Quantitative Evaluation of Flash-based Educational Visualizing Simulator

Kei Takeichi, Yoshiro Imai, Kazuaki Ando, Tetsuo Hattori
Kagawa University, 2217-20 Hayashi
Takamatsu City, Kagawa 761-0113, Japan
E-mail: {hattori, imai, ando}@eng.kagawa-u.ac.jp

Yusuke Kawakami
DynaxT Co., Ltd., 2271-6 Hayashi
Takamatsu City, Kagawa 761-0113, Japan
E-mail: riverjp2002@gmail.com

Abstract

A Flash-based simulator of CPU scheduling has been developed and utilized for educational visualization in the class of university lecture. We have designed and implemented it with Flash-based scripting language in order to execute it as a stand-alone application as well as in various browsing environment such as Microsoft IE, Google Chrome and/or FireFox (Mozilla). Based on questionnaire for our simulator in the lecture, its quantitative evaluation has been carried out by means of statistical analysis. Our report describes overview of our Flash-based simulator and the results of the above quantitative evaluation.

Keywords: Educational visualization, Questionnaire-based Evaluation, Statistical analysis

1. Introduction

As a matter of course, software system covers many important areas from fundamentals to applications. This time, we focus on Operating System, and particularly CPU scheduling algorithm. It must be cover several kinds of themes that students should understand during their school days. A simple algorithm, namely FCFS (First Come and First Served) is very natural so that it is one the most fundamental strategies to decide its priority for users, clients, processes/tasks and so on. Priority based algorithm is another candidate to determine the order of execution. It is very significant idea to choose a suitable item around potentially selected targets. It means which is better, or which is optimal of them. SPTF (Shortest Processing Time First) is to be chosen as one of the priority-based algorithms in this study. In other view point, RR (Round Robin) is evaluated as a policy of algorithm to realize equality of opportunity around the targets. It is a little complicated but very useful strategy to choose item with equal opportunities.

In order to teach students these above algorithms efficiently, we had better utilize some suitable educational tool to visualize their behavior and results for specific conditions[1,2]. A visual simulator is one of the useful solutions to provide educational tool(s) for students who wants to understand such theme of information processing education in an efficient and effective manner[3]. In this study, we have developed some useful educational tools to demonstrate practical CPU scheduling algorithm(s) and provide visual understanding for students in an effective way.

2. Overview of Visualizing Simulator

In a lecture of Operating System of our university, most important algorithms of CPU scheduling are FCFS,SPTF, and RR, we think. They are very trivial but sometimes very useful in the real operating systems. So we have employed these above algorithms as fundamental procedures to decide CPU scheduling in our visualizing simulator. The above three procedure based on FCFS, SPTF and RR are as follows;
(1) FCFS is very much simple, but clearly well-defined strategy to decide next candidate to be performed, just like First-In First-Out(FIFO queue). This algorithm is sometimes the target to be compared with other algorithm(s). And moreover, a result applied by this algorithm will be not so worse than other ones derived from complicated algorithm(s).
(2) SPTF is one of the most famous priority-based assigning/allocating algorithm. Whenever every event happens, namely conditions have changed, it must be investigated which candidate has the best priority at that time. So we had better call this algorithm Shortest Remaining Processing Time First (SRPTF), because we must consider not the total processing time but remaining processing one in order to decide which candidate has the best priority at that time or later.
(3) RR is a typical non-priority based algorithm in order to provide 'equality of opportunity' which can realize taking turns at it. This algorithm can retrieve candidates which are waiting for service and select/assign one of them who wait for the longest time or longer than others. It is a little complicated for beginners to understand details of RR-based procedure and/or develop a kind of corresponding programs. And we may sometimes meet its results with not suitable performance for specific applications. But equality of opportunity is very important for several users to receive their necessary services.

3. Quantitative Evaluation

This section presents quantitative evaluation of our visualizing simulator. As one of the quantitative evaluation for our CPU scheduling simulator, at first, we compare the execution time of simulation of native Flash player with ones on the below three major browsers. The result is summarized in the following Table 1.

Table 1. Comparison of Execution Time(s) between Different Environments.

Host Application	Execution Time
Flash Player (Ver.10) Stand alone	32.25 (sec.)
MS-Internet Explorer 11.0.9600	28.95 (sec.)
Mozilla FireFox 35.0.1	35.19 (sec.)
Google Chrome 40.0.2214.11 m	93.61 (sec.)

