CAAD Program Development:

Expectations and Results

CINELIS Gintaris, JANUSKEVICIUS Eugenijus, KAZAKEVICIUTE Geruta

Vilnius Gediminas Technical University, Lithuania
http://www.vtu.lt

This research and the paper is our attempt to study in what extent is it reasonable to fill the curriculum provided for architecture students educated as a contemporary architect with knowledge of algorithms, programming, integration of CAAD tools, etc. The experience of several years of the course "CAAD program development" was generalized and will be discussed taking into account the results of the analysis of the feedback from the students. The results of the work could be important for the definition of the guidelines for the future.

***Keywords**: Programming and architectural education, formalization of design problems, CAAD program development*

Introduction

We consider that no one CAD/CAAD system available on the market "as it is" is in general enough to realize different design strategies. One can adjust the industrial system to his design style and raise the productivity and the quality of design solutions significantly by flexible use of the features of open architecture CAD systems.

Context

Programming architects were met very seldom ten years ago but nowadays relatively large number of them are able to create the program for their needs. Some questions still arise among architects and designers in that context today. Principal question is: are customization tools of CAD of open architecture really addressed directly to a designer or are they supposed to be used by pure IT specialist? By providing this course we consider the importance of that kind of knowledge for an architect.

In general we can speak about two well-known ultimate viewpoints:

Programming technology will never become a real practical tool in the hands of the practicing architect. It is too complicated, time consuming on the stage of programming, requires too much special knowledge of computing science, the architectural design as a process is immune to formal methods and relies mostly on intuitive way of creating (pessimistic point of view);

Architectural design is based on logical thinking and functional schemes and relations. Because the nature of architecture can be rationally explained it can be interpreted as pure mathematics and formally defined almost in full extent (optimistic point of view).

Most of us admit a kind of combination of these two viewpoints when we borrow something from each of them relying in some degree on computational theories, formal rules and leaving a space for individuality (case based reasoning, prototype designing, form follows function). That issue in our opinion is important for the course in question because the general premise reflects on the content, the methods and style of teaching.

Methodology

An important question is: what is possible to learn in programming keeping in mind that master degree students already realize real architectural designing problems but, as our questionnaire says, have no or little experience in program designing and development.

In general we think about the process of learning of the formal programming language as learning of a natural language. The nature of natural and formal languages has some common features. The studying of natural language is a complicated process when one starts to use new but simple sentences, then gets some automatic skills of speaking, later he is able to comprehend complex abstract concepts of the language. The same it seems the obvious way we try to go while studying formal programming languages: make first steps using basic elements and very simple examples, then acquire some habits of programming, using the same small examples, and later use concepts and more complex structures of the language. Small examples and exercises are big psychological support in the beginning of the course because people realize that sometimes they need a few knowledge to achieve useful results. However the paradox and the fault of that common methodology is to our opinion that simplicity of examples where the difficulties are intentionally eliminated stops the positive evolving of the programming ability. Small and partially prepared exercises often create the illusion for the student that he will move smoothly further and next, more complicated exercises will be as successful as the first ones. The problem becomes clear when one works on a bigger and more complicated project.

The problem is it is not possible and maybe harmful to avoid big jumps in complexity of the tasks.

It is important to realize and to define common objectives of such course what in fact relates to its content. Is it main task to automate the most routine procedures of the design or is it more important to translate the creative tasks into formal language? The freedom to choose is actually predefined by the programming ability and what we called automatic skills of the students when they start from very beginning. That limitation is very clear (one semester) in practice. Though we consider the terms “routine” and “creative” are rather conventional in that context: designing of the program of every routine forces the person to analyse the structure and sequence of its process, represent it in a more abstract way, create the algorithm that he/she realizes it is the most suitable one. In that way the architect becomes more aware of the process of design. Representing personal idea on the computer also imposes certain discipline on the designer who acts as a programmer. That process resembles real designing with various constraints. While developing his ability of abstraction and representation using formal statements of CAD interface language, algorithmization skills a student can try to structure his design logic of more creative problems. And of course we always can expect the effect when one’s ability to pass the everyday routine to the program liberates him for more creative problems.

Course

“CAAD program development” course runs one semester for the first year master degree students. Although the emphasis of the course is on principles and it is not intended to train the students in a particular CAD program, basic CAD environment for the course we have chosen is AutoCAD and main programming language is AutoLISP. Some additional CAD customization tools and features, like script, menu, DCL programming are also integrated.

Time budget includes 1 academical hour theoretical lectures and 2 or 3 hours practical classes a week. The assignments for students include ten exercises and a final work.

At appropriate moment of the course the stu-

dents usually are asked to choose their level: common level or higher level. Ordinary assignments are quite exactly defined (well structured) for both levels. The assignment of the final work differs: for higher level students more interesting assignments are prepared together and in discussion with the teacher. In both cases the students are encouraged to solve the problem in most original way with the emphasis to personal concept. It could be interpreted like the piece of music where the theme is suggested and improvisation follows.

Research

The way we used for investigating the problem is the self-evaluating of the students. In order to check and generalize the students' opinion about their personal computer and programming knowledge and skills we have prepared a special questionnaire.

