Invited Paper:

Learning by Generation in Computer Science Education

Andreas Kerren
Institute of Computer Graphics and Algorithms, Vienna University of Technology
Favoritenstraße 9-11, A-1040 Vienna, Austria
e-mail: kerren@acm.org

ABSTRACT

The use of generic and generative methods for the development and application of interactive educational software is a relatively unexplored area in industry and education. Advantages of generic and generative techniques are, among other things, the high degree of reusability of systems parts and the reduction of development costs. Furthermore, generative methods can be used for the development or realization of novel learning models. In this paper, we discuss such a learning model that propagates a new way of explorative learning in computer science education with the help of generators. A realization of this model represents the educational software GANIFA on the theory of generating finite automata from regular expressions. In addition to the educational system's description, we present an evaluation of this system.

Keywords: Finite automata, generation, explorative learning, visualization, animation, evaluation

1. INTRODUCTION

During the past years, the development of interactive, multimedial learning software has become more and more relevant in industry and education. In Germany, research funds of several millions EUR were supplied for the research and development of new media in education. However, *generic* and *generative* approaches for the implementation and application of such educational software (ES) are neither in the center of actual research interests nor have they been realized frequently. Important advantages of generic and generative techniques are the high level of software reusability and the reduction of development costs (see [18, 20]). For example, ES mostly supports the same kind of exercises. This is a well-suited starting-point for using *generative* tools and techniques which focus on the automated development of software from formal specifications. By means of them, exercises can be generated from specifications. It is no law that such a specification has to declare only from teachers. Also the learner can use this possibility, see the next Section 2 for details. We will concentrate on such generative methods in the rest of the paper[1].

Different from this, *generic* ES is developed for a whole content area. Following, instances are built for special learning units. A common part of ES could be, e. g., a control for learning targets whereas other elements, like interfaces to knowledge databases, have a more subject-related character. The system InterTalk of Diehl and Ohlmann realizes such a generic concept. It is described at the web site [11] in more detail.

In addition to reduction of development costs and software reusability, generative methods can be used for realizing new learning models. Such a learning model, called "levels of exploration", is discussed by Diehl and Kerren in the conference paper [8] and clarified with topics in compiler construction and formal languages. Visualizations and animations of computational models, e. g., finite automata or abstract machines (see [24]) are considered which can be integrated in ES for compiler construction. The discussed learning model could also be used for other process-oriented application areas, like electrical engineering or physics. Important is the descriptiveness of these processes by specifications. It divides *explorative learning* [6] into four different levels. The degree of learner exploration is relatively limited in the first both levels: they describe a static approach (level 1) on the one hand respectively an interactive approach (level 2) on the other.

In the static approach, the execution of an instance of a computational model is animated for a given, fixed input. This approach has a strong behavioristic characteristic (see [23]). The knowledge transfer of the working of a special instance is in the foreground and the passive learner has only a low influence on the kind of the presented information. He/she has merely the control on when the animation is started or how fast the animation runs. An example for an implementation of this approach is the ADLA-system of Braune et al. [5]: an ES on the theory of lexical analysis of programming languages with the help of finite automata.

In the interactive approach, an user-defined input is possible. This approach supports the learner to examine computational models with own examples. But, the visualized instance of the computational model is—as in the static approach—fixed and cannot be changed. The ADSA-system [16, 17] for the animation of the semantical analysis of programming languages is an example for this interactive approach.

So far, generators for producing animations of instances of computational models from specifications have been ignored. In the next section, we discuss how generative methods can facilitate explorative learning with the help of both higher levels of exploration. Then, a generative ES system is presented. Its learning efficiency is analyzed on the basis of an evaluation with more than 100 participants.

[1] This paper is a slightly modified and extended version of [19] that was presented at the ACM ITiCSE '04 Conference in Leeds, UK.

