Integrating a Simulation-Visualisation Environment in a Basic Distributed Systems Course: A Case Study Using LYDIAN

Boris Koldehofe Marina Papatriantafilou Philippas Tsigas
Department of Computing Science
Chalmers University of Technology
SE-41296 Göteborg
{khofer,ptrianta,tsigas}@cs.chalmers.se

ABSTRACT

Distributed algorithms can be difficult to understand as well as to teach. A way to provide students with an experience of the execution of a distributed algorithm is the use of a simulation-visualisation environment. In this work we present a case study of integrating a simulation-visualisation environment into a distributed system course. We evaluate a distributed system assignment in which students used LYDIAN, an extensible library for distributed algorithms and animations, to implement their algorithms. In our study neither the teachers nor the students had earlier class experience with LYDIAN. The feedback received gives valuable information on what simulation-visualisation environments for distributed algorithms need to provide in order to be successfully used in class. We are not aware of any similar study in the area of distributed computing. However, the feedback we have received shows the significance of such evaluations to help users improve their performance and help them to acknowledge the wealth of tools they are provided.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*Animations*; K.3.1 [**Computers and Education**]: Computer Uses in Education

General Terms

Algorithms, human factors, measurement

Keywords

Distributed algorithms, simulation, animation, educational tools, course evaluation

1. INTRODUCTION

Recently, many simulation-visualisation environments have been introduced to support the learning of distributed algorithms [1, 2, 3, 5, 8, 9, 11, 12, 15]. These environments try to help users to understand better the complex interactions between processes within the execution of a distributed algorithm. Often users can investigate interesting behaviour and properties of algorithms by modifying respective parameters of the distributed system. Although many of these environments have been created under educational aspects and provide a wealth of tools to help students, little is known about how teachers and students actually use them in class.

In this work we take a step in describing such an experience from the perspective of LYDIAN [5, 11], an extensible library for distributed algorithms and animations. The study is based on results taken from a compulsory basic undergraduate course in computer science and engineering -distributed systems-, at Chalmers University of Technology. The teachers taking part in this study have not used LYDIAN in class before, but were positive in using it from what they heard and read about it. The feedback received shows that students succeeded well in the implementation of an algorithm. Many students experienced some behaviour of the algorithm they did not expect before, and this helped them in better understanding the algorithm. The feedback also shows that students should be asked to test their implementations by exploiting various parameters inherent in network topologies to improve their knowledge. Naturally, good simulation-visualisation environments should provide such possibilities. One should also remark that animation, as expected, is of good help. For students being able to successfully exploit features of these environments, places also high demands on the documentation of the respective tool.

Before describing the study in more detail, we first give an overview on related tools and we introduce LYDIAN.

2. OVERVIEW OF RELATED TOOLS

Compared to advanced system simulation tools as [3], LYDIAN focuses on the educational aspects of algorithm visualisation which are mainly to support the student in reasoning on the analysis of the distributed algorithm. At the time LYDIAN [11] and its animation framework [5, 4] for distributed algorithms were introduced, there was one known attempt towards a a set of animations of distributed protocols for educational purposes, ZADA [8], based on the animation package Zeus, a Modula-3 based system for specialised platforms. The effort resulted in a small archive of protocols, for each of which the set of views is fixed and the implementation is the same program as the animation (this implies essentially fixed timing, workload, etc). Of relevance was also the interesting work by Ben-Ari in [1] and [2]. There, the focus is on providing a framework for writing distributed algorithms (in a portable language) that allows students to interact with the states of a process and this way understand state changes and data structures of the algorithm. Subsequently, more tools with emphasis on different educational aspects evolved.

VADE [9] is a system that supports algorithms to be executed as JAVA processes on a server, and providing the client with a consistent view on algorithm events that happens on the server. The visualisation is based on WEB approach where users can view on a web-page the visualisation of a selected algorithm by downloading the respective Java client. The animations supports multiple views, but makes no distinction between views for special educational purposes. The approach is mainly designed to make students view a prepared protocol, but not to implement protocols on their own. It seems that it was even thought to prevent an observer from viewing the code behind an algorithm. To the best of our knowledge, there is no recent development of this tool.

