A SIMULATOR FOR TEACHING AUTOMATAS AND FORMAL LANGUAGES
FLyA

J. Raymundo Marcial-Romero, Pedro A. Alvarez Contreras
Facultad de Ingeniería, Universidad Autónoma del Estado de México, Toluca, Edo. de México, México

Héctor A. Montes Venegas, J. Antonio Hernández Servín
Facultad de Ingeniería, Universidad Autónoma del Estado de México, Toluca, Edo. de México, México

Keywords: Finite automata, Context-free grammar, Web simulator, Supporting tools for teaching, Turing machine.

Abstract: Finite automata theory is taught in almost every computing program. Its importance comes from the broad range of applications in many areas. As any mathematically based subjects, automata theory content is full of abstractions which constructively explain theoretical procedures. In computing Engineering programs, teaching is mainly focus on procedures to solve a variety of Engineering problems. However, to follow this procedures using a conventional approach can be a tedious task for the student. In this paper, a computer based software as a supporting tool to aid in teaching Automata Theory is presented. The use of an educational methodology to design the tool, remarkably contributed to the acceptance of the software amongst students and teachers as compared with existing tools for the same purpose.

1 INTRODUCTION

The constant demand of highly-qualified Computer Science (CS) graduates around the world has made necessary to continuously improve the teaching techniques for different subjects. One of the strategies to improve teaching has been the development of software based on educational methodologies. In particular, automata theory is essential in almost every CS curricula due to its applicability in many areas such as compilers, robotics, circuit design, among others. Nowadays, there exists software that helps students to give solution to typical problems in automata theory, however they lack an educational methodology development which brings up some difficulties as supporting tools for teaching. The console-type based methodologies, for example, have the disadvantage of not being user friendly. They work as a "black box", meaning that they only show partial results due to their poor interactivity. Examples of these tools are: Research with an automaton (Charras and Lecrog, 1997) and Finite State Machine (FSM) Simulator (Head, 1997).

On the other hand, there are those that present a graphical environment to interact with the user, however they have the disadvantage of requiring additional software to be used. They also have certain limitations such as: limited input alphabet, limited number of states and they work as well as a " black box". Among others things which are not desirable from an educational point of view (Marqués, 1995). Among some of these tools, it can be mentioned: Automaton Simulator (Burch, 2001), DFA Simulation (Budowski, 2001), Visual Automata Simulator (Bovet, 2006), JFLAP (Rodger, 2006), SEFALAS (Jodar, 2007).

In this work, we present an educational software to support the teaching of automata theory in CS graduate courses. We combine both educational and software development methodologies in order to design the application. We present two types of evaluations; one that compares the acceptance of the new software against JFLAP (although JFLAP was not developed under any educational methodology, it is in constant improvement). The second evaluation consist on measuring the level of learning over the first half of a course in two different groups of students. One of the groups did not use any supporting learning tool, whereas the other used the supporting learning tool presented in this work.

Raymundo Marcial-Romero J., A. Alvarez Contreras P., A. Montes Venegas H. and Hernández Servín J. (2009).
A SIMULATOR FOR TEACHING AUTOMATAS AND FORMAL LANGUAGES - FLyA.
In *Proceedings of the 11th International Conference on Enterprise Information Systems - Software Agents and Internet Computing*, pages 175-178
DOI: 10.5220/0002011601750178

The layout of the paper is as follows. In section 2 the methodological steps for the development are described. In section 3 two evaluations of the tool are presented. Finally the conclusions and further work are presented.

2 DEVELOPMENT OF FLyA

FLyA consists on three modules: the Finite Automata module (FA), the Context Free Grammar module (CFG) and the Turing Machine module (TM). The development of FLyA involved the combination of two methodologies, Cataldi et al. (Cataldi et al., 2003) and XP (Wells, 2006) which is developed based on prototypes. The application of the methodologies to the development of FLyA consisted of the following steps:

