avoid the process observation difficulties documented in the literature, it is necessary to design and to implement special software-based research instruments to diagnose the learners’ individual problem-solving strategies automatically. The requirements for these software components resulted from several test scenarios, using the learning environment Kara, with selected probands observed by human researchers. A first version of the implemented instruments was tested with more than 100 probands (12 to 13 years old) from Bavarian grammar schools. The results of these studies indicate that it is possible to identify and to evaluate different problem-solving patterns with the help of the developed instruments.

Keywords
Secondary informatics education, algorithmic problems, learning environment

MOTIVATION
In secondary informatics education the basics of algorithms are often taught using special learning and programming environments. In this context pupils also learn the basics of programming using didactically reduced, text-based or visual programming languages. Therefore in Germany in some federal countries (e. g. Bavaria), where the basics of algorithms are already taught in the 7th grade (age 12 to 13 years), age-based learning and programming environments, such as Karel, the robot (Pattis, 1994) and Kara, the programmable ladybug (Reichert, 2003), are used to teach the fundamental ideas of algorithms (Schwill, 1997). To improve the learning processes it is necessary to get more knowledge about the learners’ individual proceedings when constructing a solution (under special consideration of the respective solution quality) and to feed back these findings to the process. Therefore special research and diagnostic modules have to be added to the learning and programming environments used in the lessons, whose results can be provided to the learners for individualized support. Unfortunately, the system messages of these learning and programming environments are often very technical and therefore hardly helpful especially for weaker learners. The goal of the research project
described here is to automatically identify and categorize the learners' different proceedings with the aim to improve the learning and programming environments and therewith to enhance the quality of the learning and teaching processes of informatics. With class sizes of up to 34 pupils there remains only less than 1.5 minutes per learner for the teacher to answer individual questions. The goal is to better enable the learners with improved system feedback messages to solve problems that arise by themselves, so that the teacher does not mandatorily have to be consulted at the first arising question (see also Kiesmüller, Brinda, 2007).

PROCESS OBSERVATION METHODS AND RELATED WORKS
Previous systematic studies concerning programming novices' proceedings predominantly have taken place at colleges and universities. Hundhausen (2006) describes how to analyze the programming process of students in dependency of time using a given programming environment using screen-video recording. At first semantical subunits of the given problem have to be determined based upon theoretical considerations. The steps to be identified in the following recording process have to be manually coded afterwards in a very time-consuming process, related to these categories and finally plotted against time.

The problems detected by Hundhausen during his research as well as the difficulties described by Chi (1997) during his analysis of “verbal” data caused the authors of this work to design and to develop specialized research software, which automatically collects data and represents them graphically. A similar proceeding was chosen by Schulte (2004). In his case the probands were pupils (11th graders, 16 to 17 years old), but the topic was object-oriented modelling.

Another goal of the research project described here is to accomplish the categorization of the learners’ proceedings automatically by diagnostic software (see below) after the analysis of the first research results. Therefore, another software tool will have to be developed to identify patterns in the collected data of the tracking software (see below) with the help of pattern-recognition methods.

Beside the analysis of the process an analysis of the artefacts produced by the learners during this process is helpful. To evaluate the quality of the learners’ solution attempts, they were tested with the help of test cases specially designed for this purpose (like in Blumenstein et al. 2004). The selection of the test cases based upon the idea to test an essential solution part with each of them, so that not only a binary differentiation between “completely correct” and “incorrect” solution attempts was possible.

THE LEARNING ENVIRONMENT
Basis for the research in this work is the educational software system Kara (Reichert, 2003) by which learners can control a virtual ladybug placed in a chessboard-like world (see figure 1, right side) with fixed hindrances (e. g. tree trunks) and movable objects (e. g. cloverleaves) using simple finite state machines. Kara can turn left or right, move a single step ahead (subsequently called commands) and it possesses sensors to test, e. g. whether there is a cloverleaf on or a tree trunk in front of its current position. With the sensors it is possible to branch the program flow (see figure 1, left side). Learners have to solve typical tasks, e. g. to navigate Kara through a “forest” of tree trunks or to make it collect leaves. To cause the ladybug to fulfill certain tasks, a learner has to identify the states needed and to specify the transitions with the help of sensors. Also he or she has to compose the correct sequences of commands. The automata terminology/view does not necessarily have to stand in the foreground to solve the tasks that have been
set. Kara also can be used to teach the control structures (sequence – here of commands, selection – here the branches, iteration) as fundamentals of algorithms.