As another quantitative evaluation for our simulator, secondly, we have carried out questionnaire in the classroom lecture of Operating System in our university after using our simulator. The questionnaire includes following six questions;

Q#1 Is it easy to utilize this simulator? (yes: 2,neutral:1, no:0)
Q#2 Is it effective to learn CPU scheduling algorithm with this visualizing simulator? (yes: 2, neutral:1, no:0)
Q#3 Do you understand CPU scheduling algorithm more suitably with this simulator? (yes: 2, neutral:1, no:0)
Q#4 Are you interesting in CPU scheduling algorithm by means of this simulator? (yes: 2,neutral:1, no:0)
Q#5 Are you interesting in other themes of Operating System after usage of this simulator? (yes: 2, neutral:1, no:0)
Q#6 Do you need to utilize another type of simulator in order to learn Operating System? (yes: 2, neutral:1, no:0)

Our questionnaire described before can obtain just 20 answers from students of the class because of carrying out on a voluntary basis, although we used to have the class for Operating System with 40 students or more. The result of such a questionnaire is summarized in Table 2.
Q#1 has 17 numbers of answer "yes" per 20 students (i.e. 85%), Q#2 has 18 numbers of answer "yes" per 20 students (i.e. 90%) and Q#3 has 12 numbers of answer "yes" per 20 students (i.e. 60%). From the questionnaire, many students do feel easy to utilize our CPU scheduling

simulator and consider to be effective for learning CPU scheduling algorithm by means of using our simulator. And majority, namely six out of ten, of replying students understand CPU scheduling algorithm more suitably with this simulator.

Table 2. Result of Questionnaire about our Simulator.

Student	**Q#1**	**Q#2**	**Q#3**	**Q#4**	**Q#5**	**Q#6**
S01	2	2	2	1	1	2
S02	2	2	1	1	1	2
S03	1	1	0	2	1	2
S04	2	2	2	1	2	2
S05	2	2	1	1	1	1
S06	2	2	2	1	1	2
S07	2	2	1	1	1	2
S08	2	2	2	1	1	2
S09	2	2	2	1	1	1
S10	2	2	2	1	1	2
S11	2	2	1	1	1	2
S12	2	2	2	1	1	1
S13	1	2	1	1	1	2
S14	2	1	2	1	1	2
S15	2	2	2	2	1	2
S16	2	2	1	1	1	2
S17	2	2	1	2	1	2
S18	2	2	2	2	2	2
S19	2	2	2	1	1	2
S20	1	2	2	2	2	2

At the same time, however, Q#4 has just 5 numbers of answer "yes" per 20 students (i.e. 25%) and Q#5 has only 3 numbers of answer "yes" per 20 students (i.e. 15%). In order to perform test of independence among Q#1, Q#2 and Q#3, we will demonstrate to calculate "χ^2 test of goodness-of-fit" for relation between results from Q#1 and Q#2 as well as one for Q#1 and Q#3, respectively. Relation between results from Q#1 and Q#2 is expressed in the left-hand of Table 3, while relation for Q#1 and Q#3 is done in the right-hand. The former has 2 x 2 table-items and the latter has 2 x 3 ones.

Table 3. Relation between Results from Q#1 and Q#2 (left-hand) & from Q#1 and Q#3 (right-hand).

		Q#2			Q#3		
		yes	neutral	no	yes	neutral	no
Q#1	yes	16	1		11	6	0
	neutral	2	1		1	1	1
	no						

Based on Table 3, a two-way contingency table for Q#1 and Q#2 can be introduced, which is shown in Table 4, while another two-way contingency table for Q#1 and Q#3 can be also done, which is shown in Table 5.

Table 4. Two-way Contingency Table for Q#1 and Q#2.

		Q#2		SUM$_R$
		yes	neutral	
Q#1	yes	16(18*17/20)	1(2*17/20)	17
	neutral	2(18*3/20)	1(2*3/20)	3
SUM$_C$		18	2	20

χ^2: goodness-of-fit statistic for Table 4 can be calculated in the following expression (Eq-1);

$$\chi^2 = \sum_{i=1}^{2}\sum_{j=1}^{2}\{(y_{ij} - \frac{sum_R * sum_C}{Total})^2 / \frac{sum_R * sum_C}{Total}\}$$

$$= \{16 - (18*17/20)\}^2 /(18*17/20) + \cdots$$

$$+ \{1 - (2*3/20)\}^2 /(2*3/20) = 2.135 \qquad \text{(Eq-1)}$$

As described in expression (Eq-1), degree of freedom for Table 4 is ν = (2-1) x (2-1) = 1. So we can have χ^2 α=0.05(ν =1) = 3.8415 from the χ^2 distribution table. "Statistical independence" between results from Q#1 and Q#2 can be confirmed so that users of our simulator not only consider it to be easy to utilize but also recognize effectiveness to learn CPU scheduling algorithm with it respectively and independently.