In the contrary ViSiDiA [15] supports, like LYDIAN, an integrated approach of simulation and animation of algorithms and in this respect covers closest the aspects addressed by LYDIAN. The interconnection of processes is abstracted by a communication graph model, which can be created interactively by the user. The provided algorithms implemented in JAVA can be run on top of the selected network. Hereby, the code for processes is simulated with JAVA threads. Users can also create own protocols by using the provided library functions. However, the user has no influence on defining timing behaviour for the created links. The animations show the graph model visualising events and states by displaying labels attached to links and processes. The visualisation mainly addresses to visualise the current states, but does not provide the user with information on other issues, such as causal relations and message complexity.

The work presented in [12] provides a nice object oriented framework which allows a simple specification of protocols in JAVA. The specification of protocols reflects the automaton model presented in textbooks on distributed algorithms such as [6]. The animation, because of its non-continuous nature, cannot give the user a picture about actions that happen concurrently and it does not address other aspects in educational visualisation. Moreover, the network is specified with the definition of a process, i.e. the code for each process identifies the respective neighbours of processes.

2.1 LYDIAN

LYDIAN [5, 11] is an environment to support the learning of distributed algorithms. LYDIAN provides a database of distributed algorithms and matching continuous animations. Students can write their own algorithm implementation in a high level language and test it with their desirable interconnection of processes, using LYDIAN's supported distributed systems simulator, DIAS, which is a simplification of the DSS simulator [13]. Moreover, LYDIAN's animation framework allows interactive demonstrations of distributed algorithms. The animations, which are built using the POLKA animations library [14], do not use a fixed interconnection of processes, but permit the students to create their own networks descriptions in a visual way and apply them to the respective algorithm and animation.

It is important for students to experiment with different network structures. LYDIAN provides an easy visual way to create their own network descriptions. This component is based on LEDA [7], a library for efficient data structures and algorithms, providing many algorithms to manipulate graphs and draw them efficiently. In LYDIAN the user simply draws a network based on a graph whose vertices and edges represent processes and links between processes, respectively. To achieve a pleasant layout, the user can apply a wide range of layout algorithms or move edges in a desirable way. For specifying the timing behaviour of the network, it is possible to choose among different options valid for all processes, but also define a specific behaviour for a link or process. When saving the graph, the files will be available for the simulator as well as for the animator. This way the user can see the same network in the animation as it was drawn in the graph editor.

LYDIAN is intended to serve a wide range of educational purposes. For instance LYDIAN can be used (i) to give a demonstration of prepared animations, (ii) let students to create their own network structures, which can be linked to the respective animations, or (iii) let students create own protocols, which can be executed on the simulator used by LYDIAN. The interaction with LYDIAN is based on a graphical user interface written in TCL/TK [10] which allows to access LYDIAN's archive of created resources as well as to create own resources.

Since in the execution of a protocol there are many components involved, LYDIAN introduced the concept of *experiments* in which the user can describe the properties for relevant components. An experiment contains information about

- the *protocol* the user wants to execute,
- the underlying *network structure* describing how processes are interconnected and the characteristics of the timing behaviour,
- a *trace file* in which during the execution of the algorithm significant events are stored, which can be traced by the user,
- and an *animation program* which can give a graphical representation of the events in the trace file.

The experiment is described in a single window containing all experiment-specific information. The user interacts with the experiment by pressing buttons representing different actions or modification choices. Having specified the desired parameters, the user can run the experiment, i.e. the respective protocol will be executed under the experiment-specific environment; or the user can animate an experiment, i.e. an animation under the experiment-specific environment will be shown. The user may animate the same execution multiple times (using the same trace-file), thus having the possibility of studying details that might otherwise go unnoticed. An experiment can also be saved for future use.

The interface of LYDIAN is designed such that the user can view all relevant information in one window and can change components of a selected experiment on the fly. For instance, in order to change the network, a user simply selects another network description file in the experiment and can run the experiment and view an animation as before, but using the new network structure.

