Case Study: Visualizing Computer System Programming Concepts for Education

Florian Schmidt[1]
Technische Universität Berlin
Berlin, Germany

Johannes Ohlemacher
Technische Universität Berlin
Berlin, Germany

Vincent Hennig
Technische Universität Berlin
Berlin, Germany

Odej Kao
Technische Universität Berlin
Berlin, Germany

Jan Nordholz
Technische Universität Berlin
Berlin, Germany

Conference Key Areas: Open and online teaching and learning, New Complexity quest in engineering sciences

Keywords: interactive visualization, large scale courses, complex animations, system programming

ABSTRACT

In the area of computer system programming, theoretical concepts between hardware and software solutions need to be explained. Even though this topic is highly practical, as it covers the theoretical concepts of any modern machine like computers and smartphones, it is still difficult to provide easy access to any practical experience as the gap between current hardware solutions and basic implementation of operating system processes is too large for undergraduate students. As it is not possible to test those concepts in an applied way, we aim to provide an interactive web simulation framework called SysprogInteract. SysprogInteract provides modern animations of the behaviour of single computer components as well as more complex ones that show the relationship between components. The goal is to provide high quality media content, accessible any time to any student, providing

---

[1] Corresponding Author
F. Schmidt
Florian.schmidt@tu-berlin.de

individual feedback and fostering exchange between students through the overall platform. This paper presents the first animated hardware component (CPU) of SysprogInteract, visualizing the concept of scheduling processes. Based on this tool, we show how it can be integrated within a lecture with more than 800 students at the TU Berlin, Germany. Furthermore, we present a case study with 59 students, showing a high usability through the System Usability Scale with a score of 79.5; more than 86% of students responded they would use this application for exam preparations. Additionally, we present the overall concept of the platform, which is ongoing work.

1 INTRODUCTION

Computer system programming describes theoretical concepts between hardware and software solutions, which are present in operating systems. Even though this topic is highly practical, as it covers the concepts of any modern machine like computers and smartphones, it is still difficult to provide access to any practical experience as the gap between current hardware solutions and basic implementation of operating system processes is too large for undergraduate students in the first year of their studies.

Furthermore, in recent years the number of students in our course Systemprogrammierung at the Technische Universität Berlin (TU Berlin) was constantly growing, such that currently more than 800 students attend our lectures, making it almost impossible to provide individual feedback.

Due to this lack of individual feedback and due to the difficulties of teaching these concepts in an applied way, we have decided to create an interactive web simulation framework called SysprogInteract. With this framework we provide a set of modern animations, some of them rendering the behaviour of a single computer component, others simulating several components together with the relationships between them. The goal is to create high quality media content, accessible any time to any student, providing individual feedback and fostering exchange between students, while at the same time integrating seamlessly into the educational process within the course.

Thus, this paper presents the following core contributions:

- Description of the open source, interactive web application SysprogInteract, showing in detail the different simulations for the hardware components of CPU scheduling, memory placement and resource management.
- Integration possibilities of the application into the educational process within the undergraduate course with more than 800 students at TU Berlin, Germany.
- Showing preliminary results of a user study with 59 students in order to verify the acceptance, interactivity and motivation to use this tool.

Outline: The rest of the paper is structured as followed. The next section describes the state of the art of thematically and methodically related applications and animations for computer system programming education. Afterwards, we continue with the description of our current web application SysprogInteract and our vision of the final architecture in section 3. We continue with information on how SysprogInteract can be integrated within a large scale course in section 4. Section 5 presents the user study we conducted to evaluate the interactivity and motivation for students using SysprogInteract. Lastly, we conclude this work in section 6.

2 RELATED WORK

Discussing the effectiveness of animation in educational environments, Rieber [1] has shown how animation can be integrated beneficiently. While he gave general recommendations for educational animations in different disciplines from geography to physics, Lawrence et al. [2] have focused on the effectiveness of algorithm animation in particular. English and Rainwater [3] studied the use of animations in an operating systems course and concluded that they proved to be more effective for teaching procedural topics than basic concepts. To make use of these benefits Jones and Newman [4] developed a simulated operating system. Like SysprogInteract, their tool simulates essential operating system parts, however, it was last updated in 2003 and is therefore outdated.

