How to Teach Architects (Computer) Programming

A Case Study

Gabriel Wurzer[1], Sema Alaçam[2], Wolfgang Lorenz[3]
[1,3]Vienna University of Technology [2]Istanbul Technical University
[1,3]http://www.iemar.tuwien.ac.at [2]http://www.mimarliktabilisim.itu.edu.tr/main.aspx
[1,3]{wurzer|lorenz}@iemar.tuwien.ac.at [2]semosphere@gmail.com

Abstract. *Computer programming in architecture seems to be commonplace throughout the eCAADe Community. Yet, a critical evaluation of a programming course as seen from a student's side is still missing. During a week-long programming workshop in a fellow university, we have been assessing subjective parameters such as mood, quality of presentation and comprehensibility, comparing these to the actual topics that were covered at this instance. Our results contribute to understanding architecture students in their quest towards algorithmical thinking. We are convinced that the discussion given in this paper will help other teachers to further increase the quality of their lectures. Furthermore, the structure of our approach may serve as basis for further research into recording student behavior during programming courses.*
Keywords. *Teaching, Programming, Assessment.*

INTRODUCTION

In recent years, the question of how to bring programming to architecture students has often led to uncertainty (Leitão, Cabecinhas and Martins, 2010, Bourdakis, 2010, Boeykens and Neuckermans, 2009, Burry, Datta and Anson, 2000, Fricker, Wartmann and Hovestadt, 2008): As advocates of functional programming rally with those who prefer imperative programming, people who prefer one programming language with those that prefer another, and everyone is at odds with oneself on how to meet the goals being set forth for their course, there seems to be a strong wish for sharing insights on "how to teach architecture students programming", which has led to a variety of publications on the topic. One thing largely missing from the discussion is, however, a fresh look at the subject from a student's side. In the course of a programming workshop, we have therefore been assessing the subjective impressions of students, and have compared these to the actual presented content. Being a case study, we see our contributions both in the conducted work (see "Case Study") as well as the discussion of the results as actual consideration points which we believe can help fellow lecturers improve their courses (see "Discussion"). Given the perspective that we would like more teachers to conduct student-centered assessments, we also bring forward a format which we believe can serve as practical assessment basis, in the form of questionnaires to be handed to the students (keyed by us as mood charts).

RELATED WORK

During the last decade, it has been discussed how far programming courses support the architecture students' way of thinking. Researchers state that it improves understanding of descriptive geometry when it comes to representations and relations between the entities (Bourdakis, 2010, Burry, Datta and Anson, 2000, LaBelle, Nembrini and Huang, 2009, Kajima and Panagiotis, 2008). Typically, programming in the architecture domain is performed by writing code (e.g. Visual Basic for Applications, ANAR+ for processing). However, visual programming approaches that promise to fit the graphical mindset of aspiring architects better are also gaining ground, e.g. Bentley's Generative Component and Grasshopper for McNeel Rhino 3D. The reason for this is reported to be that algorithmical thinking requires a problem-centered approach, while architecture usually takes a solution-centered strategy (Lawson, 2005).

In our case study, we focus on programming by writing code, targeting the question of teaching quality as seen by architecture students in a workshop environment, using a questionnaire-based approach as method. To the best of our knowledge, this has so far not been conducted, although we can find examples of qualitative thoughts for example in (Knight and Brown, 2010). In detail, we will focus on the different subjects to be covered in such a course, relating the sequence given and the time spent on each subject to the comprehension and perceived quality of presentation. It is safe to say that our report is not generalizeable (given the number of participating students as well as the form of survey); we see our contribution as a pre-study, on which further work can be based. Therefore, we also cover the shortcomings and address fields where more effort should be invested when performing a full-fledged study (see "Case Study").