3. DESCRIPTION OF STUDY

The evaluation is based on results taken from a basic undergraduate compulsory distributed system course at Chalmers University of Technology. We received answers from 50 students of the course. The questionnaire was anonymous. The main subject of most students was computer science and engineering, however there were also some students with other major subjects. Most students were in their final year of studies, but all of them had studied for at least two years in a program at our university. Hence, most of the students were experienced in programming. The percentage of female students participating in the study was around 10%. The age of students varied from 21 to 40, where most of the students were younger than 25.

For our study it is important to mention that neither the students nor the teachers had earlier class experience with LYDIAN. The teachers had to work out their own assignment, which would correspond to approximately one week of work for each student. The idea was to use LYDIAN for a programming assignment in which students had to implement some distributed algorithm based on elementary algorithms introduced in the course, i.e. the echo broadcast algorithm, logical clocks and voting. The students could choose to implement one of the following algorithms:

- leader election, based on an echo-broadcast approach,
- leader election, based on a voting approach,
- resource allocation, based on logical clocks.

The outline of the algorithms was given with the assignments, so the students essentially had to understand the algorithm and try to implement it. Parts of the algorithms, as the echo broadcast, were also available together with an animation, but needed to be changed to be usable within the algorithm. Most students decided to implement the algorithm based on the echo broadcast approach. This is probably because this concept seemed easier to realize. The algorithms were supposed to work on any arbitrary network structure.

For our study and evaluation our interest was focused on the following aspects:

- the students' performance in implementing an algorithm,
- how students test and reason about their implementation,
- whether/how much can LYDIAN help the students get an insight into distributed algorithms and their behaviour,
- whether students consider LYDIAN to be helpful,
- general feedback on the tools administration and maintenance.

4. OUTCOME AND OBSERVATIONS

The questions of this study and the answers received are summarised in Appendix A. In Appendix B, we examine the correlation between answers of students, in order to examine the relation between:

- the factors that helped the students most in getting a better insight into distributed algorithms,
- the help that the students got from using LYDIAN and their performance in carrying out the assignment,
- and the students performance in the assignment and their appreciation of LYDIAN.

Because of the anonymity of the answered questionnaire, we cannot associate the success of the students in their assignment with the answers that they gave to our questions. We simply trust their answers regarding their understanding of the assignment material. Below we discuss the results of our study.

- About 60% of the students were done with understanding how to use LYDIAN and with implementing and testing their solution in 1.5 working-days[1], while 80% of them were done in 2.5 to 3 working days. It should be mentioned here that the whole assignment was intended to take a maximum of 5 working days.
- Nearly half of the students tried the animation part —although it was not required in the assignment. As expected, animation stimulates the students' interest in studying.
- Every third student experienced some behaviour/property of the algorithm they implemented, which they had not thought about before. Of those students the majority thought that this experience helped them to understand the algorithm better. The animation part of LYDIAN can be even more beneficial in this aspect, since the same execution can be seen multiple times, and difficult scenario can be "scrutinised" and digested better by the students' minds.
- Approximately 60% of the students tested their implementations on more than one network structures.
- Although some students experienced some difficulties with using specific parts of the tool -mainly where documentation was not sufficiently detailed– the overall impression is that the class found the tool to be useful or relatively useful for understanding distributed algorithms.
- The majority of students who got better insight into their algorithm also tried the animation part. Maybe these students were more interested in the subject. However, the use of animations does not show any relation to the number of network topologies students used for testing purposes.
- Conforming to our expectation, the students who got more insight into the algorithm they implemented, tried more network structures in their testing. It seems worth the effort to try to stimulate students to test

[1]measured with 8 hours per working day

more and experiment more with their implementations. It is also good that LYDIAN's supported simulator provides this possibility.

- One main observation is that students who experienced unexpected behaviour of their algorithm mainly thought LYDIAN to be helpful.
- Students who used many network topologies did not think that LYDIAN is more helpful than those who did not use this feature. However the opinion is more biased, i.e. students who experienced with this feature have a stronger attitude whether they like or dislike LYDIAN in a course, while students who did not use it tended to be more neutral.
- A similar observation can be made for people who used the animation feature of LYDIAN.