More recent developments of educational simulation software are rather similar approaches, mostly focusing on CPU scheduling. The tool developed by Suranauwarat [5] visualizes various process scheduling algorithms like FCFS, RR, SJF, SRTF and MLFQ. It also features a practice mode in which students can check their predictions on the algorithms or test alternative scheduling decisions. Although the visualization is quite detailed, it lacks a user-friendly evaluation part, e.g. time-based graphs. A short evaluation may be sufficient as the process queue is limited to a size of four. Unlike SysprogInteract, this tool does not allow its users to study the behaviour of the supported algorithms in larger test setups.

A similar approach was proposed by Kotalny and Spinczyk [6]. Their tool AnimOS features the same algorithms (apart from MLFQ), however, it does not include a practice mode. Still, it is more flexible as it allows the user to define custom algorithms and to simulate an unlimited amount of processes. But especially that makes the lack of proper presentation for the collected performance data quite severe.

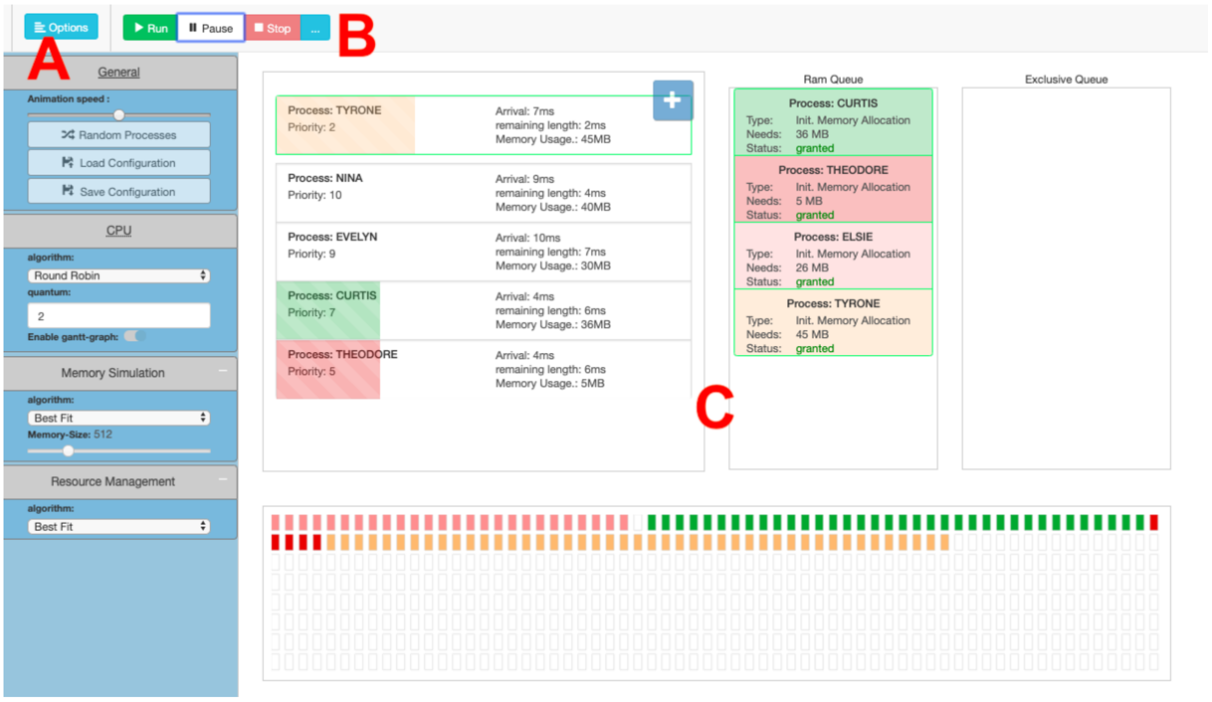
With OSLAB, Zareie and Najaf-Zadeh [7] are combining two essential parts of operating systems' lectures: in addition to CPU scheduling algorithms they also simulate memory management. OSLAB comes with only two predefined algorithms (FCFS, RR) for both parts but allows, like AnimOS, to add further ones. Due to the layout, it is not possible to observe memory and CPU behavior simultaneously. Like the tools discussed above, OSLAB also lacks appealing metric presentation.

