Rethinking HCI Education: Teaching Interactive Computing Concepts Based on the Experiential Learning Paradigm

Željko Obrenović
Backbase | obren@acm.org

Interactive computing technologies such as sensors, actuators, and interactive graphical displays have become increasingly common in cars, household equipment, and other consumer products. As such, industrial and product design professionals, traditionally concerned with the physical form and material properties of products, must now take into account issues related to these interactive technologies. These designers need to understand the possibilities and limitations of computing technologies at a sufficient level to be able to engage in a constructive discussion with computing professionals and to be able to create feasible concept proposals for products that use this technology.

Many design schools have begun to introduce courses on computation to prepare students for these new challenges. These approaches are usually based on adapting and simplifying courses developed in computer science schools, such as teaching students the basics of programming, or introducing the general principles of a particular computing technology. With current tools, however, students generally cannot develop a sufficiently good understanding of the capabilities of computing technologies unless they, themselves, are very skilled programmers or developers. In practice many students do not succeed in mastering the syntax of programming. Computing concepts are often introduced with activities (such as generating lists of prime numbers and making simple line drawings) that are not connected to students' interests or experiences. Additionally, such approaches do not recognize that two radically different education models need to be bridged. Design and craft schools generally follow the experiential learning paradigm, in which knowledge is acquired mainly through doing and working on practical projects [1]. Computer science education, on the other hand, has its roots in mathematics, often emphasizing formal methods and models, articulation of general principles, and a top-down approach to problem solving.

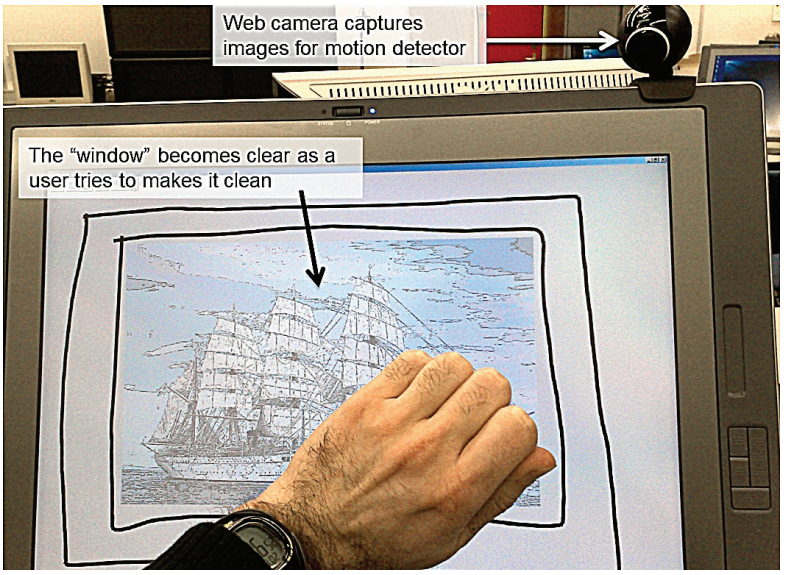
Here we discuss our experiences applying a new educational framework for teaching advanced computing concepts compatible with the practice-oriented educational models used in design and craft schools. Figure 1 provides an illustration of our approach. This example shows an exploration of an "intelligent window" concept, where the visibility of the window is changed by "cleaning" it with a hand gesture. Through this example, we want to illustrate several innovations we have started to introduce in the education of interaction designers:

- The student explores a simplified version of a complex computing component, in this case a camera-based motion detector, *directly experiencing* its possibilities and limitations without needing to know its technical details.
- The exploration and understanding of technology happens *in action*, in the context that is meaningful for students, directly related to the problem they are dealing with (the "intelligent window" concept), and connected with familiar tools, such as sketching tools and graphical editors.
- The exploration is more holistic, enabling the student to reflect on

the relations among user issues, technological possibilities, and overall dynamics of the interaction.

• Even without providing detailed understanding, such experiences can pinpoint the limitations of a technology, such as the need for a clear visual field between the sensor and a user's hands, the influence of lighting on the performance of the sensor, the delay caused by the processing of data, and some indirect consequences, such as the user fatigue when the interaction is prolonged. In early phases, this can lead the student to create solutions that overcome these limitations, such as clever positioning of the sensors, adding lighting elements, and making the interaction sessions short enough to avoid user fatigue.

In our educational framework, this direct experience of exploring computing technologies is a starting point of the learning process, enabling students to come up with an understanding of computation by reflecting on their experiences.



▸ Figure 1. In this example a transparency of a window changes in response to the estimated intensity of hand motion. A motion detector is used to control the transparency of the image representing the window.