WORKSHOP DESCRIPTION

The programming workshop was given in November 2010 at the architecture faculty of Istanbul Technical University over the duration of five days. The content of the course was on programming using Visual Basic for Applications in AutoCAD for the sake of form generation, i.e.:

- To learn basic constructs of an imperative programming language and use them to generate geometry.
- To develop algorithmical thinking by solving a programming assignment in a studio-like context, in order to learn how to generate form and automate daily CAAD work routines.

There were 18 students attending, half of which already had or were in the process of being in a basic course on PASCAL. As lecture material, a specially-produced booklet covering all presented topics was given to the students, allowing them to revise the subjects taught and catch up once needing to be absent for some hours. Furthermore, a description of the programming assignment was handed out. The chosen topic was on automated building using a grid-based house-generator, which would call up each student group's program in order to build a house.

Due to the constrained nature of our schedule, it was decided to give the introduction to programming in only two days, after which practical work on the programming assignment would begin. For a programming lecture, this rather unusual approach boils down to imposing a very intensive work program in a short time frame, and clarifying topics still unclear to a student in a face-to-face manner, in parallel to actual work being already done by all others.

The topics covered were the same ones that are usually given in a standard programming lecture. However, to meet the time constraints, care was taken to bring two or more topics at the same time (e.g. learning the basic program structure while getting introduced to debugging):

- Debugging a "Hello World" program, variables (only numbers at first), functions with and without return value, parameters, calculations with numbers, nesting of calls
- AutoCAD drawing library (using Object-Oriented Programming without actually saying so), Booleans and conditions, arrays (static and dynamic)

- Implementation of a staircase generation program using loops, creation of a user interface to enter parameters for the staircase program, strings and string operations, introduction to Object-Oriented Programming

The final goal and motivation of the workshop was to present the results of the programming assignment to the dean of Istanbul Technical University's architecture faculty, as well as to conduct the case study on perceived quality of each topic given.

CASE STUDY

The survey on teaching perception was conducted using a questionnaire-based approach. At the beginning of each day, the students were given a sheet in which they could chart general mood, comprehension of the presented material and perceived quality of presentation over time, using a five-fold scale (excellent, good, fair, satisfactory, poor). In order to emphasize deviations from the average case ('fair'). The survey sheets, also keyed as mood charts, were completely anonymous - participation was conducted on a voluntary basis. Students were told to sample at the end of each topic, and encouraged to take additional samples as they saw fit. Furthermore, the noting of sentiments on the graph was explicitly welcomed (e.g. "11:15 - boring!"), although very seldom seen. At the end of the day, mood charts could be thrown into a bin (filled or unfilled). The results were then analyzed and interpreted by correlating the presented topics (of which time and duration was known) to the mood, quality of presentation and comprehension at the given instance (see Figure 1).

