Empirical studies as a tool to improve teaching concepts

Carsten Schulte

Pelizeaus Gymnasium
Gierswall 2
33102 Paderborn
Germany
carsten@uni-paderborn.de

Abstract: Alarmed by the outcome of the PISA-Study in informatics education a discussion about empirical measurements has emerged. What instruments should be particularly developed for this subject?

The ideas discussed in this article are based on the debate on interaction between media and methods and in addition about the general aim of such approaches.

We argue for a combination, which relates learning outcomes to the learning and teaching process itself. This type of empirical studies tries to improve learning environments and can be a valuable addition to empirical studies measuring and comparing learning results.

1. INTRODUCTION

In context with PISA, measurements are likely to be seen as a way of comparing educational systems in different places as a whole, by questioning a representative sample survey of all learners in a given group. By ranking the measurements it becomes obvious which ways of learning and teaching are more advantageous than others. But unfortunately, as the discussion about PISA shows, things aren't as easy as that because changing things means adaptation since it is not possible to copy an entire approach. But how to choose those 'things', which are responsible for the effects? Therefore one need not only to know which concepts is the bests but why and how they are working. For example in every 'programming'- 'software development'- or 'algorithmic problem solving'-class a programming environment is used. The teacher chooses languages and tool support in order to meet curriculum needs and in hope to improve learning effectiveness. But how can he be sure to make the best choice (see McIver)?

Questions like this arouse in all areas where computer is used to support learning and therefore are subject in many studies. A possible research design is to compare two groups interacting with two different tools with a pre- and a post-test revealing which group learned more. But results of different studies are often contradictory, which is due to this type of research design, in which the computer is used as a learning tool or media. Clark concluded in 1983 “media do not influence learning under any conditions” (quoted from Kozma). Kozma answered in 1994, “if there is no relationship between media and learning it may be because we have not yet made one” (Kozma, p.7).

The consequence is to study the use of the learning tools more closely. The general research question shifts from searching the best media to searching effective learning environments in which teaching methods, media / tool usage and learning activities are effectively combined. Sometimes this idea is being marketed as blended learning.

The conclusion for empirical measurement is to supplement pre-post designs with instruments to measure the interaction between learners and the learning-tools (or: media). These studies aim to find out and describe successful 'learning patterns': effective user-tool interactions that result in meaningful learning processes.

In the next part of the paper two related instruments will be described: Log file-analyses and the categorization-based examination of screen-videos. Thereafter problems of interpreting results will be discussed and it is argued for a theory-based interpretation.

2. Instruments

The idea of interaction analysis isn’t new. Flanders’s concept uses a fixed schema in which in a fixed interval the observer marks the actual category, which might describe the interval properly (or: that fits the interval). This process can be supported with tools allowing coding videotaped lessons afterward. In these tools, the video can be played repeatedly and stopped at will. Examples of such tools are nud*ist, aquad, catmovie...

2.1 Log file-analysis of a software-development process

Lab-phases or small projects seem to be an important learning method and therefore worthwhile to be studied in order to find out how this method can be used to stimulate learning.

Log files are a means to protocol user actions with the tool. Usually the given command, a time-stamp and the user-input (texts,..) are logged. A Log file gives a summary of the development process. Compared to an analysis of the result (the developed software), a log enables the researcher to gain insight into the failures made and corrected, the difficulties which costs much time to overcome, etc. To do this, the log must be analyzed and interpreted.

I will give an example from a study with two novice-courses in two secondary schools; students were about 16-17 years old. After some month of introduction the task was given to develop software in a group of about four to six persons, which enables two users to play the game memory.

The program was developed using Fujaba. It is an UML-based tool that generates code from class and activity diagrams. The tool logs user actions: Class- or method-diagram, the name, source-code statements, variable declarations, compiler invocations and debugging sessions. The logs were visualized using a simple schema: From left to right the time-flow is given, and from top to bottom the user-actions are shown: On top actions with class-diagrams, then with methods, followed by compiler and debugger-invocations. In each category the names of the classes or methods are listed: the earlier a name is used, the higher it is drawn (see figure 1).