RESEARCH METHODOLOGY

Research Objectives
Because algorithmic problems often cause difficulties for beginning learners, the goal of the project described here is to investigate the problem solving processes more exactly with the aim to enhance the learning processes. Therefore the authors want to find out, which typical proceeding patterns of learners solving algorithmic problems can be distinguished automatically, to what extend such knowledge can be helpful to improve the system feedback of the learning and programming environments and to better adapt them to the learners’ individual proceedings.

Research Instruments
To avoid the process observation difficulties documented in the literature (see above) and to be able to collect and to analyze data relevant for the investigation of the problem-solving processes, within this research project specialized research instruments (a tracking and a diagnostic software) were developed.

- The tracking software (named TrackingKara here) records all solution-relevant steps in dependency of time and assigns these steps to pre-defined categories. Based upon preliminary studies, categories were derived from the problem-solving process in state-oriented modelling: working with states, creation of branches, insertion and handling of commands. Additionally, number and type of sensors, commands, system error messages and the frequency of learners’ program executions are recorded in their chronological distribution. Also, snapshots of the learners’ solution attempts are taken during their problem solving processes to reconstruct and to evaluate the quality of the solution process afterwards (see also Higgins et al. 2002).

- The diagnostic software (named EvalKara here) makes the analysis of the data, collected with TrackingKara, possible in shorter time also for larger groups of test persons. The development of all abovementioned data should be analyzable over time. Additionally, cumulative results are available, which can be analyzed with the help of statistical methods. All data should be visualized graphically for further analyses by human researchers. Furthermore, an
automatic evaluation of the solution quality by means of test-cases is included to test the understanding of basic solution-relevant ideas.

Research Process
These software-based research instruments were designed and developed on the basis of the requirements described above. Afterwards, they were used in case studies to ensure the significance of the collected data.

In a first case study 10 test persons with different pre-knowledge levels in informatics (novices up to students of informatics) were asked to solve some simple algorithmic tasks. Their solution steps were automatically recorded using the tracking software. To ensure the correct interpretation of the collected data the probands were asked to additionally document their proceedings with the “thinking aloud” method. These statements were recorded by a researcher and led to first rules for the interpretation of the data collected by the tracking software. On this basis, further requirements for the diagnostic software were derived and integrated into it afterwards.

After this first revision of the software-based research instruments, the first studies with larger groups of school probands were conducted. About 100 pupils (7th grade, 12 to 13 years old) of two Bavarian grammar schools took part in these studies. In Bavaria in the 6th and the 7th grade informatics education is mandatory for all learners (1 hour per week). In their previous informatics education the pupils received an introduction into basic concepts of informatics, e. g. object-oriented analysis and representation of information using typical application software, e. g. vector graphics tools and word processors (Hubwieser et al., 1997). At the end of the 7th grade the curriculum stipulates the description of sequences with algorithms. Before the learners analyze and construct such representations by themselves, they first learn about the basic control structures such as sequence, choice and loop (in this case using the learning and programming environment Kara) in about 8 lessons. Because of the given constraints it can be assumed that the learners’ pre-knowledge level is similar. During the studies the learners were asked to solve given tasks individually (one pupil per computer), but communication between the test persons was allowed. Within a session of 45 minutes the probands were assigned the following three tasks:

- A: Kara has to invert a pattern of cloverleaves straight in front of it while it moves towards a tree trunk, in front of which it has to stop (see figure 1).
- B: Kara has to find the entry of a “tunnel” formed by tree trunks straight in front of it and to stop there.
- C: Kara has to find the entry of a “tunnel” formed by tree trunks straight in front of it and to stop at the end of the “tunnel”.