2. LEARNING BY GENERATION

Figure 1 shows a coarse overview of a potentially HTML-based ES which uses generative techniques. The HTML-document covers the learning content of a process-oriented topic, i. e., it contains definitions, descriptions, images, simple examples, etc. Additionally, the learner can formulate a specification of a process described in the HTML-document. This specification is transferred to a generator embedded into the ES. The generator produces a didactically wise representation of the specified process, e. g., a visualization (including animations) or an auralization. The ES offers an—possibly generated—interface to this representation that is used by the learner for input, interaction or control purposes.

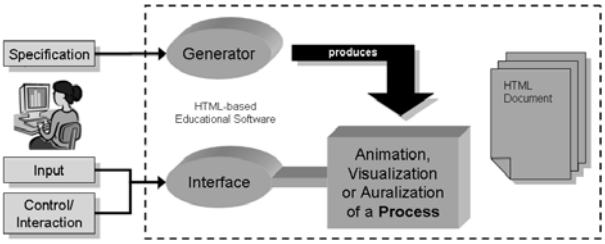


Figure 1: Use of generative methods for developing ES

In essence, the last paragraph discussed the generative approach of the third level [8]. Beside the pure knowledge transfer, the understanding and the interpretation of the learning content should be encouraged. Information is not structured by a teacher but learners have to explore and to order their own information. This approach (called "first-order generative approach") enables learners to formulate new hypotheses and validate them by changing specifications or user inputs. The GANIMAM-system [10] is an example for an implementation of the first-order generative approach. It is about a web-based generator for interactive animations of abstract machines. The learner can enter a specification of an abstract machine as instance of a given machine model. A generator produces a visualization of the specified abstract machine in the shape of an interactive JAVA applet. This generated machine can interpret machine instructions written by hand or automatically translated from a high level programming language. The learner can find out much about the computational model, but nothing about its generation process.

This disadvantage is avoided by the generative approach of the fourth level (called "second-order generative approach"). Here, we can check hypotheses about the generation process itself because the generator is represented within the ES, e. g., in form of an animation, see Figure 2. Such a system supports the learner to better understand the computational model (or any process in general) but also the working of generators which produce instances of such models. This is one of the most important insights on the teaching of compiler construction. According to this, learners should really understand mathematical basics on the one hand and coherences at generating computational models on the other hand. Section 3 describes an ES system that realizes this second-order generative approach. In the context of learning, we combine these two generative approaches to the term "learning by generation".

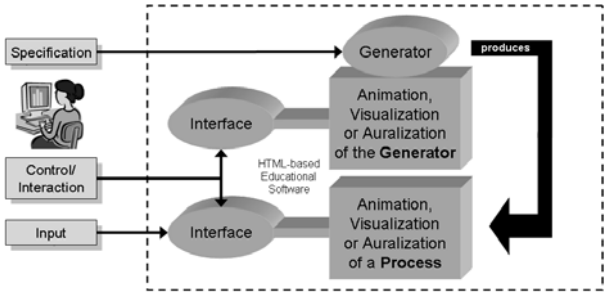


Figure 2: Use of the second-order generative approach

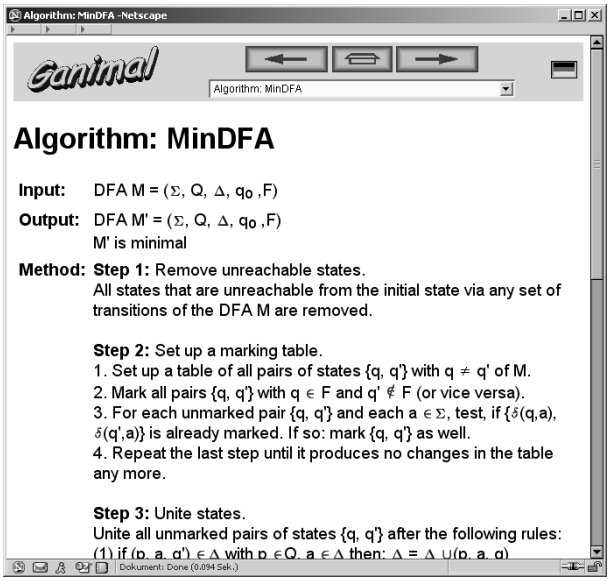
Our second-order generative approach leads to an analogy with the so-called "microworlds" metaphor. This term means a tiny world inside which a learner can explore alternatives, test hypotheses, manipulate constituent parts, and discover facts that are true about that world, see for example [23]. Thus, microworlds are similar to the well-known simulations which can be found in ES on physics or electrical engineering. A microworld differs from a simulation in that it can be considered as a second-order simulation that supports the own construction or modification of models instead of a pure monitoring of a model. Blumstengel [2] states that microworlds (and simulations too) often promote active and explorative learning and act motivating if the ES has a good didactical design.