Background

Experiential learning is a guided process of questioning, investigating, reflecting, and conceptualizing based on direct experiences. In this learning process, the learner is actively engaged, has freedom to choose, and directly experiences the consequences of their actions. There are several models of the experiential learning process, including Kolb's cyclical learning process [2], Schön's reflective practice model [1], Joplin's action-reflection cycle [3], Kesselheim's learning process [4], and Dewey's three-stage process of learning [5]. Though there are differences among these models, the nature of experiential learning is fairly well understood and agreed upon, and all experiential learning models share the following elements [6]:

• actions that create an experience,

• reflections on the action and experience,

• abstractions drawn from the reflections, and

• application of abstractions to a new experience.

Particularly relevant for our work is the experiential learning model developed by Donald Schön. It is one of the most influential and widely accepted models in design- and practice-oriented schools [1]. This model, sometimes also called "reflective practice," stresses the dynamic, cyclic, and reflective nature of design, in which practitioners approach the solution in cycles. In each cycle they interactively frame the problem, generate moves toward a solution, and reflect on the outcomes of these moves.

The Framework

We have begun to develop a framework for teaching advanced computing concepts based on the experiential learning paradigm. With our framework we wanted to enable industrial design students to experience the design of systems that employ advanced computing technologies, such as speech- and camera-based sensors, or Web services, and to learn from that experience. More specifically, we had the following goals:

• To *empower students* to explore computing technologies without intensive programming. Most of our students are not programmers, and creating systems that employ advanced computing technology using conventional programming languages is beyond their reach.

• To *increase students' awareness* of the possibilities, limitations, and complexity of computing systems. Many of our students are not aware of the availability and opportunities of emerging computing technologies, and they often have unrealistic expectations about technologies and their complexity.

Having in mind these goals and the discussion about the previous work, we adopted several guiding principles for development of our framework:

• We follow the general philosophy behind experiential-based learning: When what we experience differs from the expected or intended, disequilibrium results and our adaptive (learning) process is triggered. Reflection on successful adaptive operations (reflective abstracting) leads to new or modified concepts. The challenge is to create learning environments that are complex enough to lead to unexpected experiences, but not too complex to be inaccessible to students.

• *Unguided or minimally guided experience and reflection is not effective* [7]. We must provide a structure and a set of plans that support the development of informed exploration and reflective inquiry without taking initiative or control away from students.

• *We need to create a personally meaningful context for students.* For a problem to foster the learning of powerful computing ideas, the students must accept it as their problem. We need to take noncontextualized computing ideas and embed them in a meaningful context for student investigation.

Our framework supports these principles with a collection of software tools, conceptual frameworks, and guidelines, which we classify into two groups (Figure 2):

• tools that facilitate the *creation of useful experiences* for exploring advanced computing technology, and

• tools that *guide and support reflection* on such experiences.

Creating experience. The key element of our approach is empowering students to have relevant experiences with advanced computational technologies, because without such experiences, the students do not have a basis to reflect and learn. To support this goal, our framework includes:

• a collection of software tools to empower students to explore the limitations and opportunities of technologies without intensive programming, and

• tools and spaces for creating the contexts for such experiences.

Existing approaches to teaching computing concepts do not usually allow exploration of advanced computing technologies unless a student is willing to become a skilled programmer and learn a significant amount of technological detail. Our goal was to facilitate creating learning experiences, as described in our introductory example, without requiring students to program or to obtain detailed technical knowledge.

The main software tool for our educational activities was the Sketchify toolkit (http://sketchify.sf.net). This software toolkit helps non-programmers build systems with complex computing technologies; it served as the basis for supporting the experiential learning in our courses. In this way, we could bring components from various domains within students' reach, allowing them to directly experience possibilities and limitations of technologies without needing advanced programming skills. Sketchify enables students to combine these technology samples with drawing tools and simple end-user programming techniques, such as spreadsheets. We have incorporated many samples of computing technologies within Sketchify, including text-to-speech engines and speech recognizers, Web services (such as the Google search engine), Phidgets, Arduino, semantic services (such as the Wordnet definition service), camera-based face and motion detectors, MP3 and MIDI players, Wii Remote, a Car Simulator, and many others. (A detailed description of the tool is available in [8]).

In addition to using and developing software tools that can enable students of diverse backgrounds to explore advanced computing technology, we have been working on creating contexts for such experiences. Sketchify focuses on enabling students to exploit the computing technologies available in their environment by allowing them to use everyday computing objects and artifacts, such as their mobile phones, game devices, cameras, or microphones. We developed a number of Sketchify adapters for these objects.

We are also working on creating spaces with specialized equipment and a more stimulating atmosphere. One such space is the ConceptLab, a studio that reflects our vision of what a design studio of the future could look like.

Guiding experience. With the tools described earlier, we can empower students to engage in useful learning experiences. In reference to our second general principle, which states that unguided experience is not effective, we also developed several conceptual tools that can help educators guide and structure students' experience and reflection.