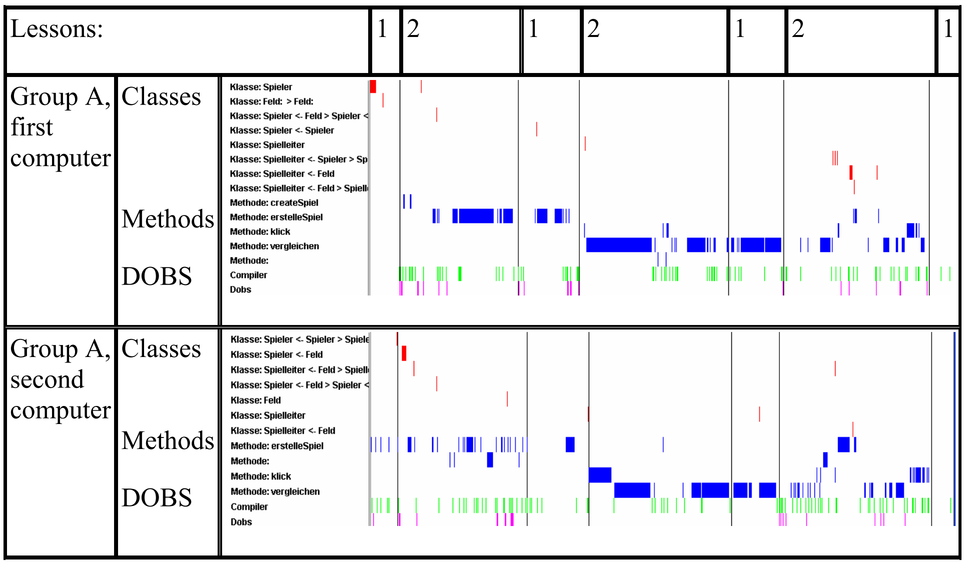

Figure 1:Visualisation of a Log file

This means that a development process starts with implementing a class-model, followed by the implementation of the methods and ends with a debug session would result in a diagram with different blocks from top left to lower right. If there are corrections or additions on the class model being made later, then a mark will appear that is more right than another in the class-section. This way the log-file shows difficulties with the original class-design. It also shows, what methods e.g. took most time to be produced and how many debug-session were necessary. For example the diagram shows, that the class model was quite stable, while two methods consumed most of the development time. An additional analysis of the project shows, that these two methods implement most of the functionality. The group solves implementation difficulties by sticking to the original design, without considering alternative designs, which might have led to shorter method-bodies. A conclusion for the learning process could be to give hints on refactoring strategies, e.g. to split complex methods.

The figure also shows a way to handle the collected log data. It is an aggregation of several thousand items, sorted by time (x-axis) and type (y-axis). Therefore it is easier to make use of the data. Of course, log files are objective and reliable by nature, but what about validity? As log files report user inputs they do not report intentions, they don't even distinguish between purposefully input and simple typing errors. So log files are raw data, which have to be interpreted to make use of the information contained in them. And this results in problems about whether the interpretations are valid.

In general, it seems necessary to distinguish between interpretations as 'normative' and reporting 'possibilities'. Consider the given example. It shows how a team of students was programming a piece of software. And as such, we see a possible way of accomplishing the given task (as we have shown there successful examples). By comparing with other groups the given example could be evaluated as more or less successful. The used time could be the indicator for ranking the successful groups.

Another way of using the results is to compare the empirical development pattern with a normative pattern. This way of using log files has the advantage that in either case some kind of outcome will be generated: maybe the empirical patterns will stick to the normative one, or they won't. Is the measured pattern closely related to the normative one, and then it is being interpreted as a good result.

In the given example a normative pattern may be that expert groups solve easy programming tasks by defining a class structure, implementing methods and a final test run. Therefore a successful pattern would be build out of blocks from the upper left corner to the lower right (see figure). But, what if a group uses some kind of test-driven approach so that in the given figure often entries in the lower line (using debugger) can be found? Due to the given normative frame this approach would be seen as unconventional.

So implicitly made assumptions about good habits in software development should be made explicitly. Such, the norms for interpreting the empirical data are open and can be proven, also.

In a process of empirical studies these norms should be getting more and more accurate – that is one of the main intentions of a theory based approach discussed in the over next section. But before that some conclusive remarks on log files and then the supplement of this instrument with screen videos should be given.

Analysis and interpretation of log files rely on implicit or explicit assumptions about 'good' development process for novices and therefore couldn't be separated from the processes taught to the students, which therefore should be evaluated, too. Second, a process useful for a given task might not be working in another problem domain. Third, in a learning and teaching process – which software development processes are in our context – errors may be a good opportunity to learn.

So it might be useful to support log files with an instrument allowing a closer look on the aims and intentions learners are influenced by while working on their software development task. Such an instrument is to videotape their work including discussions between group members.

2.2 Category-based evaluation of screen-videos

With capturing tools like camtasia screen videos are recorded automatically capturing the screen the learners see and, additionally, their utterances and discussions.

On the one hand, this instrument can supplement log files. For example movements of a mouse cursor aren't logged but they contain information, also: e.g. a programming task might last longer than estimated simply because students weren't accustomed to the IDE and had to search menus for commands to use.

On the other hand it supplements information with the things students say or discuss during work. With this information it is possible to give a more detailed interpretation of the log file patterns. From the utterances it should be possible to say whether there was a plan or an idea the students where following or there wasn't.

In order to lead them to think aloud in a natural way, one can let them work in groups or pairs on one computer.