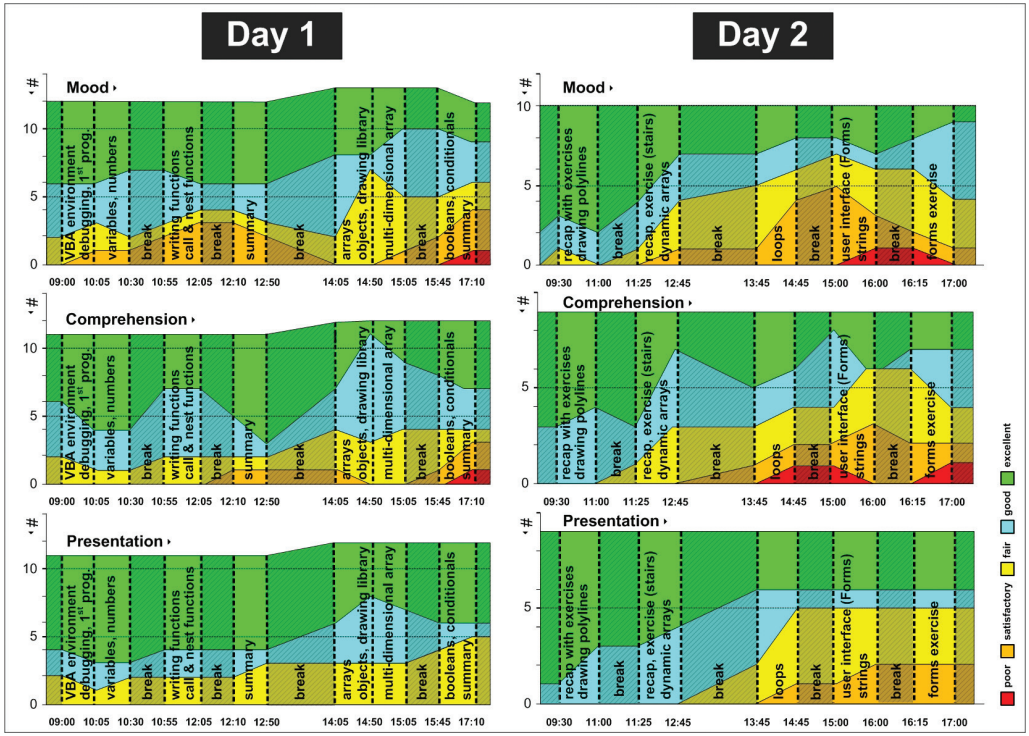

Figure 1
Analysis of „mood charts" created during the introductory programming course. Mood, comprehension and quality of presentation are charted over time. The covered topics are furthermore entered at the time of their appearance. The results clearly show a change in all supplied measurements from very excellent marks in the morning (color-coded as green) to poor marks in the afternoon (red).

For the three days of programming that followed the two-day introductory course, only mood was measured at discrete times (morning, midday, evening). The reason behind this was that, since all students had to work individually, there was no presentation given and therefore also no comprehension to measure. The assignment itself was presented at the start of day three, implemented on day 4 and the outcomes presented on day 5.

Observed attitude towards the presented material during the workshop

The first half of day 1 was occupied with "diving in" and getting fully focused on programming. Presentation was seen excellent, comprehension was generally good. However, as can be seen from the mood chart, the rapid flow of information lead to a steady decline in mood, from good (morning) to fair (midday). We see this as being related to the presentation of functions and parameters, which was given using the concept of "Input-Process-Output" (IPO) and then mapped to either function (with output into a variable) or procedure (without output). As mental bridge, a relation to visual programming using flowcharts and Rhino Grasshopper was given, which worked remarkably well. The relation to mathematics (e.g. function sin(x)), however, did not have a beneficial contribution to the understanding.

After lunch break, the mood returned to normal values. The presentation of arrays was hard to digest, it seems, and thus led to a dramatic decrease in comprehension and to a decline of mood in the afternoon. As soon as practical work with arrays using the AutoCAD drawing library was done,