FIRST RESULTS
Cumulative Analyses
In the current version EvalKara provides two tools for cumulative analyses of the data collected with TrackingKara: one for the analysis of the time distribution of learner-system interactions and one for the analysis of the frequency of system error messages in dependency of the message type. Table 1 shows the time distribution of the learner-system interactions.
Table 1: Time distribution of learner-system interactions

<table>
<thead>
<tr>
<th>Activity category</th>
<th>Task A</th>
<th>Task B</th>
<th>Task C</th>
</tr>
</thead>
<tbody>
<tr>
<td>State editing</td>
<td>7%</td>
<td>14%</td>
<td>17%</td>
</tr>
<tr>
<td>Transition and command editing</td>
<td>36%</td>
<td>34%</td>
<td>37%</td>
</tr>
<tr>
<td>Kara world editing</td>
<td>7%</td>
<td>6%</td>
<td>0%</td>
</tr>
<tr>
<td>Program execution</td>
<td>34%</td>
<td>38%</td>
<td>35%</td>
</tr>
<tr>
<td>Error correction</td>
<td>16%</td>
<td>8%</td>
<td>11%</td>
</tr>
</tbody>
</table>

The collected data can be interpreted as follows. The time slice for editing the world of Kara is the lowest and decreases to zero in the course of the sessions, because the learners recognize that the learning environment already provides predefined worlds. In the error correction category learning effects can be observed from task A to B, which vary only a little concerning their difficulty. The percentage of the error correction category rises again to task C, which is of higher difficulty (two states required for the solution). The time slice spent on state editing, which is closely connected with structuring the problem solving process, increases with increasing difficulty of the given task from A to C.

The time slice for transition and command editing as well as that for program execution sums up to about a third of the time needed for the complete solution and is obviously independent of the setting of a task (and therefore independent of the difficulty of the task, too).

Table 2 shows the distribution of errors of different categories during the sessions compared for the different tasks. To interpret these data the learners’ statements given in interviews after the sessions were additionally considered.

Table 2: Distribution of error categories per session

<table>
<thead>
<tr>
<th>Error category</th>
<th>Task A</th>
<th>Task B</th>
<th>Task C</th>
<th>Total</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>No transition</td>
<td>0.89</td>
<td>0.32</td>
<td>0.77</td>
<td>1.98</td>
<td>38.4</td>
</tr>
<tr>
<td>Command</td>
<td>1.12</td>
<td>0.07</td>
<td>0</td>
<td>1.19</td>
<td>23.1</td>
</tr>
<tr>
<td>Ambiguous transition</td>
<td>0.62</td>
<td>0.22</td>
<td>0.15</td>
<td>0.99</td>
<td>19.2</td>
</tr>
<tr>
<td>No initial state</td>
<td>0.44</td>
<td>0.39</td>
<td>0.13</td>
<td>0.96</td>
<td>18.6</td>
</tr>
<tr>
<td>Others</td>
<td>0.03</td>
<td>0</td>
<td>0</td>
<td>0.03</td>
<td>0.6</td>
</tr>
</tbody>
</table>

The error frequency falling in the course of the sessions in the categories “command”, “ambiguous transition” and “no initial state” can be explained with learning effects – in spite of the increasing task difficulty. The most frequent error category “no transition” shows a similar course from task A to C like the category “error correction” in table 1 and can be explained analogously. Errors with transitions indicate learners’ problems with structuring the problem solution. The “ambiguous transition”–messages can be avoided by trial-and-error, whereas the “no transition”–messages really indicate deeper structural problems of a learner with the given task.

Analyses of Individuals

With the help of the graphical representations of the data collected with EvalKara the learners’ individual proceedings can be analyzed. So-called activity-time diagrams show at what point in time during the problem solving process the probands perform activities belonging to certain categories (see figures 2 and 3).
In the diagram on left hand side of figure 2 it can be observed for example that the test person at first generates all states, then completes the branches/transitions and finally fills all branches with commands (see section “The Learning Environment”). This represents a well-structured top-down problem-solving pattern. The diagram on the right shows also a well-structured process – but in this case every created branch is filled with commands, before a next branch is edited. The solution is finally built of the several parts at the end of the process. This indicates a bottom-up method.