The questionnaire included the questions about the initial knowledge of computers, various CAD systems and programming, a particular question of self-evaluation as an AutoCAD user, the questions about ability of abstract description and algorithmization of design tasks in general and particularly using AutoLISP, the questions about the most difficult parts of the course, extent and significance of the course, future intentions to use programming in architectural activity and study it further independently. Prepared answers as a rule with a growing evaluation level from 'no idea' to 'excellent' were attached to each question.

To our opinion it was logical to expect the relation between the general computer knowledge and skills, level of knowledge of standard features of the CAD system, evaluation of the extent, the need, and the perspective of the programming course on one hand and the people's program development ability on the other hand. The survey shows, that students in general are able to work with computer, most of them have some practice working with industrial CAD systems. Some of them have also some practice in programming and some of them used earlier even more than one programming language. The preliminary visual analysis gives an impression of dispersion of the opinions in rather broad range. The quantitative analysis allowed us making stricter conclusions.

Results

Generalization of the topic can be made in two ways: as the remarks made on the basis of the impressions of teachers and as the conclusions made on the basis of the feedback from the students.

Inspite of popularity of the course in general, enthusiasm of some students, constantly increasing user friendliness of programming systems (for example Visual LISP Integrated Development Extension) some frustrations and hesitations can be expressed.

In general we can say the students personally succeed during the course very differently. The success of the students solving practical tasks depends also on their knowledge of the basic graphical editor. For those of them having no good exposure in common CAD environment it is a bigger problem to work practically on that basis. Sometimes the people try to create the programs that cover the functionality of basic CAD commands.

Often it is not enough to know and be able to program using one programming language (f. e. AutoLISP) when speaking about convenient solutions of real designing tasks. Other CAD interface tools (different types of menu, DCL language, Visual Basic, system variables, integration tools, etc.) need appropriate additional knowledge about their semantics and syntax and also experience when using them in practice. In addition it is not always as friendly as we expect. These reasons cause the difficulties for young program developers.

It is supposed that the knowledge achieved could be extended and generalized for various CAD

software environments and play important role of common CAAD education. Still we have to admit it is only partially true mainly in the part of the methodology.

Mostly the people agree that there is no industrial CAD system that is both universal and suits for personal needs the same time. But sometimes it is argued that the designer is able to adapt himself to the system's standard functionality and ignore "a little" loss of productivity. The other argument that relates same aspect is that the architect can address with his special need to a professional programmer. The last statement is controversial because only appropriate informat

ics knowledge enables the architect to communicate with IT specialist the same language.

In general, students have too high opinion about their skills in computers and CAD systems. More than 90% of students evaluate their knowledge of computers and AutoCAD as 'satisfactory' and 'good'. But when they are asked about their programming skills, which are directly related to knowledge of computers, less than 30% of them have skills in high-level programming languages, like C, C++ or Pascal. Predominant answer is 'embedded languages, ' like VisualBasic, AutoLISP, TeX/LaTeX. This is not bad, as any knowledge of any programming language will help the student to start the course. Just when we speak about more complicated tasks, knowledge of object-oriented or even high-level programming language will help student more than just knowledge of programming language.

Things are changing when we come to the questions about their programming abilities, like formalization of the task, finding a suitable algorithm and writing the code: 55-75% of these answers are marked as 'bad' and no one of them has chosen the answer 'good' or 'excellent. ' We can see the same relation when students are asked about the most difficult part in the course: predominant answer is that the most difficult part in the course is modular design of the program. This is directly related to their knowledge of high-level programming language.

These correlations point out a question: who must teach students to think in abstract terms? One of the possible solutions could be introducing general course of goal-seeking strategies as well as logic to the curriculum. This should be done not only for the sake of the course of 'CAAD program development, ' but for the sake of the student to help him to find solutions of the problems he/she will encounter in his practical work as an architect.

Acknowledgements

Thanks to our colleague Lionginas Ciupaila for his survey preparation work.

References

Eastman, Charles M.: 1992. Use of data modeling in the conceptual structuring of design problems. In Computer Aided Architectural Design Futures: Education, Research. Applications, ed. Gerhard N. Schmitt, 225-243. Braunschweig/Wiesbaden: Friedr. Vieweg & Sohn Verlagsgesellschaft mbH.

Madrazo, Leandro: 1993, Keywords: Notes on Form, Space and Architecture. ETH Zurich.

Mitchell, William: 1990, The Logic of Architecture: Design, Computation, and Cognition. MIT Press, Cambridge, Massachusetts.

Mitchell, William, Robin S. Liggett, and Thomas Kvan. 1987, The Art of Computer Graphics Programming: a Structured Introduction for Architects and Designers. Van Nostrand Reinhold, New York.

Schmitt G., Dave B., Kurmann D., Refvem S., Shih S.: 1993, CAAD Programmentwicklung, Wintersemester 93/94, ETH Zurich.

Schmitt, Gerhard. 1988, Micro Computer Aided Design for Architects and Designers. John Wiley & Sons Inc., New York.