An analysis of these results, from the perspective of seeing what such simulation-visualisation tools need to provide in order to be successfully used in class and how users (teachers, students) can be helped to improve the performance of the learning process, leads to the following lessons:

Tools: Since the "bottleneck" in understanding distributed algorithms is the actual concurrency and the parameters that can affect the step interleaving in each execution, it is very important for a tool which aims at facilitating the learning process in this area, to provide:
• means for the user to experiment by varying all parameters (this helps in revealing a scenario that the user had not thought about before),
• a good way to visualise concurrency,
• the possibility for a user to observe the same execution multiple times,
• a good documentation and good user guides.

Users: Since more experimentation is shown to be effective, it is important that instructors are explicit –e.g. as part of an assignment– in asking the students to use the visualisation/animation possibilities, as well as to experiment by changing the parameters of the system and by coming up with "special", unusual constellations.

5. CONCLUSION

Teaching is improved by pointing out unexpected properties/instances of the taught material. This is especially important in teaching distributed algorithms and systems, where there is a large number of parameters that affect the sequence of steps that a process will follow in each execution. LYDIAN was shown to be of good value in this respect, since it helped students to observe such instances, in an efficient manner.

Furthermore, teachers can help and get helped by using such tools in class to provide special case studies, to test cases, and to stimulate students to do own experimentation. Simulation and animation environments such as LYDIAN are shown to be useful in this respect, as well. Animation helps in understanding and also stimulates students.

LYDIAN helped in providing insight into the taught material, even though it had not been used in class by the teachers before and even though there were parts of the documentation which were not complete. The effort to improve on the supporting material –improved manuals, more examples– is expected to be appreciated and further increase the tools use-basis.

Acknowledgements

We wish to thank the Sven-Arne Andréasson and Ha Hoai Phuong, the teachers of the course, as well as the students who participated in this study.

6. REFERENCES

[1] Ben-Ari. Distributed algorithms in java. *SIGCSEB: SIGCSE Bulletin (ACM Special Interest Group on Computer Science Education)*, 29:62–64, 1997.

[2] M. Ben-Ari. Interactive execution of distributed algorithms. *Journal of Educational Resources in Computing (JERIC)*, 1(2es):2–8, 2001.

[3] S. Khanvilkar and S. M. Shatz. Tool integration for flexible simulation of distributed algorithms. *Software: Practice and Experience*, 31(14):1363–1380, Nov. 2001.

[4] B. Koldehofe. Animation and analysis of distributed algorithms. Master's thesis, Universität des Saarlandes, 1999.

[5] B. Koldehofe, M. Papatriantafilou, and P. Tsigas. Distributed algorithms visualisation for educational purposes. In J. Impagliazzo, editor, *Proceedings of the 4th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education (ITICSE-99)*, volume 31.3 of *SIGCSE Bulletin inroads*, pages 103–106, N.Y., June 1999. ACM Press.

[6] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann, San Francisco, CS, 1996.

[7] K. Mehlhorn and S. Näher. *LEDA: A Platform of Combinatorial and Geometric Computing*. Cambridge University Press, Cambridge, England, Jan. 1999.

[8] A. Mester, P. Herrmann, D. Jager, V. Mattick, M. Sensken, R. Kukasch, A. Ritter, S. Bunemann, P. Unflath, M. Bernhard, F. Austel, T. Alders, and A. Rohrbach. Zada: Zeus-based animations of distributed algorithms and communication protocols., 1995. *http://ls4-www.cs.uni-dortmund.de/RVS/zada.html*.

[9] Z. Moses, Y.and Polunsky, A. Tal, and L. Ulitsky. Algorithm visualization for distributed environments. In *IEEE Symposium on Information Visualization*, pages 71–78, 1998.

[10] J. K. Ousterhout. *Tcl and Tk Toolkit*. Addison-Wesley, 1994.

[11] M. Papatriantafilou and P. Tsigas. Towards a library of distributed algorithms and animations. In *4th International Conference on Computer Aided Learning and Instruction in Science and Engineering (CALISCE '98)*, pages 407–410, Gothenborg, Sweden, 1998.