None of these tools are aiming to simulate multiple components of an operating system at the same time, so it is impossible to observe interdependencies between components and selected algorithms and giving intuitive possibilities to monitor the performance of different configurations. Thus, we will now introduce our tool SysprogInteract which provides the possibility of visualizing such dynamic interdependencies between different components.

3 SYSPROG-INTERACT

The course of Systemprogrammierung focuses on selected algorithmic approaches employed by operating systems to handle the computation on conventional computer hardware. Thus, the course concentrates on the following main hardware components: Central Processor Unit (CPU), Random Access Memory (RAM) and Hard Disk Memory (HDD). Additionally, the concept of a process is introduced, which embodies a certain computational workload. The operating system faces the task of allocating the available hardware resources to each process, such as "scheduling", i.e. distributing the CPU time among the processes, and partitioning memory in order to load the code and data of each process. Furthermore, processes might make use of the same resource, which may eventually result in a conflict when these accesses coincide. Those need to be resolved by deadlock handling mechanisms.

SysprogInteract provides configurable animations for each of these individual tasks, illustrating several different algorithmic solutions. The application allows to choose between visualizing individual components or a complete system, showing all the relations between its components. In order to meet the expectations of a modern user interface, we use the Javascript library D3 [8] which gives us the opportunity to create several different aesthetic visualizations.



Animation speed :

algorithm: Round Robin
quantum: 2
Enable gantt-graph:

algorithm: Best Fit
Memory-Size: 512

algorithm: Best Fit

Process: TYRONE
Priority: 2
Arrival: 7ms
remaining length: 2ms
Memory Usage.: 45MB

Process: NINA
Priority: 10
Arrival: 9ms
remaining length: 4ms
Memory Usage.: 40MB

Process: EVELYN
Priority: 9
Arrival: 10ms
remaining length: 7ms
Memory Usage.: 30MB

Process: CURTIS
Priority: 7
Arrival: 4ms
remaining length: 6ms
Memory Usage.: 36MB

Process: THEODORE
Priority: 5
Arrival: 4ms
remaining length: 6ms
Memory Usage.: 5MB

Process: CURTIS
Type: Init. Memory Allocation
Needs: 36 MB
Status: granted

Process: THEODORE
Type: Init. Memory Allocation
Needs: 5 MB
Status: granted

Process: ELSIE
Type: Init. Memory Allocation
Needs: 26 MB
Status: granted

Process: TYRONE
Type: Init. Memory Allocation
Needs: 45 MB
Status: granted

Fig. 1. SysprogInteract user interface structure

Fig. 1 shows the overall structure of the user interface. Part A shows the option bar (blue), which contains the configuration settings for all of the algorithms. Furthermore, the user can decide to use the overall visualization of all components or select parts of the components, which is useful while some components have not yet been introduced to the students. Part B of the image shows the stepwise control bar. Like a music player, users can start, stop, and pause the animation and go stepwise forward and backward in order to closely inspect crucial parts of algorithms. Part C shows the animations and advanced diagrams.

Currently we support the following components and features:

- CPU scheduling algorithms: First the user has to specify the set of processes whose execution shall be simulated. The scheduling methods FCFS, RR, SJF, SRTF and MLFQ are supported and animated. Fig. 3 shows the resulting diagram, which is also used when introducing these algorithms in the lecture. Fig. 4 shows process information containing average waiting time, longest waiting time, etc.
- Memory placement strategies: First-, best-, worst-, rotating first fit are integrated. Running processes can be configured to use memory resources, which are illustrated in the bottom part of Fig. 1. Average memory usage and further metrics are presented as an additional diagram (see Fig. 5).
- Resource management strategies: For resource management, on the upper right part of C animations are presented, but additionally a diagram as shown in Fig. 2 is animated to illustrate the complete history of resource usage for each process. Again, this diagram can be used within the lecture.
- Saving and loading of configuration files: It is possible through the option bar to save and load configurations of processes and algorithms. This enables the possibility to prepare showcases and tasks.