In the given example, students often discussed their designs, leading to pauses, but only very seldom one member of a group engaged in 'trial-and-error'-programming, although a first impression of the log files might look like trial-and-error. There are several methods to make videotapes searchable for patterns. Most of them rely on transcribing the video. For example all utterances can be transcribed and therefore be searchable.

	Working Style	
	Number	**%**
Trial and error	2031	17,2%
Clear intention	4600	38,9%
Hint from outside person (teacher, other group)	2263	19,1%
Hint from an older project	100	0,80%
Group discussion	2837	24,00%

Figure 2: Working styles

Alternatively the use of fixed codes allows using statistical operations to analyze the data collected by videotaping. The codes represent categories that might be interesting. Here, for example, a code could mean ‘purposeful change of a given method body’ vs. ‘change made by random, just to explore what happens’. Having coded all videos one can just count them.

In this case, all categories were coded by a fixed interval of 10 seconds. This interval-based scheme allows counting out. Alternatively one can count them turn-by-turn. This was done with the log files (see figure one), therefore showing the length of a measured category more accurately. Note, however, that it is more useful to code all data by one of the two possible methods in order to make them statistically comparable. Coding can be done in real time, while transcribing takes two to three times longer.

3. Interpretation of the results

What might be fruitful results of such studies? For example a set of different programming styles: Styles A, B, C. These styles then are mapped to the quality of programs indicating that lets say style A seems to produce better programs then style C. Romero et.al. have chosen this approach to gain a model of program comprehension and debugging expertise, using log files, screen recordings, verbalizations and a category-based coding strategy. To interpret the results: „a cluster analysis can allow us to categorize groups of programmers according to their displayed strategies and to compare this categorization with their performance data. This categorization can also be complemented with the findings of the quantitative analysis. In this way, a model of program comprehension and debugging expertise in terms of behavior and strategy can be empirically derived“ (Romero et.al. 2004, p10).

Such models or sets of different styles are useful for a teacher to describe the level of expertise of each learner. Maybe the teacher even can see specific learning needs. Thereby results of empirical studies are used as tools to describe learner types and to develop specific learning concepts for them.

But this means to use such a ‘model of program comprehension’, or a set of empirically derived programming styles in a normative way: aim of a learning environment then is to train learners to follow style B …

And, a learning environment is seen as successful if it ‘produces’ students that follow a certain style regarded as expertise style.

There seem two objections to this kind of interpretation (or: use of) empirical studies: The direct mapping to a learning environment assumes that one approach fits for all students, but there isn’t such thing as the best teaching method for all (Blömeke). Second, such models or ‘type systems’ describe empirically observed patterns which maybe not the best possible patterns.

Another problem is, how to use such comparisons to develop more successful learning environments? It may be useful to supplement these results with another method of interpretation so that the captured data could be mapped closer to the learning process itself.

4. Conclusion: theory based approach

So far, empirical studies are described here as a means to categorize, mark, evaluate or grade a learning environment. But this means, that the empirical results are only indirectly helpful to improve a given learning environment / teaching concept.

So a method to use empirically studies of the types described above more directly to improve a teaching concept should be useful and will be outlined in this section. The idea is to use a given learning theory as a guideline for developing and evaluating a learning environment. This guideline helps to establish a closer connection between certain results and specific aspects of the teaching concept – and thereby it is possible to develop fine-grained improvements.

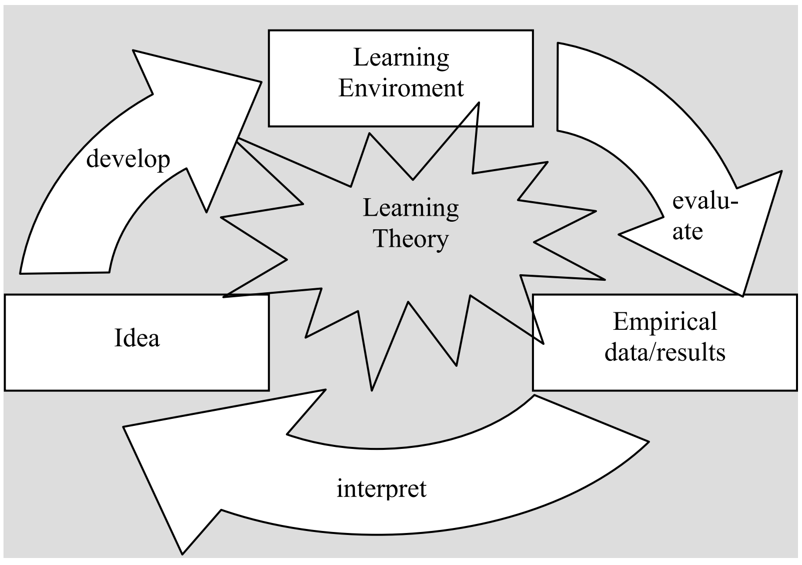

Figure 3: theory based approach

In order to get specific handles to improve an empirically studied learning environment, the studied learning environment is based on a learning theory, which predicts certain results. For example the theory gives hints how to develop a learning environment: for example to situate learning in order to support students ability to use gained knowledge for 'real' problems. So in the learning environment a real world example is used to situate the tasks. Given a cause-effect-structure like this, the empirical data can be used to compare results to the intended ones: If students fail to use knowledge to solve a real problem in a test, the learning environment has failed to situate learning. Maybe the given examples should be changed, …

