Experience Report of a Software Development Course in a Faculty of Fine Arts

Luis Corral
School of Information Technology and Electronics
lrcorralv@itesm.mx

Monterrey Institute of Technology and Higher Education

Abstract

This paper describes the implementation of Computational Thinking techniques to promote the development of software development skills in Fine Arts postgraduate students, understanding that the population does not have formal academic training in Computer Science. We present a case study observed during a software development course taught to a non-expert population of Fine Arts postgraduate students. As a result of the implementation of Computational Thinking concepts, students executed software solutions applicable to real-world problems, and Computational Thinking competencies were observed through characteristics of concepts, practices and perspectives. This paper discusses as well situations that may set basis for strategies of assessment or evaluation of Computational Thinking principles.

1 Introduction

Traditional education in Computer Science enables students to become professionals able to apply computer knowledge in problems that occur in everyday life, thus supporting or serving several other disciplines. As Computer Science students acquire a core skillset, it is expected that they develop a sufficient command on [Perkovic 2010]:

In: A. Piotrkowicz, R. Dent-Spargo, S. Dennerlein, I. Koren, P. Antoniou, P. Bailey, T. Treasure-Jones, I. Fronza, C. Pahl (eds.): Joint Proceedings of the CC-TEL 2018 and TACKLE 2018 Workshops, co-located with 13th European Conference on Technology Enhanced Learning (EC-TEL 2018), 03-09-2018, published at http://ceur-ws.org

1. **Computer literacy**: the ability to use basic computer applications, for example, an operating system, a word processor, or a web browser;
2. **Computational fluency**: consists of having a high level of understanding of the functioning of a complete computer system;
3. **Computational thinking**: refers to the ability to apply computer techniques to problems and projects in all areas, for example multiple aspects of science, arts and humanities [Wing 2006].

The current professional market makes this taxonomy very relevant: concepts like *digital transformation*, *Industry 4.0*, *cyber-physical systems* and others make very relevant having a good command of Computer Science concepts, even for non computer scientists. In this front, Computational Thinking lays the foundation for incorporating software development skills in a wider span of profiles. It is not uncommon that professionals in certain domain of a business, market, or science, require to expand their knowledge to gain basic or complex skills on software development. The reason behind this is a strong push of the market to make professionals to participate in the development of software tools of different complexities that enable the daily execution of their jobs, regardless of the discipline or context of application (for example a macro, an automation script, or a web page). A software suite like Microsoft Office offers to the user the capacity of automating repetitive tasks using *macros* that can be built without coding. For users who need more complex features, it is possible to implement simple code using Visual Basic. This illustrates how common products enable users to start developing basic software pieces, that are created without strong basis on programming languages or software development.

In a traditional perspective, the incorporation of Computer Science topics in non-Computer Science curriculum has been directed mainly to *computer literacy*, being the development of operational capacities one of the most common learning objectives (that is, the ability to *use or operate* a computer application or package) [Bizzarri 2011]. However, this does not necessarily mean that having a good level of computer literacy will lead to having a good command of the underlying principles of Computer Science. This becomes particularly important if we consider that those principles are necessary for profiles who will develop software as part of their professional jobs even though they are not computer scientists or trained software developers.

This paper describes the implementation of Computational Thinking techniques to promote the development of software development skills in a group of postgraduate students of a Fine Arts Faculty, understanding that the target population has no previous experience nor formal training in Computer Science. Moreover, the paper discusses situations that may set basis for strategies of assessment or evaluation of Computational Thinking principles.

2 State of the Art

According to the definition of Computational Thinking (CT), it *"involves solving problems, designing systems, and understanding human behavior, making use of the fundamental concepts of Computer Science"* [Wing 2006]. From this viewpoint, CT can be understood as applying scientific-computer thinking when facing and solving a problem. This capacity should be made available not only in higher education programs in Computing, but in other higher education programs and even in basic education programs. In consequence, the need to teach this skillset to students of all educational levels becomes relevant. To address the issue, research efforts have focused on the definition of curricula that include teaching these skills; nevertheless, there is still room to continue deepening in the topic, and carrying out empirical research [Grover 2013].