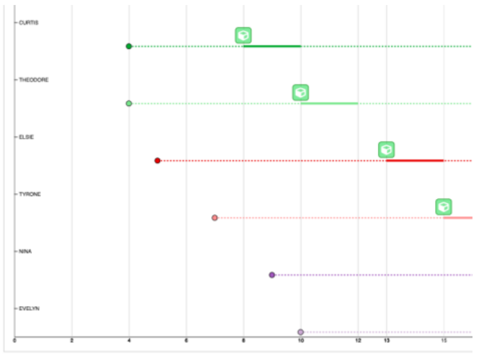


Fig. 2. Resource management

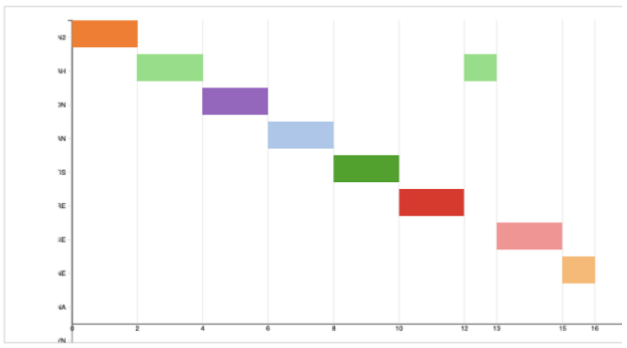


Fig. 3. CPU scheduling

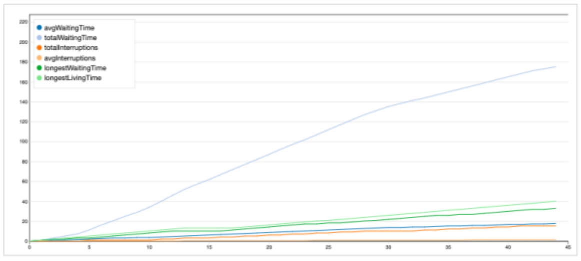


Fig. 4. Process metrics

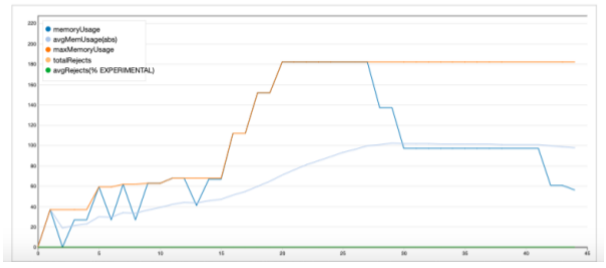


Fig. 5. Memory metrics

Another feature worth considering is the integration to edit algorithms by students themselves, in order to provide more practical usage of simulation by creating or adapting algorithms. This is already possible by directly editing the publicly available code of our application, but the usability of such a feature may be increased in the future by providing an editor inside the web application.

We hope that SysprogInteract will be used in further education facilities to gain an impact in the computer science. Thus, we published SysprogInteract under the open source Apache 2.0 license on the Github[2] with a demo page[3]. For students as well as teachers, videos have been produced to show the usage of the tool and the setup on own servers. The videos are collected on a Youtube playlist[4].

4 INTEGRATION INTO EDUCATIONAL PROCESS

The application can be used within the educational process in various ways. We will now describe different scenarios of how the tool has already been used or how it could be used within the course.

Within the lecture, SysprogInteract is used for demonstrating the different algorithms during the presentation. While running the animations, the lecturer can pause and go steps back at any time in order to showcase key phases of the illustrated approaches. As the solution provides visualizations for interdependencies of system components, different chapters of the course are visually connected and may give further motivation why specific algorithms are selected for demonstration.

Additionally, SysprogInteract provides the possibility to save and load configurations (for process definition and algorithm configurations for the individual components). This enables the creation of tasks where results can be analyzed through the simulation. Such exercises can be used for exams or interactive sessions within small group exercises. In addition, the tool is used for group discussions, where students explain certain concepts to each other or give predictions on the overall outcome for certain statistics.

The code of the application is available as open source, such that students could be also asked to add further algorithms. This has not been tested within our course yet, but we plan to give this option to students who think they are capable of doing so.

As the tool is developed as web application, students can also use it at home for individual learning. This is useful, as it can be used for homework and preparation for the final exam. Besides the video material of the usage of SysprogInteract, we also aim to publish open access material for teachers such as slides or exercises to improve the overall usability and to facilitate the integration into further courses at other universities.