Table 5. Two-way Contingency Table for Q#1 and Q#3.

		Q#3			SUM$_R$
		yes	neutral	no	
Q#1	yes	11(12*17/20)	6(7*17/20)	0(1*17/20)	17
	neutral	1(12*3/20)	1(7*3/20)	1(1*3/20)	3
SUM$_C$		12	7	1	20

Just like the same way, χ^2: goodness-of-fit statistic for Table 4 can be calculated in the following expression (Eq-2);

$$\chi^2 = \sum_{i=1}^{2}\sum_{j=1}^{2}\{(y_{ij} - \frac{sum_R * sum_C}{Total})^2 / \frac{sum_R * sum_C}{Total}\}$$
$$= \{11-(12*17/20)\}^2/(12*17/20)+\cdots$$
$$+\{1-(1*3/20)\}^2/(1*3/20) = 0.1365 \qquad \text{(Eq-2)}$$

As described in expression (Eq-2), degree of freedom for Table 5 is ν = (2-1) x (3-1) = 2. So we can have χ^2 α=0.05(ν =2) = 5.9915 from the χ^2 distribution table. "Statistical independence" between results from Q#1 and Q#3 can be confirmed so that users of our simulator not only consider it to be easy to utilize but also understand CPU scheduling algorithm more suitably with our simulator respectively and independently.

In other words, both of results from Q#1 and Q#2 are not only very good scores, namely 85% of the former's answers are "yes" and 90% of the latter's answers are "yes", but also the two scores are statistically independent each other, namely there is no reason that one scores can become good because another scores are good. And major part (i.e. 60%) of users, whose answers from Q#3 are "yes", understand CPU scheduling algorithm more suitably by means of our simulator independently from its operability.

4. Conclusion

The paper describes an Adobe-Flash based educational visualizing simulator for students to learn CPU scheduling algorithm graphically and practically. Our Flash-based simulator can execute on the major Web browsers such as Microsoft Internet Explorer, Mozilla FireFox and Google Chrome and provide efficient explanation for students of lecture "Operating System".

As quantitative evaluation for our simulator, we have carried out questionnaire for the students using the simulator and apply statistical analysis for the results of the questionnaire. We can obtain and confirm a good results from the above performance through analysis.

Namely,

(1) It can be confirmed that users of our simulator not only consider it to be easy to utilize but also recognize effectiveness to learn CPU scheduling algorithm with our visualizing simulator respectively and independently.

(2) It can be confirmed that users of our simulator not only consider it to be easy to utilize but also understand CPU scheduling algorithm more suitably with our simulator respectively and independently.

References

1. Sarjoughian,H., Yu Chen, Burger,K. : A component-based visual simulator for MIPS32 processors. Proceedings of 38th Annual Conference on Frontiers in Education(FIE 2008)., pp. F3B-9 – F3B-14 (October 2008)..
2. Kabir,M.T., Bari,M.T., Haque,A.L. : ViSiMIPS: Visual simulator of MIPS32 pipelined processor. Proceedings of 6th International Conference on Computer Science & Education (ICCSE2011), pp. 788 – 793 (August 2011).
3. Imai,Y., Tomita,S., Niimi, H., Kitamura,T. : Web-Based Computer Visual Simulator. Technology Enhanced Learning (IFIP International Federation for Information Processing), Volume 171, pp 111–120(Summer 2005).
4. Lee, K.-C., Lee, J. : Programming physics softwares in Flash. Computer Physics Communications, Volume 177, Issues 1-2, pp 195–198 (July 2007).
5. Stodulka, P., Privitzer, P., Kofranek, J., Tribula, M., Vacek, O. : Development of WEB accessible medical educational simulators. Proceedings of 6th EUROSIM Congress on Modelling and Simulation, 6 pages (September 2007).
6. Imai,Y., Takeichi, K. : Development and Evaluation of Adobe Flash based CPU Scheduling Simulator Executable on Major Multiple Web Browsers. Proceedings of 2015 IEEE International Conference on Intelligent Networking and Collaborative Systems (INCoS-2015, Tamkang University, Taipei, TAIWAN(RoC)), pp.149–155 (September 2015).