The development of CT skills also means the improvement of certain competences, which are usually organized by a progression table, which includes, among others, collecting, analyzing and representing data, decomposing and abstracting problems, systematizing, automating and simulating solutions. Previous work in CT has focused on issues of definition of the concept and the tools that foster CT. Repenning [Repenning 2010] lists a series of conditions that a computerized tool must meet for the systemic impact: low learning threshold, allow prototype development, facilitate the transfer of knowledge, be systemic and sustainable. On the other hand, it is recommended the balance and universality in the previous training of the population that implements CT [Cooper 2010].

In a practical approach, the question about the teaching of computer skills to non-computer scientists has caught the attention of the academic community over the time, from the 80s to the present. However, during recent years, the current perspective has focused on CT as an efficient strategy to accomplish the mission. This approach has attracted the attention of a broad academic community, and several scientific articles have tried to capture the essence of CT and its field of application [Bloss 2001, Walker 2010]. Previously, it has been discussed and researched how CT helps in understanding the capabilities of computer science applied in other areas. For example, the applicability of CT in other fields has been studied, including Medicine [Gong 2011], Astronomy [Gray 2010], Archeology [Troccoli 2005], Journalism [Corral 2010], Political Science [Conitzer 2007], etc. In this paper, we extend the experience previously reported incorporating an additional domain: Fine Arts.

3 Objectives

The goal of the implementation of CT as a learning strategy is to improve the ability of students to conceptualize, understand and use information technology in different fields of application. Likewise, the presence of study programs that involve software development in non-specialized schools and faculties, as well as graduation profiles, include the training of professionals capable of interacting with others in order to find ideas to solve problems, and imperatively requires the implementation and systematization of a computational strategy.

Research Goal

Understanding if through Computational Thinking, didactic and methodological tools can be created to explain computer concepts in a way that facilitates students with non-software profiles to create computational solutions applicable to real-world problems.

4 Research Setting

A work setting was developed in the form of software development training courses in a segmented non-software population. A group of five graduate Fine Arts students was observed. The students came from different undergraduate profiles, mainly graphic design and other visual arts, pursuing a graduate degree on Web Design in a Mexican state-funded, public University. The profile of this population is approximately 30 years old, with a university degree, preferably in artistic or creative careers, which allows observing students with relatively limited knowledge in software development.

The graduate program they course is a professional degree designed to train specialists in Web design, providing our students with theoretical and methodological elements to solve needs through technological innovation in the field of *digital communication* in *Web environments*. This mesh of disciplines requires starting from strong basis on visual communication and design, but transcend those skills into abilities to design and implement the software product (that is, a web system, web site or web app). This provides an ideal working setting to implement CT techniques and evaluate outcome products. Students are expected to expose themselves to sort out a real-world challenge from an industrial setting, which can be solved in the form of a software product. Hence, students shall understand the problem, abstract it, propose a flow of execution of a solution, and leverage the different information sources.

Our research setting is a course called *Dynamic Web Development*. The course spans in one semester, four hours a week through about 16 weeks of coursework. The teaching methodology was frontal lessons with laboratories to practice the acquired knowledge. The period assigned to laboratories comprised half of the workload in the course.

To guarantee the development of CT skills, the course aims to develop certain competences that includes, among others, collecting, analyzing and representing data, decomposing and abstracting problems, systematizing, automating and simulating solutions. In addition to these competences, there are also three fundamental dimensions: (1) **computational concepts** (sequences, cycles, events, parallelism, conditions, operators and data), (2) **computational practices** (incremental and iterative development, testing, reuse, modularization), and (3) **computational perspectives** (expression, connection and questioning) [Brennan 2012].

The range of computational tools to be used is segmented to the typical technological stack of the web environment: applications developed in HTML, CSS and JavaScript, adding complementary instruments such as jQuery, Ajax and AngularJS. The final product shall be a fully functional web application. The software systems developed are part of an industrial domain selected by the student, under the supervision of the course advisor.