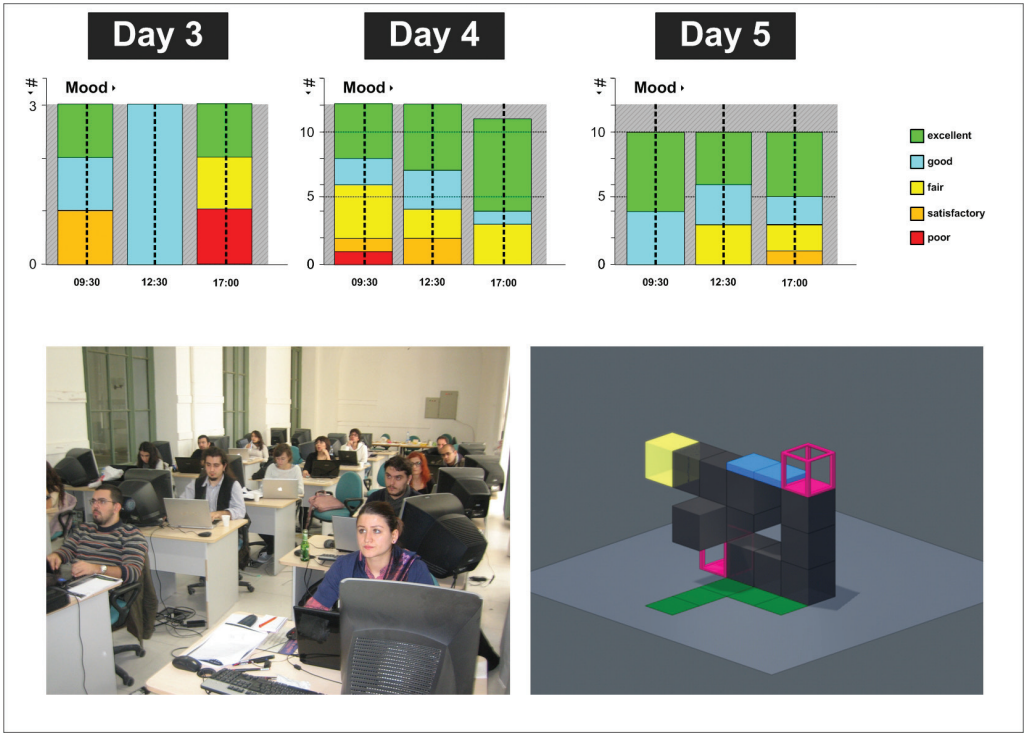

Figure 2
Upper part: Mood during the programming assignment. Even though the point samples at morning, midday and evening are of limited significance, one can see that the trend in Day 4 is significantly different than any of the other days in the sense that the mood constantly gets better. Lower part: (left) Photo taken during workshop. (right) Example of a grid-based house generation algorithm used in the assignment.

the mood settled on an average level. The introduction of multi-dimensional arrays which followed immediately afterwards without giving practical examples was seen as inadequate, with mood and comprehension decreasing to poor at around 4pm. The last lecturing hour, in which Booleans and conditions were to be introduced, was not digestible any more. Therefore, only an overview followed by the day's summary was given, with the intention of deepening the knowledge in a recapitulation in the beginning of the next day.

Day 2 started with practical exercises (drawing geometry), in which conditions and Booleans were brought in. This approach worked remarkably well, with all measured parameters being excellent until midday. The introduction of dynamic (i.e. growable) arrays led to a mood shift at around 12am, with the presented content being generally understood, but not well-received due to the lack of benefits for the program written. After lunch break, the introduction of loops required the re-writing of a previously written program and led to a drop in all sampled values. Either the adaptation of a program was not sufficiently presented, or the introduction of new content was not adequate. Between 3pm and 4pm, an overview over Forms and Strings was given. Even though this subject was, in our view, already adapted to the declining attention and mood, it was nevertheless not well received, with mood values being at their minimal peak of the day. This afternoon decline leading to a no-go situation at 4pm is to be investigated in further courses. After a 15-minutes break, a practical example on forms as well as the end of the day being in close reach, the mood values increased again.

In day 3, we did not gather enough measurements in order to be able to discuss them here. From our observations, we saw a generally bad mood, connected to the rapid pace of the previous days, which we had to amend by giving individual help to the students, leading to a significant improvement over day 4. In day 5, we observed a decline in mood due to the hand-in stress.

Lessons learned

From our observations during the workshop, there are several points that are worth consideration in subsequent courses:

- The explanation of functions with and without parameters, with and without return value benefits from a clear analogy to an IPO metaphor, if which grasshopper is an excellent visual example.
- Simple data types (strings, Booleans, numbers) can be handled in a straightforward manner. Arrays, however, are hard to explain - especially in the multidimensional case. It is probably hard to imagine the necessity for collections of values, which would otherwise have been taught in a course dealing with algorithms and data structures (definitely also a point for further emphasis in a programming lecture for architects). Objects are naturally observed as "packages of functions and variables", with no further explanation neended. This might be as to the nature of VBA, which exposes all functionality of AutoCAD as objects.
- As opposed to a bottom-up lesson, which tries to present all topics independently, we have experienced that is does sometimes make sense to present a seemingly complex topic right away and then decompose it into its constituent parts. In this way we were able to present many concepts in parallel, without losing reference to the overall picture.