---

[2] https://github.com/citlab/SysprogInteract

[3] https://citlab.github.io/SysprogInteract/

[4] https://www.youtube.com/watch?v=CzDeF80RYHs&list=PL2wkohQ2DA3vtUHH2F6yO2iwwX5O_QOTl

5 USER STUDY

Within the course, we conducted a user study with 59 students in order to evaluate the usability of the tool and motivational aspects for education. For those interviews, we gave students access to the application without any instructions and a paper-based questionnaire. Students were allowed to decline to answer any questions.

The participants were enrolled in computer science (39), computer engineering (10) or other major (9). 55 persons pursue a Bachelor degree and 1 person a Diploma degree. One of them is in the 1st, 46 in the 2nd, 1 in the 3rd, 5 in the 4th, 3 in the 6th and 1 in the 7th or higher semester. 11 participants identified as female and 45 as male.

For the usability study, we used the questionnaire developed by Brooke [9]. It consists of 10 questions in order to evaluate the usability of applied applications in the industry. We added further questions at the end in order to evaluate the motivation of using the application within the educational context. We integrated the possibility for open answers in order to give our students the opportunity to submit additional feedback. All questions and results are shown in the following diagrams (Fig. 6).

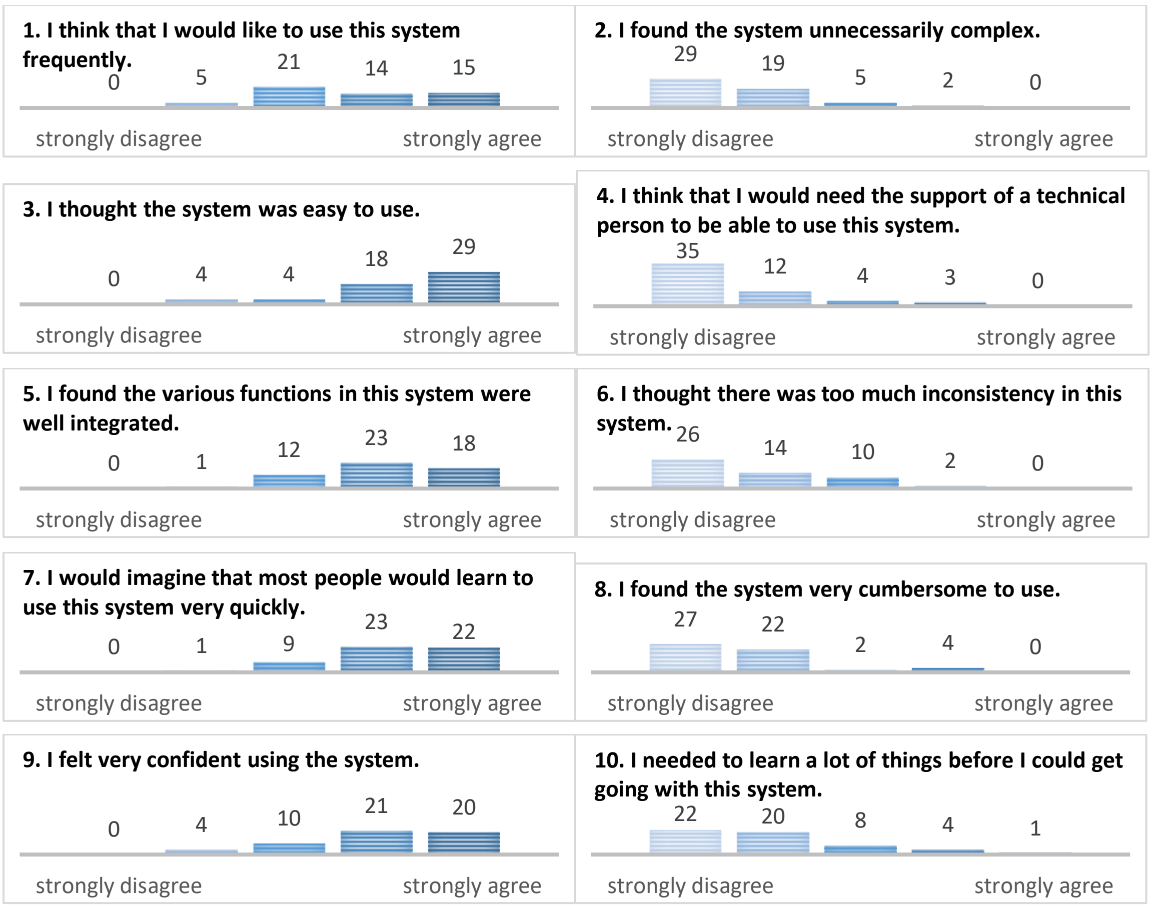


Fig. 6. Main ten usability questions from Brooke [9].