Furthermore, learning by generation is closely related with the constructivist view of learning. The term "constructivism" refers to the idea that learners construct knowledge for themselves as he or she learns. Each learner individually and socially constructs meaning. Therefore, knowledge cannot be transferred in a traditional way, e. g., by instruction. The learner has to integrate the knowledge into his/her individual mental structure and existing knowledge constructs. Consequently, moderate constructivism attaches great importance to the creation and design of stimulating learning environments that give the learner the possibility to generate individual knowledge constructs (cp. [15, 23, 2]).

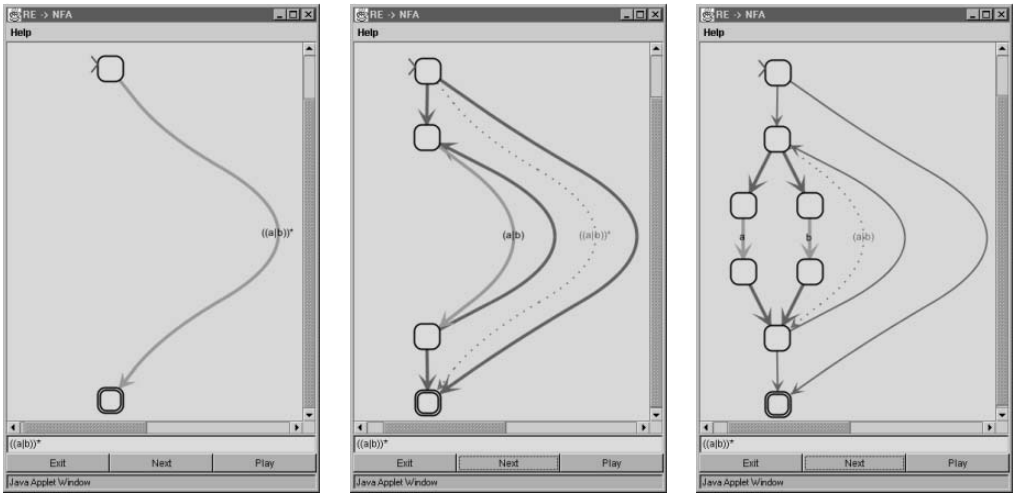
3. TEST CASE: GANIFA

In context of the GANIMAL project[2] [13], we have designed a development environment for the realization of the fourth level of exploration. The resulting ES system, called GANIFA, is an electronic, HTML-based textbook on the theory of finite automata and their generation from regular expressions [9]. GANIFA can be locally used as well as via the Internet [12, 4].