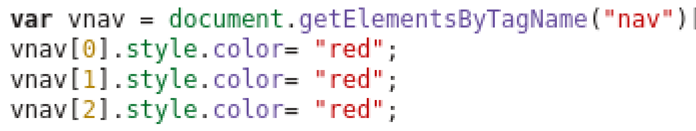
5 Evaluation

Considering the age range and previous knowledge of the students, they were first asked for a paper and pencil design with a graphic sequence of their computer system. This allows conceptualizing and sequencing in a dimension that is familiar to the student making use of creative and plastic skills and then moving to software level implementation. Considering the framework proposed in [Brennan 2012], the practice allows to sequence, express, connect and question. Nonetheless, the implementation in HTML has as a consequence that there are no graphical interface tools to offer assistance when it comes to actual software programming: the aid offered by ”what you see is what you get” (WYSIWYG) editors like Dreamweaver fall short to assist the integration of control structures such as cycles, events or repetitions in JavaScript. Student face a situation of more independence to choose software development techniques, and as a result they encounter more implementation problems.

As evaluation technique for the outcome products (namely homeworks and final projects) we proposed a two-fold strategy that includes:

- **Automatic code review**: implementing code inspection and analysis using an automatic tool (`http://jshint.com/`) looking for complexity and eventual code errors;
- **Visual code scrutiny**: implementing a visual code inspection, looking for optimization opportunities and implementation vices.

Performing automatic analysis on about 15 homework assignments, it is uncommon to detect code that exceeds a McCabe cyclomatic complexity number greater than 2. Performing visual code scrutiny, it is often found errors where the student expresses a solution in a strictly sequential manner without discovering which pieces of code can be reused or associated with events. For instance, in Figure 1, we can observe a deficiency of implementation, where the student repeats three times an instruction that could be expressed in a single line of code.



```
var vnav = document.getElementsByTagName("nav")
vnav[0].style.color= "red";
vnav[1].style.color= "red";
vnav[2].style.color= "red";
```

Figure 1: Deficiency in the implementation of a common loop

Instead, the student did not identify the ability to cycle a code that can be embedded in a cycle where an index is increased:

```
  for (i = 0; i <2; i ++) {
vnav [i] .style.color = "red";
}
```

6 Discussion and conclusions

The frontal lectures in the course guaranteed that all participants have sufficient knowledge of all CT concepts through the resolution of examples and joint exercises. However, during the execution part of the project, it was noticed that the students experienced problems using the concepts of CT autonomously in their own work context. During the project, in fact, they were required to identify a problem themselves, select the most effective solution based on the introductory part, and finally, the creation of the solution. In this way, we have a first indication that the research question, considering that the Computational Thinking strategy effectively allowed students to take advantage of Computational Science concepts, the creation of computational solutions applicable to real problems, considering that the observed group did not have formal education in software development.

As a limitation of this work, it is clear that the number of students participating in the course is rather small and will not necessarily lead to firm conclusions. Further research or replicating studies are needed to shed more light in the behavior of similar populations on similar educational contexts.

In line with related literature, we concur with the idea that CT is a valuable resource for students, because it allows for timely efforts to develop systematic thinking. Having a design and implementation methodology collaborates to cultivate and benefit from CT skills and put them at the service of subjects of different fronts of their studies (in light that web technologies can be approached from the commercial, communication, visual design and software development viewpoints). However, a clear need can be identified as the observed students typically struggled using directly in source code.

This article describes the implementation of an educational framework to teach CT skills in a non-software context. Five graduate students of a Faculty of Art were able to implement software systems in a web environment using the common web stack in a context of industrial application. Students and teachers participated in a collaborative effort that unites not only Computational Science but other topics such as Art and Design. A two-fold CT assessment strategy is proposed, which includes both automatic analysis and visual inspection. Yet the strategy is efficient helping the analysis of the outcome products, it is acknowledged that the strategy can be improved for robustness and depth. In conclusion, CT skills delivered the necessary resources to help students analyze and decompose a problem, understand their complexity and feasibility, and design a solution to and develop a successful software application.

References

[Bizzarri 2011] G. Bizzarri, L. Forlizzi, G. Proietti; Informatica: didattica possibile e pensiero computazionale. *Proceedings of DIDAMATICA*, 2011.

[Bloss 2001] Adrienne Bloss. Teaching Fundamentals for Web Programming and e-Commerce in a Liberal Arts Computer Science Curriculum. *Journal of Computing Sciences in Colleges.* vol. 16, no 2. pp. 297-302. 2001.