- **The Feasibility (FEA) Step.** In this step the need for a supporting tool to teach automata theory was established. The proportion of student failure at the University Autonomous of the State of Mexico (UAEM) was 45%. The costs calculated to build the tool were minimum including a MSc graduate student to implement the system and the University hosting resources were used.
- **The System Requirements Definition (SRD) Step.** During this step, several subtasks were defined: the requirement of graphical interfaces, the shape of the objects to be included and visualized in the application and the details of interaction between the user and the system. The minimum requirements for the use of the application (web navigator and the Flash-Player 7.0 plug-in or above) was also established.
- **The Requirement Specification of Prototype (RSP) Step.** A detailed verification of the requirements, the interfaces and the outputs of the application were considered in this step. Questions such as: is the graphical interface adequate for the kind of application?, is the interface intuitive enough for the end user? is the partial results shown by the application adequate? were answered at this step.
- **The Prototype Design (PD) Step.** In this stage of the software development, the design of the first prototype was developed taking into account the previous steps and identifying the main modules to be developed. The FA module consisted of four submodules, the GIC module consisted of seven submodules and the TM was developed in three submodules. The submodules design considered areas for: the input alphabet, the evaluation process, the intermediate results among others.
- **The Detailed Prototype Design (DPD) Step.** At this stage the data structures employed to implement the algorithms for each of the modules were established. The algorithms implemented were those mentioned in Hopfcroft et al. (Hopcroft et al., 2006) sections 2.3, 7.11, 7.13, 7.14 and 7.15 .
- **The Prototype Design Codification (PDC) Step.** During the PDC step, the first "functional" prototype was generated. The coding of the prototype was based on the extreme programming methodology (XP) (Wells, 2006) which has short deliveries in order to make all the necessary corrections previous to the next step. The prototype implementation of FLyA can be found at http://fi.uaemex.mx/aylf or http://zervero.prohosts.org/. There is no need to register in order to use FLyA, it is only required to go to the above address.
- **The Testing and Implementation of Prototype (TIP) Step.** Once the prototyped was concluded, two hosting accounts were opened, one at the University server and the other in an external server. According to the XP methodology short deliveries should be evaluated. There were two initial evaluations of the software, the first one carried out by the course's teachers and the second one by the students. A complete explanation is given at Section 3.
- **The Iterative Improvement of Prototype (IIP) Step** The IIP was based on the observations made by lecturers and students registered during the 2007B term for the module at the school of Engineering in the UAEM (http://www.uaemex.mx), who have made both perceptual and procedural observations.
- **Final Steps.** The final steps, namely, Final System Design (FSD), Final System Implemetaion (FSI), Maintenance and Operation (MO) and Retired (RET) are not discussed as the FLyA software is in constant improvement.

3 EVALUATION

In this section the evaluation of the educational application is discussed. In section 3.1 the acceptance by the students and the pedagogical considerations evaluated by the lecturers are presented. How the tool influence the teaching-learning process in different groups of students is discussed in section 3.2.

3.1 External Evaluation

Two instruments were designed at this stage. The first instrument was to evaluate the acceptance by students and the other to evaluate the pedagogical aspects of the software. The evaluation instruments are based on Cataldi (Cataldi et al., 2003) methodology.

The students evaluation instrument is mainly concern on three aspects: how the student perceive the application, how easy it is to use and how the procedures are described. Each question of the instrument had five possible options as answers. These are: very easy, easy, acceptable, complicated and very complicated.

The students evaluation instrument consisted of seventeen questions each one ranked according to Cataldi (Cataldi et al., 2003) methodology. The highest score been 63 points. This evaluation instrument was applied during three terms 2007B, 2008A and 2008B at the UAEM. During term 2007B a group of fourteen students were selected to evaluate both JFLAP and FLyA. The average score of JFLAP was 30 points compare to 45 point of FLyA. It is worth to mention that some of the main differences at the evaluation were the spanish version of FLyA (native languages of the students), the easy to install and use of FLyA and the minimum number of buttons and commands of FLyA compared to JFLAP.

At 2008A term, a groups of nine students evaluated the applications. This time FLyA had incorporated suggestions made by students in the first evaluation such as adding options for save, open, and print the work done. The average score of JFLAP was 31 points in this second evaluation in comparison to 47 points of FLyA.

During 2008B term, two different groups of students were considered. One group evaluated JFLAP and the other FLyA. This time the average score of JFLAP was 46.33 points compare to 50.3 points of FLyA (we note that the highest score was 63 points).

The second evaluation instrument was answered by nine lecturers, three of them teach at the UAEM and the rest at other universities. All of them evaluated both JFLAP and FLyA.

This evaluation instrument in comparison to students evaluation instrument, did not consider five options as possible answers, it only allowed three. The options were not the same for all the questions but generally speaking they consider the better, the middle and the worst case. The highest score was 95 points. FLyA had 72.5 points compared to 49.5 points obtained by JFLAP.

Marqués (Marqués, 2000) proposed a contextual evaluation which determines the characteristics of the software developed. A summary of the results of this evaluation to both JFLAP and FLyA are presented in table 1. Three are the possible scores for each question: high (totally recommended), medium (acceptable) and low (not recommended).

Table 1: Contextual evaluation of JFLAP and AyLF.

Evaluation aspect	**Score**	
	JFLAP	AyLF
Efficiency	high	high
Installation and use	medium	high
Changeability	low	low
hline Audiovisual quality	low	high
Content quality	low	medium
Originality	medium	high
Pedagogical perpective	low	high
Motivational	low	medium
Documentation	medium	medium

Table 2: Students marks at the AyLF course.

Student	**Written Mark**	**Practical Mark**	**Final mark**
1	44	50	94
2	21	50	71
3	8	45	53
4	19	50	69
5	50	40	90
6	36	40	76
7	25	45	70
8	25	30	55

3.2 Learning Evaluation

The learning evaluation was applied in two different Universities, one in a private university and the other in a public university.