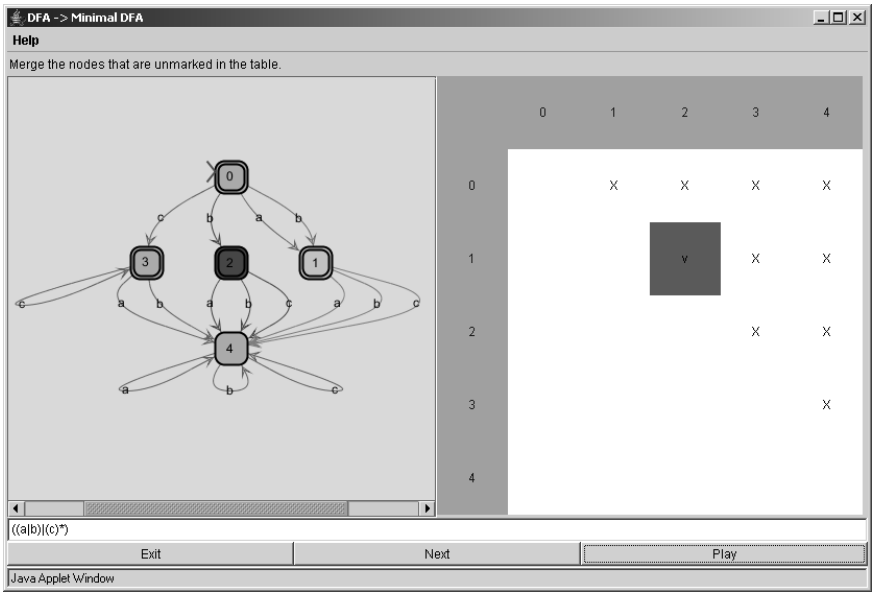
[2]This research has been partially supported by the German Research Council (DFG) under grant number WI 576/8-1 and WI 576/8-3.



(a) Textbook: explanations to a minimization algorithm



(b) Animation of the generation of a NFA from a RE $(a|b)^*$



(c) Animation of the minimization algorithm based on a DFA that was generated from the RE $a|b|c^*$

Figure 3: Screenshots of the GANIFA system

System Description

The textbook introduces into the theory of finite automata. Particularly, it gives an overview of general formal languages, regular languages and expressions. Afterwards, transition diagrams, non-deterministic and deterministic finite automata are described in more detail. The system shows formal definitions in a separate browser window if the learner chooses the corresponding hyperlinks in the textbook.

Advanced techniques for software generation as used in compiler construction have been applied to the automatic generation of animations contained in the learning system, i. e., in the form of a customizable JAVA applet. It is possible to specify any finite automata with the help of regular expressions and to animate the respective generation process. Figure 3 shows three different screenshots: on Subfigure 3(a), one page of the electronic textbook is displayed. It contains explanations to the theory on minimizing finite automata as well as to an appropriate algorithm. The minimization is the last phase of the whole generation process. Via hyperlink, the learner can change to another page containing a form to enter a regular expression. First, the generator produces a non-deterministic finite automaton (NFA) from a regular expression, e.g. $(a|b)^*$. This generation process is represented by an animation. Subfigure 3(b) illustrates three steps of the generation of a NFA. After that, the user could enter an input word to watch the acceptance behavior of the generated automaton or he/she could completely skip over all intermediate phases of the generation (see below) to watch the minimization phase only. An example animation of the minimization algorithm on the basis of the regular expression $a|b|c^*$ is shown by Subfigure 3(c). Each generation phase can be animated with the help of the GANIFA system:

1. Generation of a non-deterministic finite automaton (NFA) from a regular expression (RE) [24].
2. Removal of ε-transitions of a NFA [22, 24].
3. Transformation of a deterministic finite automaton (DFA) from a NFA without ε-transitions [22, 24].
4. Minimization of a deterministic finite automaton (minDFA) [14].

(5.) The GANIFA-applet can visualize the computation of each generated automaton on an user-defined input word.

The applet draws all generated automata as transition diagrams using a special graph layout algorithm in order to preserve the mental map [21], see the paper of Diehl et al. [7] for details. In addition, it is possible to visualize the acceptance behavior of generated automata for an input word entered by the learner (see Item 5 of the enumeration). The GANIFA-applet is customizable through a large set of parameters, and it is easy to integrate the applet into existing web pages. So, it is possible to visualize only a range of algorithms and to pass a finite automaton or a regular expression respectively an input word to the applet directly. Based on these functionalities, GANIFA supports the second-order generative approach.

Evaluation

An evaluation of new ES should give information about its usability and effectiveness. Unfortunately, most of new ES systems are not evaluated adequately. One reason could be that there is a risk to get bad results. Consequences are a bad usability, inconsistent design or impractical preparation of learning contents. The acceptance and effectiveness of GANIFA were proved by a *summative* evaluation (see e. g. [2]) based on learning experiments with more than 100 test persons. Aim of this evaluation is to prove that persons learning with the generative ES system GANIFA have an equal or a better learning performance than persons learning with other teaching methods. We used inferential statistics (t-test resp. analysis of variance (ANOVA)) to execute the analysis.