[12] W. Schreiner. A java toolkit for teaching distributed algorithms. In *Proceedings of ACM ITiCSE*, pages 111-115. ACM press, 2002.

[13] P. Spirakis, B. Tampakas, M. Papatriantafillou, K. Konstantoulis, K. Vlaxodimitropoulos,

V. Antonopoulos, P. Kazazis, T. Metallidou, and S. Spartiotis. Distributed system simulator (DSS). In A. Finkel and M. Jantzen, editors, *Proceedings of Symposion on Theoretical Aspects of Computer Science (STACS '92)*, volume 577 of *LNCS*, pages 615–616, Berlin, Germany, Feb. 1992. Springer.

[14] J. Stasko. POLKA animation designer's package. Technical report, Georgia Institute of Technology, 1995.

[15] Visidia project: Visualization and simulation of distributed algorithms (2000), 2000. *http://dept-info.labri.u-bordeaux.fr/ stefan/.*

APPENDIX

A. QUESTIONS AND ANSWERS

1. Approximately how long have you used LYDIAN?

hours	0-4	5-8	9-12	13-16	17-20	21-40
students	2	11	17	7	9	4

2. Which algorithm did you select to implement?

Election with Echo	Election with Voting	Resource Allocation
41	5	4

3. Approximately how long did it take you to understand LYDIAN's interface?

hours	0-4	5-8	9-12	13-16	longer
students	30	8	7	2	4

4. Approximately how long time did you spend on studying the algorithm that you had to implement?

hours	1	2	3	4	15
students	31	11	4	3	1

5. Approximately how long did it take you to implement and test the same algorithm in LYDIAN?

hours	0-4	5-8	9-12	13-16	17-20	21-30
students	13	17	9	5	2	4

6. Did you try to use the animation part of LYDIAN?

yes	no
24	26

7. When or after you implemented the distributed algorithm in LYDIAN, did you experience any behaviour of the algorithm that you did not think about before?

yes	no
17	33

8. If yes, did this help you understand better the algorithm or other material discussed in the course?

yes	no
12	5

9. How many different network topologies did you use when testing your implementation?

number	0-1	2-5	15
students	21	28	1

10. LYDIAN is useful for understanding algorithm in distributed computing.

Strongly disagree	Disagree	Neutral	Agree	Strongly agree
4	10	14	21	1

11. LYDIAN is easy to use.

Strongly disagree	Disagree	Neutral	Agree	Strongly agree
17	22	7	4	0

12. Which parts of LYDIAN do you think need to be improved?

Documentation in general	Example implementations	Interface
39	17	27

Stability	Documentation on network creation	No comment
14	21	5

13. What is your year of study?

year	3	4	5	6	?
students	9	30	1	1	9

14. Age.

age	21-23	24-26	older
students	27	13	5

B. CORRELATION BETWEEN ANSWERS

Students who experienced some behaviour of their algorithm they did not expect before, gave the following answers to these questions:

1. Did you try to use the animation part of LYDIAN?

yes	no
10	7

2. Approximately how long did it take you to implement and test the same algorithm in LYDIAN?

hours	0-4	5-8	9-12	13-16	17-20	22
students	4	8	2	2	0	1

3. How many different network topologies did you use when testing your implementation?

number	0-1	2-5	15
students	7	10	1

4. LYDIAN is useful for understanding algorithm in distributed computing.

Strongly disagree	Disagree	Neutral	Agree	Strongly agree
1	1	3	12	0

The following groups of students thought as follows about LYDIAN being helpful:

1. Students tested their algorithm only with one network topology or were not aware of them.

Strongly disagree	Disagree	Neutral	Agree	Strongly agree
2	2	9	8	0

2. Students tested their algorithm with multiple network topologies.

Strongly disagree	Disagree	Neutral	Agree	Strongly agree
2	8	5	13	1

3. Students who tested the animation part of LYDIAN.

Strongly disagree	Disagree	Neutral	Agree	Strongly agree
0	7	4	13	0

4. Students who experienced behaviour they did not expect before.

Strongly disagree	Disagree	Neutral	Agree	Strongly agree
1	1	3	12	0