[Brennan 2012] K. Brennan, M. Resnick; New frameworks for studying and assessing the development of computational thinking. *2012 Annual Meeting of the American Educational Research Association (AERA'12)*, Vancouver, Canada, 2012.

[Conitzer 2007] V. Conitzer, T. Sandholm, J. Lang, When are elections with few candidates hard to manipulate? *Journal of the ACM*, vol. 3, no. 54, 2007.

[Cooper 2010] S. Cooper, S. Cunningham; Teaching computer science in context. *ACM Inroads*, vol. 1, no. 1, pp. 5-8, Mar. 2010.

[Corral 2010] L. Corral; Educational Techniques and Classroom Experience on Multimedia Systems Development by Journalism and Communication Undergraduate Students *2010 International Conference on Software Engineering: Theory and Practice (SETP 2010)*, pp. 62-67, ISRST. 2010.

[Gray 2010] J. Gray, A. S. Szalay, A. R. Thakar, P. Z. Kunszt, C. Stoughton, D. Slutz, J. vandenBerg; Data Mining the SDSS SkyServer Database. *4th International Meeting on Distributed Data and Structures 2010*, pp. 189-210.

[Gong 2011] H. Gong, P. Zuliani, E. Clarke; Model checking of a diabetes-cancer model *3rd International Symposium on Computational Models for Life Sciences*, 2011, pp. 234-243.

[Grover 2013] S. Grover, R. Pea, Computational thinking in K12, A review of the state of the field. *Educational Researcher*, vol. 42, no. 1, pp. 38-43, 2013.

[Hambrusch 2009] S. Hambrusch, C. Hoffmann, J. T. Korb, M. Haugan, A. L. Hosking; A multidisciplinary approach towards computational thinking for science majors. *SIGCSE Bulletin*, vol. 41, no. 1, pp. 183-187, Mar. 2009.

[Koh 2010] K. H. Koh, A. Basawapatna, V. Bennett, A. Repenning; Towards the automatic recognition of computational thinking for adaptive visual language learning. *Proceedings of the 2010 IEEE Symposium on Visual Languages and Human-Centric Computing, IEEE.* 2010, pp. 59-66.

[Layman 2008] L. Layman, L. Williams, K. Slaten, S. Berenson, M. Vouk; Addressing diverse needs through a balance of agile and plan-driven software development methodologies in the core software engineering course, *International Journal of Engineering Education*, vol. 24, pp. 659-670, 2008.

[Leoni 2011] L. Leoni; Competenze e competizioni di informatica: valutazioni sperimentali. *Master Degree Thesis*, University of Bologna (Italy), 2011.

[Perkovic 2010] L. Perkovic, A. Settle, S. Hwang, J. Jones; A framework for computational thinking across the curriculum. *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE '10.* ACM, 2010, pp. 123-127.

[Repenning 2010] A. Repenning, D. Webb, A. Ioannidou, Scalable game design and the development of a checklist for getting computational thinking into public schools. *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, 2010, pp. 265-269.

[Troccoli 2005] A. Troccoli. New methods and tools for 3D-modeling of large scale outdoor scenes using range and color images. *Doctoral Degree Thesis.* Columbia University. 2007 Stanford University. Computational Law. `http://complaw.stanford.edu/` 2005.

[Tucker 2003] Tucker, M. D., et al.; A model curriculum for K-12 computer science: *Report of the ACM K-12 Task Force Computer Science Curriculum Committee.* Association for Computing Machinery, 2003.

[Walker 2010] Henry M. Walker, Charles Kelemen. Computer Science and the Liberal Arts: A Philosophical Examination. *ACM Transactions on Computing Education.* vol 10, no. 1, 2010.

[Werner 2012] L. Werner, J. Denner, S. Campe, D. C. Kawamoto; The fairy performance assessment: Measuring computational thinking in middle school. *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education.* 2012, pp. 215-220.

[Wing 2006] J. M. Wing; Computational thinking. *Communications of ACM*, vol. 49, no. 3, Mar. 2006.

[Wing 2014] J. M. Wing; Computational thinking benefits society. *Social issues in computing.* 2014.