Test Design: At first, we performed an *usability test* together with some students to detect mistakes of the electronic textbook. Typos, dead hyperlinks, runtime errors of the JAVA applet, etc. lead to a painful system appliance and increase the risk of a negative evaluation result from the first.

As test persons (TP) we chose 118 students of a computer science foundation course at the Saarland University in Germany. The evaluation was not announced, i. e., the students could not agree to participate. In this way, we could eliminate registrations of test persons who have a positive/negative preference for interactive, multimedial ES. They were informed at the beginning of the lecture and evenly divided into four groups. Each group was escorted to a separate room. As schematically shown in Figure 4, these four learner groups studied the theory of generating finite automata each with the help of different methods: classical instruction, text book, and two ES systems with (GANIFA) and without (ADLA) generative part. All groups had a time frame of 35 minutes. We had to take care of identical presentation of learning content between the groups. Following the learning phase, an exercise with nine knowledge and ten transfer questions was performed. Knowledge questions ask for the reproduction of learning content described by the learning method. Additionally, transfer questions ask for a deeper understanding of the content. Finally, all test persons should complete questionnaires with regard to personal statements and teaching aids (learning methods). They are based on 11-ary rating scales with numeric markers and on so-called “open questions”: test persons could formulate their answers in written form. Note that all participants were informed about their results on a special web page. For this purpose, they could leave their matriculation number on the questionnaire.

Results: In a first step, we performed a descriptive analysis of our data. Through summation of all achieved