Diagrams 1 to 10 provide the results of the questionnaire by Brooke [9]. They all use a linear scale from 1 to 5, where 1 denotes a strong disagreement and 5 a strong agreement to the provided statement. Brooke [9] developed the System Usability Scale (SUS), which serves as a standardized test to evaluate applications, thus making them comparable. The scale ranges from 0 to 100 points. Even though the scale is linear within its range of 0-100 possible points, a large number of studies has shown that it cannot be inferred from the linearity of the scale that an average application gets a scale of 50 points for usability. Thus, Brooke presented in [10] further classifications to map the SUS to other scales.

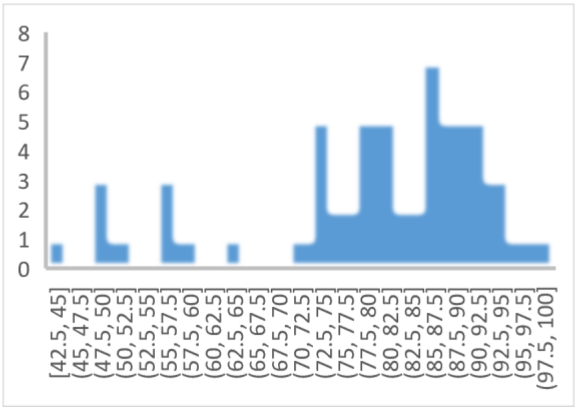


Fig. 7. Histogram of SUS scores

The usability questionnaire shows that for all statements the tendencies lean towards the positive side, such that most declared that e.g. the system is easy to use. Looking at the individual SUS, we see 9 people giving scores between 50 and 60 points, while the rest gave higher SUS, up to the maximum of 100. Fig. 7 shows the distribution of all individual SUS. The average SUS is 79.52 points, with a standard deviation of 14.23 points. The confidence interval is [75.81, 83.23] with confidence of 0.95. Lewis and Sauro [11] showed in their study that the average is between 70.1 and 62.1, standard deviation is 21.7 and 22.2 for more than 2,300 and 300 individual SUS, respectively for two different data sets. Thus, our average is higher and standard deviation smaller than the shown study. Using the further classifications of the given scale by Brooke [10], the average lies within the adjective rating for a good usability. Next, we show the results of the additional interview questions (Fig. 8).