In the first evaluation JFLAP and FLyA were introduced to a group of eight students. Each one of them had to choose among JFLAP and FLyA as its supporting tool during the course. After two weeks of evaluation, seven of the students decided to use FLyA. The course consisted on two hours of oral presentation and a set of practices realized in the supporting tool. The final marks at the end of the term (four working months) are presented in table 2. As it can be seen, 50% of the total marks corresponded to written exams and the other 50% corresponded to solving exercises and writing programs. It can also be seen that 25% of the students failed and 75% succeeded.

In the evaluation at the public university the students only used FLyA as their supporting tool. The

Table 3: Students marks during terms 2008A and 2008B where Std=Students, WM= written marks, PM=practical marks, FM=final mark.

Std	Semester 2008B			Semester 2008A		
	WM	**PM**	**FM**	**WM**	**PM**	**FM**
1	18.4	60	78.4	17.6	40.4	58
2	23.2	54	77.2	4.8	52.4	57.2
3	22.4	54	76.4	22.8	42	64.8
4	31.2	60	91.2	24.4	43.6	68
5	16.8	24	40.8	10	46.4	56.4
6	28	54	82	10.4	40.4	50.8
7	17.2	39.2	56.4	28	55.6	83.6
8	18.4	48	64.4	11.2	45.2	56.4
9	3.2	51.2	54.4	6.4	28.4	34.8
10	24.8	54	78.8	16	13.6	29.6

main idea in this evaluation was to compare the marks obtained by the students at 2008A term which did not use a supporting tool against the marks obtained by students at 2008B term who used FLyA as supporting tool. Both of the groups were taught by the same lecturer and evaluated using the same criteria.

Similar to the case at private university, the students at term 2008B had to solved a series of exercises using FLyA. Columns five to seven of Table 3 show the marks of ten students during 2008A term. In this case 60% of the total marks correspond to written exams and 40% to exercises and programs. Columns two to four in table 3 show the results of ten students during 2008B term. As previously mentioned, the evaluation criteria was equivalent in both terms, e.g. same number of exercises, same number of exams and same number of programs. An easy calculation shows that there was a 21.4% increase in the success of student with the use of FLyA as a supporting tool.

We believe that the design of the tool contributed to the results obtained since the interface makes use of a minimum number of controls (steps FEA, SRD, RSP). Hence, the user need not to worry about learning the application, it only focuses its attention on the steps to obtain the results.

4 CONCLUSIONS

Finally, it can be said that FLyA is a tool in development, with user acceptation. It is worthy to say that when a tool deliveries correct results besides of being accepted by the community, it is without a doubt a tool with growing possibilities. This application together with the results are product of a methodology for the development of educative software. There is a special emphasis on the aceptation by of the user. As pointed out by Marqués (Marqués, 2000), it must be facilitated the use of educative applications and to avoid installation and use of additional software. It is highlighted that the user must not be saturated with controls or menus on the display. This makes feels the user impressed over the magnitude of the tool, leading to the rejection of the application.

REFERENCES

Bovet, J. (2006). Visual automata simulator, aplicación desarrollada en java. *Online:www.cs.usfca.edu/ jbovet*.

Budowski, Y. (2001). Dfa simulation, aplicación desarrollada en visual basic. *Online:www.vb-helper.com/contest_dfa.html*.

Burch, C. (2001). Automaton simulator,aplicación desarrollada en java. *Online:www.cburch.com/proj/autosim/*.

Cataldi, Z., Lange, F., Pessacq, R., and Garcia, R. (2003). Metodología extendida para la creación de software educativo desde una visión integradora. *Revista Latinoamericana de Tecnología Educativa*, 2(1).

Charras, C. and Lecrog, T. (1997). Research with an automaton, a java aplication. *Online:www-igm.univ-mlv.fr/ lecroq/string/node4.html*.

Head, E. (1997). Finite state machine (fsm) simulator. *Online:http://www.cs.binghamton.edu/ software/fsm/fsmdoc.html#grader*.

Hopcroft, J., Motwani, R., and Ullman, J. (2006). *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley.

Jodar, J. (2007). Software para la enseñanza de las fases de análisis léxico y análisis sintáctico (sefalas), aplicación desarrollada en java. *Online:http://lsi.ugr.es/ pl/software.php*.

Marqués, P. (1995). Metodología para la elaboración de software educativo. *Online:http://www.blues.uab.es/home/material/programes/t023151/uabdisof.htm* .

Marqués, P. (2000). Diseño y evaluación de programas educativos. *Online:http://www.xtec.es/ pmarques/edusoft.htm*.

Rodger, S. (2006). Jflap, aplicación desarrollada en java. *Online:http://www.jflap.org/*.

Wells, D. (2006). Extreme programming. *Online:http://www.extremeprogramming.org/*.