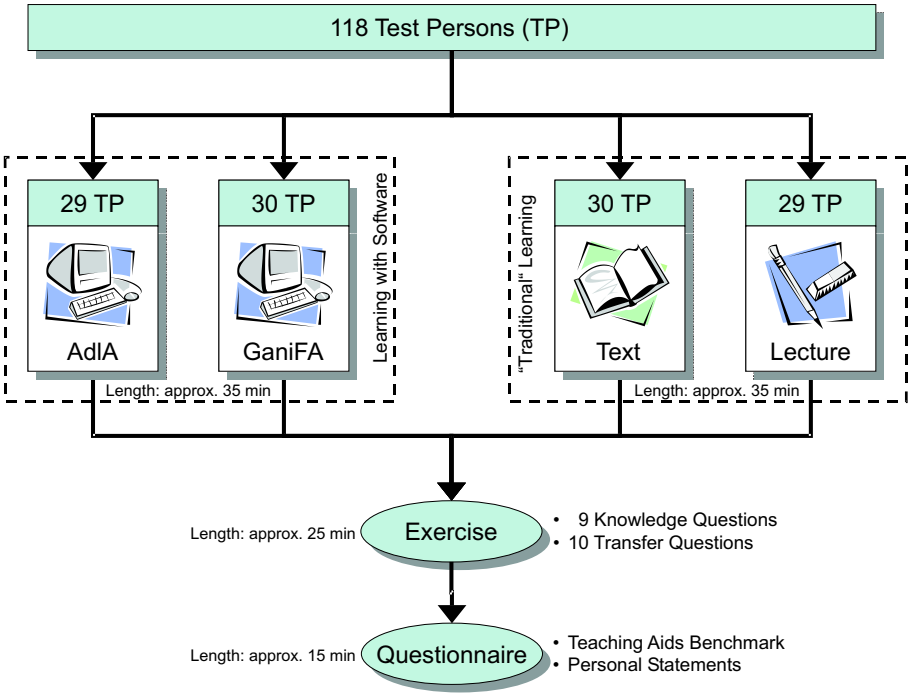


Figure 4: Test design of our evaluation

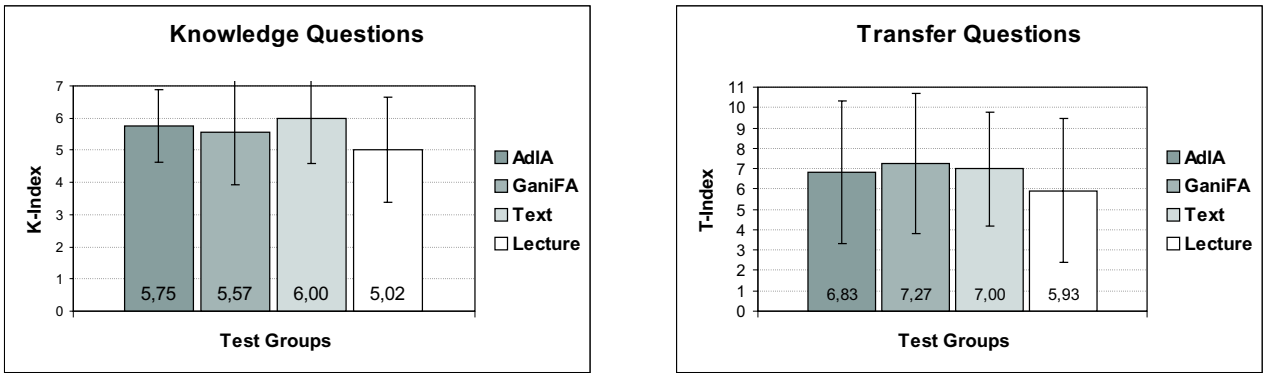


Figure 5: Histograms of means of summative performance indices for the knowledge (maximal score: 7) and transfer questions (maximal score: 13)

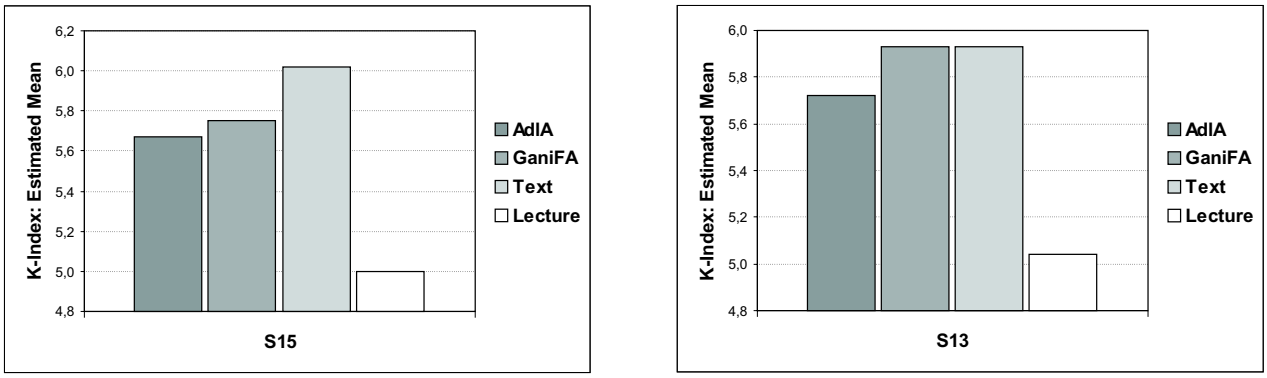


Figure 6: Estimated performance differences between teaching methods for the knowledge questions with involvement of the control variables S15 and S13

scores for the exercise answers, we obtain the statistical values of the sample, shown at Figure 5. The standard deviations are represented as error bars. In case of the knowledge questions (K-Index), the performance mean PM of the text group is the highest one ($PM = 6,00$). Otherwise, the highest performance mean for the transfer questions (T-Index) goes to the GANIFA group ($PM = 7,27$). This result would confirm our assumption that a conventional text book is a better choice to learn basic knowledge than one of the both ES systems if a suitable statistical test model would have a significant result. In contrast, the both ES systems would have advantages in the consolidation and deeper understanding of the learned knowledge.