Comparing to a full-fledged study

We are aware that the number of participants in our study is statistically insignificant. The results nevertheless reflect, to a large extent, out own insights gained through years-long teaching practice, and can be used as a guidance when setting up a larger study. Points worth considering when doing so are:

- The prior knowledge of the students was not asked in a structured manner (rather by raising hands). It might be important to sample, for example, who has had experience in each of the presented topics

before actually beginning the course.

- For comparison reasons, comparison groups will have to be set up.
- A fivefold scale offers the opportunity to state average levels. It might be interesting to exclude this possibility, thus forcing the student to be explicit.
- The assumption that each topic can be sampled immediately afterwards (i.e. the topic as cause, the comprehension as effect) is naïve. It might be interesting to see when each topic has settled (using questions targeted not only at the last topic given, but at a variety of topics).
- As side note, we must mention that the students were of different interests and knowledge concerning programming, which led to segregation. It might be easier to hold a workshop in an environment where all students have the same knowledge (e.g. by the way of establishing programming in a fixed place in the curriculum).

DISCUSSION

During the teaching process of computer programming in architectural design education, the feedback from the students, their motivations and their learning curves should be taken into consideration by the lecturers. There should be more explorative studies about what kind of strategies in teaching computer programming are crucial for introductory courses.

From our collected data, we can see a direct connection between mood and comprehension, while the quality of presentation is separate and declines slightly from the morning to the evening. Topics that are new or hard to digest must be therefore brought in the morning. If this is not possible, more time than usual has to be invested in the afternoons. In general, recapitulations and exercises are of utmost importance, due to the nature of architectural work (solution-centered). However, the format of a workshop might not always allow for such deepening of knowledge, because of time constraints. Therefore, establishing an educational scheme that covers all programming topics while at the same time being time-efficient and qualitatively useful as seen from a student's side is a field worth researching in.

REFERENCES

Boeykens, S and Neuckermans, H 2009 'Visual Programming in Architecture: Should Architects Be Trained As Programmers?', *Proceedings of CAADFutures 2009 CD-Rom*, Montréal, Canada.

Bourdakis, V 2010 'Designing Interactions: A step forward from time based media and synthetic space design in architectural education', *Proceedings of eCAADe 2010,* Zurich, Switzerland, pp. 151-156.

Burry, M, Datta, S and Anson, S 2000 'Introductory Computer Programming as a Means for Extending Spatial and Temporal Understanding', *Proceedings of ACADIA 2000*, Washington D.C., United States, pp. 76-86.

Fricker, P, Wartmann, C and Hovestadt, L 2008 'Processing: Programming Instead of Drawing', *Proceedings of eCAADe 2008*, Antwerp, Belgium, pp. 525-530.

Kaijima, S and Panagiotis, M 2008 'Simplexity, the programming craft and architecture production', P*roceedings of the First International Conference on Critical Digital,* Cambridge, United States, pp. 181-194.

Knight, M and Brown, A 2010, 'Increasing Design Reflection and Improving Feedback using Wikis', *Proceedings of eCAADe 2010*, Zurich, Switzerland, pp.51-55.

Labelle, G, Nembrini, J and Huang, J 2009 'Programming framework for architectural design ANAR+: Object oriented geometry', *Proceedings of CAADFutures 2009*, Montréal, Canada, pp. 771- 785.

Lawson, B 2005, *How designers think: the design process demystified*, Architectural Press, Burlington, p. 43.

Leitão, A, Cabecinhas, F and Martins, S 2010 'Revisiting the Architecture Curriculum: The programming perspective', *Proceedings of eCAADe 2010*, Zurich, Switzerland, pp. 81-88.