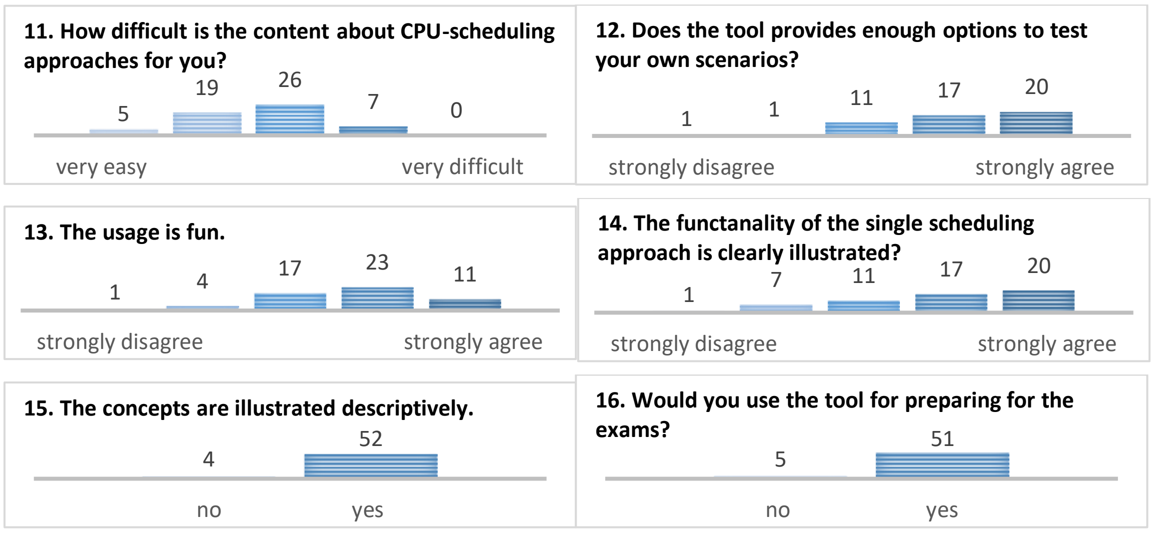


Fig. 8. Further questions of the user study questionnaire.

Question 11 shows that the overall topic of scheduling algorithms is perceived as moderately difficult. Questions 12, 14 and 15 show the tendencies that the application functions positively for learning purposes. Furthermore, question 16 shows that 86.44% of students would use this tool as additional material for exam preparation. Question 13 evaluates whether the application is fun to use, to which more than 57% of students responded favorably, i.e. agreed or agreed strongly. For the open questions, we got suggestions for improvements such as adding further scheduling algorithms and making the processes configurable in order to select your own process name, which is currently randomly assigned. Furthermore, one participant stated that the presentation of the scheduling algorithms as Gantt chart (cf. Fig. 3) is great as it shows the connection to the lecture.

Summarizing, the SUS evaluation indicates that our application has a good usability. Also, the students emphasized that they are going to use this tool for exam preparation. Further improvements will be done in the future, e.g. by extending the visualization, and we plan to conduct follow-up evaluations to continuously monitor the usability of our application.

6 CONCLUSION

This paper introduced SysprogInteract, a web application for animating complex concepts of low-level programming and operating system design in an educational context. We have described the details of our application and presented possible integrations into the educational process. Furthermore, a user study shows promising results as students are motivated to interact with this application and think the tool will help them to gain deeper understanding of the algorithms. We plan to continuously evaluate while developing further animations. We also hope that other universities will integrate this tool into their courses, so we can learn from their integration and adapt our application to fulfill the needs of all its users.

REFERENCES

[1] Rieber, Lloyd P. "Animation in computer-based instruction." Educational technology research and development 38.1 (1990): 77-86.

[2] Lawrence, Andrea W., Albert M. Badre, and John T. Stasko. "Empirically evaluating the use of animations to teach algorithms." Visual Languages, 1994. Proceedings., IEEE Symposium on. IEEE, 1994.

[3] English, Brian M., and Stephen B. Rainwater. "The effectiveness of animations in an undergraduate operating systems course." Journal of

Computing Sciences in Colleges 21.5 (2006): 53-59.

[4] Jones, David, and Andrew Newman. "RCOS. java: A simulated operating system with animations." Teaching package 1 (2001).

[5] Suranauwarat, Sukanya. "Using an interactive animated tool to improve the effectiveness of learning CPU scheduling algorithms." Frontiers in Education Conference (FIE), IEEE, 2015.

[6] G. Kotainy and O. Spinczyk, “AnimOS,” AnimOS CPU-Scheduling. [Online]. Available: https://ess.cs.tu-dortmund.de/Software/AnimOS/CPU-Scheduling/.

[7] Zareie, Farzaneh, and Mahsa Najaf-Zadeh. "OSLab: A Hand-on Educational Software for Operating Systems Concepts Teaching and Learning." (2013): 31.

[8] Bostock, Michael, Vadim Ogievetsky, and Jeffrey Heer. "D^3 data-driven documents." IEEE Transactions on Visualization & Computer Graphics 12 (2011): 2301-2309.

[9] Brooke, John. "SUS-A quick and dirty usability scale." Usability evaluation in industry 189.194 (1996): 4-7.

[10] Brooke, John. "SUS: a retrospective." Journal of usability studies 8.2 (2013): 29-40.

[11] Lewis, James R., and Jeff Sauro. "The factor structure of the system usability scale." International conference on human centered design. Springer, Berlin, Heidelberg, 2009.