An one-factorial analysis of variance that scans the random differences of the learning efficiency between our four groups adduces no significant results neither at the knowledge questions nor at the transfer questions. Also, we could not discover any difference computing the effect value [3]. In consequence, we tried to improve the precision of our analysis with the help of a covariance analysis. With involvement of two control variables S15 "computer science course at school as intensive/basic course, voluntary study group, or no course" and S13 "mathematics in school as intensive/basic course", we obtained with S15 a significant performance difference at the knowledge questions in favor of the text group (error probability: 4,1 %). The GANIFA group and the text group had nearly the same performance with control of S13. For both control variables, the GANIFA group achieved a performance improvement in comparison with the ADLA and the lecture group (error probability: 4,7 %), cp. Figure 6. So, we have a solid base for further optimizations and development of the GANIFA system and of the discussed second-order generative approach[3].

A critical view on the test design lets infer that a part of the knowledge questions was chosen too simple because of the very good average performance results in this category. Is is possible that some effects could fail to appear for this reason. Future evaluations should include a pretest to check the previous knowledge of the participants. Furthermore, it is probably that the short time frame of the evaluation was not sufficient to discover all advantages and interactive possibilities of the GANIFA system. In general, this could be a drawback of our (widely-used) test design, cp. the statements of Baumgartner [1].

The answers of the open questions show that the electronic textbook GANIFA is well-convenient as meaningful supplement of traditional learning/teaching methods because of the interactive animations, clarity, and better motivation. The students liked the generated animations of GANIFA and esteemed to work indepedendly and to use all possibilities of our generated animations. This good opinion holds also for graphics, didactical design and formal definitions in a separate definition window. Some students disliked the form of the definitions: "too formal". There was a further problem if students tried to enter their own regular expressions: regular expressions had to be completely set in brackets. This did not meet the established standard in this area. We could not correct the non-intuitive input of them early enough.

[3] A more detailed description of the realization and analysis of this evaluation is contained in the author's PhD thesis [18].

Web Access Statistics

We have analyzed the web access to the electronic textbook GANIFA on the basis of the web server's log files. This was done within a time period of 27 days. In this period of time, we could identify web accesses to at least one web page of the textbook from 479 IP addresses (no robots). Note that all accesses from the same IP address were counted several times if they did not occur within one hour. 44.9 % of them were accesses to at least three web pages of the textbook, i.e., we had an average value of approximately 8 user accesses per day. More than 50 % of these users accessed to at least one web page that contained the embedded JAVA applet. But we could not ascertain how many users had really run the applet.

4. CONCLUSION

In this paper, an approach for explorative learning with generative methods was presented. Learning by generation has a constructivist orientation and supports self-organized and self-controlled learning. Our educational software GANIFA is a realization of this approach and was developed in context of the GANIMAL project. We discussed the system in more detail as well as an evaluation of the system with more than 100 participants. The results of the following statistical analysis prove that the learning efficiency of GANIFA is higher than the one of classical instruction and nearly as good as the textbook's. GANIFA was observed as learning aid very positively. The most participants would like to use GANIFA as lecture supplement. Admittedly, the test design itself can be critically regarded because of the necessary improvements with respect to the influence of user interaction at the educational software.

5. ACKNOWLEDGEMENTS

I would like to thank Stephan Diehl, Carsten Görg, Torsten Weller, Julia Kneer, Christoph Glasmacher, and all other people involved in the GANIMAL project.

6. REFERENCES

[1] P. Baumgartner. Evaluation of Media-based Learning (in German). In M. Kindt, editor, *Projektevaluation in der Lehre – Multimedia an Hochschulen zeigt Profil(e)*, pages 61–97. Waxmann, Münster, 1999.

[2] A. Blumstengel. *Development of Hypermedial Learning Systems (in German)*. Wissenschaftlicher Verlag Berlin, WVB, 1998.

[3] J. Bortz. *Statistics for Social Scientists (in German)*. Springer, Berlin, Heidelberg, 5. edition, 1999.

[4] B. Braune, S. Diehl, A. Kerren, T. Weller, and R. Wilhelm. Generating Finite Automata – An Interactive Online Textbook, 2002. URL: `http://www.cs.uni-sb.de/GANIMAL/GANIFA`.