Some test persons used pure trial-and-error strategies: Every single solution step was tested by program execution.

The graphical representation of the respective process pattern (see figure 3) differs clearly from those presented in figure 2. It can also be assumed that the occurred error messages influence the probands’ directly following steps. For a deeper analysis, the types of errors in such cases must be considered too. Furthermore, strategies of stepwise improvement and also hybrids of the abovementioned process patterns can be identified. A further result is that learners, who start with a completely new solution attempt after a not successful one, often show the same process pattern as before.

**Analyses of Test Case Results**

By the automatic evaluation of test cases EvalKara provides a tool to rate the quality of a proband’s solution attempt. These test cases are derived from the given tasks (A to C, see above). A set of test cases, which concern central ideas of the respective problem solution, was created for each of the tasks. The set of task A contains for example the test case “endcondition”, which checks, whether Kara stops, if it reaches a tree trunk. Another one is “Kara runs” – here it is tested, whether Kara takes available cloverleaves and lays down leaves on empty fields. For each of these test cases EvalKara returns either “success”, if a program run is successfully completed, “endless loop”, if the learner’s program gets stuck in an endless loop, or otherwise the respective system error message, if the program run
ends with an abnormal termination (see figure 4). EvalKara’s feedback after applying the test cases (see figure 4) in combination with the decision, if the program is a solution to the given problem, allow a rough classification of the solution attempts’ quality. An example is: if all test cases are finished with “success” the quality is probably high. A medium quality level is reached in the case of the feedback “success” only for the test cases “Kara runs” and “inverting”, because the basic requirement to stop at the tree trunk was not fulfilled and the special case (Kara direct in front of the tree trunk standing on a cloverleaf) was not considered.

CONCLUSIONS AND OUTLOOK
The results of the studies presented here indicate that it is possible to identify different types of problem solving patterns with the help of the developed research instruments. To identify different types of learners’ problem solving strategies it is necessary to analyze all collected data thoroughly. In view of the evaluation of the solution attempts quality with the help of test cases, a final metric must be found. This enables further analyses of correlation based on methods of descriptive statistics.

The in-depth-analysis of the collected data (about 200 sessions) to confirm the previous categorization of the proceedings and refine it afterwards, is the next step. To support the automatic categorization of problem-solving patterns, pattern recognition methods are going to be utilized.

Afterwards, the basic ideas of the research software developed for Kara should be applied to other common learning environments, such as “Karel, the robot”, to create similar tools there. In that context of didactically reduced imperative programming languages it must be particularly clarified what the relevant (and therefore to be recorded) learner-system interactions are.

To improve the learning environments in the intended manner, it must be finally analyzed, to what extent the collected information about a learner’s proceeding can be used to enhance the learner-system dialogue.

REFERENCES


Biographies

Ulrich Kiesmüller studied Mathematics and Physics at the University of Würzburg (Germany). From 1992 to July 2006 he worked as a grammar school teacher. In addition he passed the state examination in Informatics in 2003. Since August 2006 he works as a scientist in the “Didactics of Informatics” group at the University of Erlangen-Nüremberg. His research interests concern the occurring problems while teaching algorithmics in secondary education using learning environments.

Torsten Brinda studied Computer Science at the University of Dortmund (Germany) from 1992 to 1998. From 1998 to 2005 he worked as a scientist in the “Didactics of Informatics” groups at the Universities of Dortmund (until 2002) and Siegen (2002 to 2005), where he 2004 finished his dissertation. In 2005 he became a professor for “Didactics of Informatics” at the University of Erlangen-Nuremberg. His current research interests are didactic systems and educational standards of Informatics.

Copyright Statement

This work is licenced under the Creative Commons Attribution-NonCommercial-NoDerivs2.5 License. To view a copy of this licence, visit http://creativecommons.org/licenses/by-nc-nd/2.5/ or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA. med author.