The general approach is to look for differences between given and estimated results, which can lead to a closer look. Thereby it should be possible to relate certain failures or flaws to certain aspects of the learning environment.

Of course, the results have to be interpreted. In the example above maybe to change the example isn't the right solution. Again, here the theory should help, because it stresses certain aspects of the learning environment as especially important. These crucial aspects should be regarded while designing the instruments for the empirical evaluation. And, of course, even a step earlier: while designing the environment itself. Using this general approach, a detailed comparison between intended and empirically measured results should be possible.

I will try to demonstrate this approach be discussing another example, based on the empirical study already mentioned in earlier sections. The example is the evaluation of a learning environment for introducing students to object oriented modeling based on cognitive apprenticeship (CA). CA suggests introducing novices to a topic like it is done in apprenticeship: One idea is to let students participate in solving realistic tasks with the help of an expert which let students solve easy sub-tasks on their own. The expert demonstrates how to solve and use them in the context of the realistic task. It's crucial to avoid working on isolated sub-tasks without the given context and to give novices an overall understanding of the subject domain. In the context here, novices should participate in the developing of an object-oriented program. So they should understand the difference between classes and objects. Given an overall understanding and a training in using tools, programming language and modeling technique students should be able to develop a software on their own – if, as CA claims, they have gained a general understanding which enables them to use knowledge and skills together in order to solve the task.

Empirical results show that students were able to solve the task, but they weren't able to explain the difference between classes and objects. Mean score in question 'Explain the concepts 'class', 'object' and their difference) was 0.5 from a possible range from 0-2 points.

So it seems, they didn't understand the technology they have used. Maybe they solved the task just by trial-and-error. But (see table working style) trial-and-error was used far less than a purposeful working style. In this case, learning theory helps to explain the empirical data: In the learning environment graphical (UML-based) tools and programming language were used. This lead to a merely visual understanding of object oriented concepts. Classes and objects were visually described. An examination of the videotaped lessons also showed that verbal explanations of teachers and students often used informal language. For example the word 'class' and 'object' were not consistently used. According to Meyer's theory of learning with multimedia, which is based on Paivio's dual coding theory, pictures and word lead to different mental models (Moreno Mayer, figure 1). In this article, Moreno and Mayer derive some principles for the instructional use of multimedia, which can be adapted to this case: students learn better, when verbal information is presented auditorily as speech rather than visually (Modality principle); students learn better, when visual information is presented simultaneously to verbal information (Redundancy principle). As a consequence, the learning environment was (very slightly) changed according to these principles: So in the next course, students were regularly asked to explain visualizations (UML diagrams or visual source code) to the class. The teacher took care that the students made correct use of the terms (especially class and object).

After the course the same questionnaire was given to the students. Mean score (question class and object) changed from 0.5 to 1.7 points, which can be regarded as a significant improvement.

So, empirical studies can and should be directly used to improve the quality of learning environments. To do so, standardized instruments are helpful to compare results.

5. REFERENCES

(Blömeke) Blömeke, Sigrid: Lehren und Lernen mit neuen Medien – Forschungsstand und Forschungsperspektiven. Unterrichtswissenschaft 1, 57, 2003.

(Kozma) Kozma, Robert: Will media influence learning. Reframing the debate. Educational Technology Research and Development 2,42, 1994.

(McIver) McIver, Linda: Evaluating Languages and Environments for Novice Programmers. PPIG 2002. (www.ppig.org)

(Moreno Mayer) Moreni, Roxane; Mayer, Richard E.: A Learner-Centered Approach to Multimedia Explanations: Deriving Instructional Design Principles from Cognitive Theory. IMEJ, 2,2, 2000. (http://imej.wfu.edu)

(Romero et.al. 2004) Pablo Romero, Benedict du Boulay, Richard Cox, Rudi Lutz and Sallyann Bryant: Dynamic rich-data capture and analysis of debugging processes. PPIG 2004.

(Schulte 2004) Schulte, Carsten: Lehr-Lernprozesse im Informatik-Anfangsunterricht. Theoriegeleitete Entwicklung und Evaluation eines Unterrichtskonzepts zur Objektorientierung in der Sekundarstufe II. Dissertation, Paderborn, 2004.