[5] B. Braune, S. Diehl, A. Kerren, and R. Wilhelm, Animation of the Generation and Computation of Finite Automata for Learning Software. In *Automata Implementation, Proceedings of the 4th International Workshop on Implementing Automata (WIA '99)*, volume 2214 of *Lecture Notes on Computer Science, LNCS*, pages 39–47, Potsdam, 1999. Springer.

[6] J. S. Bruner. The Act of Discovery. *Harvard Educational Review*, 31:21–32, 1961.

[7] S. Diehl, C. Görg, and A. Kerren. Preserving the Mental Map using Foresighted Layout. In *Proceedings of Joint Eurographics – IEEE TCVG Symposium on Visualization (VisSym '01)*, Eurographics, pages 175–184, Ascona, Swizerland, 2001. Springer.

[8] S. Diehl and A. Kerren. Levels of Exploration. In *Proceedings of the 32nd ACM Technical Symposium on Computer Science Education (SIGCSE '01)*, pages 60–64, Charlotte, NC, USA, 2001. ACM.

[9] S. Diehl, A. Kerren, and T. Weller. Visual Exploration of Generation Algorithms for Finite Automata. In *Implementation and Application of Automata*, volume 2088 of *Lecture Notes on Computer Science, LNCS*, pages 327–328. Springer, 2001.

[10] S. Diehl and T. Kunze. Visualizing Principles of Abstract Machines by Generating Interactive Animations. *Future Generation Computer Systems*, 16(7), 2000. Elsevier.

[11] S. Diehl and M. Ohlmann. InterTalk, 2002. URL: `http://www.cs.uni-sb.de/~diehl/InterTalk/`.

[12] GaniFA. Download Page, 2002. URL: `http://www.cs.uni-sb.de/GANIMAL/download.html`.

[13] Ganimal. Project Homepage, 2002. URL: `http://www.cs.uni-sb.de/GANIMAL`.

[14] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[15] D. H. Jonassen, T. Mayes, and R. McAleese. A Manifesto for a Constructivist Approach to Uses of Technology in Higher Education. In T. M. Duffy, J. Lowyck, and D. H. Jonassen, editors, *Designing Environments for Constructive Learning*, volume 105 of *NATO ASI, Series F, Computer and System Sciences*, pages 231–247. Springer, Berlin, Heidelberg, New York, London, 1993.

[16] A. Kerren. Animation of the Semantical Analysis (in German). In *Proceedings of the 8th GI Conference on Informatics and School (INFOS '99)*, Informatik aktuell, pages 108–120. Springer, 1999.

[17] A. Kerren. Visualization and Animation of the Semantical Analysis of Programs (in German). *Informatica Didactica – Journal on Didactical Basics in Informatics*, 1(1), 2000.

[18] A. Kerren. *Generation of Interactive Animations for Compiler Design (in German)*. PhD thesis, Saarland University, Saarbrücken, Germany, 2002. Shaker Verlag, Aachen, ISBN 3-8322-0899-2.

[19] A. Kerren. Generation as Method for Explorative Learning in Computer Science Education. In *Proceedings of the 9th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE '04)*, pages 77–81, Leeds, UK, 2004. ACM Press.

[20] A. Kerren, R. Wilhelm, and S. Diehl. MALL – Final Report (in German), June 2000. URL: `http://www.cs.uni-sb.de/RW/projects/mall/`.

[21] K. Misue, P. Eades, W. Lai, and K. Sugiyama. Layout Adjustment and the Mental Map. *Journal of Visual Languages and Computing*, 6(2):183–210, 1995.

[22] M. Rabin and D. Scott. Finite Automata and their Decision Problems. *IBM J. Res. Dev*, 3/2:115–125, 1959.

[23] R. Schulmeister. *Basics of Hypermedial Learning Systems (in German)*. Addison-Wesley, Bonn, 1996. English version accessible under URL: `http://www.izhd.uni-hamburg.de/paginae/Book/Frames/Start_FRAME.html`.

[24] R. Wilhelm and D. Maurer. *Compiler Design: Theory, Construction, Generation*. Addison-Wesley, 2. edition, 1996.