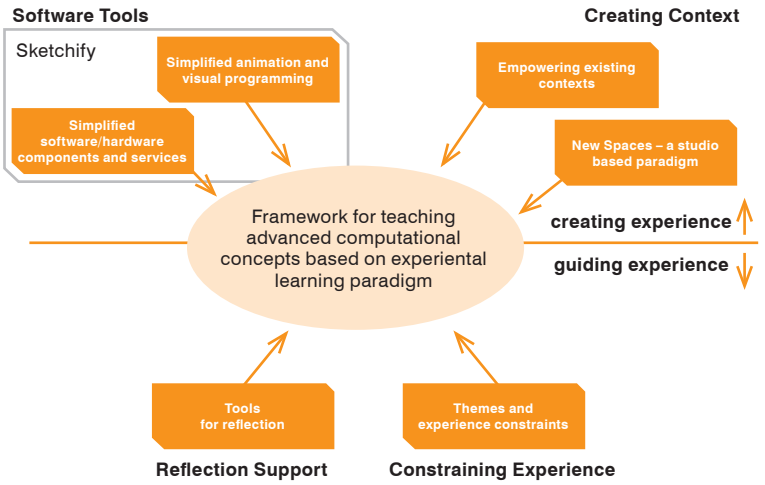
In all of our educational activities, we asked students to keep a creative logbook in which to write down what they had learned and to reflect on the techniques they were using. We also encouraged public discussions, not as mere presentation activities, but as an opportunity to reflect on the experience and learn something new.

To help guide students reflect on discussions, presentations, and notes, we provided several structured frameworks. The main purpose of these frameworks is to give students a structure to

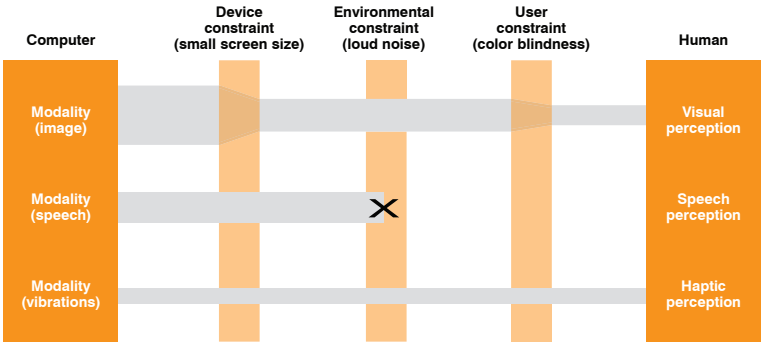
reflect on their experiences and to provide them with a shared vocabulary they can use to critically review each other's work. A systematic analysis and reflection on concrete systems can reveal potential problems and inspire new features. Frameworks for reflection also provide a way to introduce and give meaning to computing concepts. For example, in courses related to the design of interactive systems, we used an adaptation of the framework for modeling human-computer interaction in terms of interaction constraints [9]. This model is presented in Figure 3. The idea behind this modeling framework is that an interactive system could be described in terms of requirements that it imposes on users, such as usage of the visual field or audio perception. This description is discussed in terms of potential constraints that may influence the interaction, such as device limitations, user (dis)abilities, and the environmental influence.

Another important component of our framework is *experience constraints*. While we stimulate students to work on their own problems and set their own goals, when working in groups we found it useful to give a direction to students' activities. Rather than setting a concrete goal, experience constraints are aimed at giving the "mood" to the whole educational setting and activities. Experience constraints thus serve two roles (Figure 4):

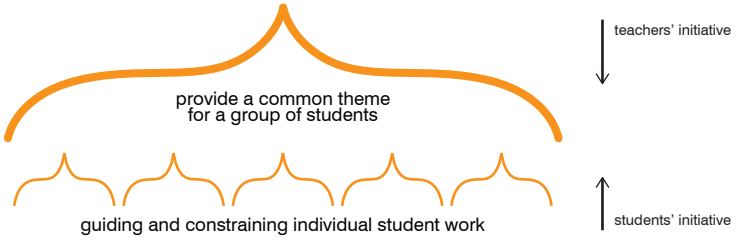
- constrain student explorations, providing inspiration, giving direction, and focusing students' activities; and
- provide a unified theme for student actions and projects, in order to facilitate communication among students.



▸ Figure 2. The framework for experiential learning of computing concepts, consisting of tools that facilitate creation of useful experiences and tools that guide and support reflection on such experiences.



▸ Figure 3. An example of reflection about interaction through interaction constraints. Presenting information through image modality is limited by screen size and users' color disability. Speech may not be perceived in very noisy environments. Using vibrations, such as on a mobile phone, is not affected by the device screen size, environmental noise, or user color blindness, although it has limited information bandwidth.



▸ Figure 4. The role of experience constraints in our framework.

Existing approaches to teaching computing concepts do not usually allow exploration of advanced computing technologies unless a student is willing to become a skilled programmer and learn a significant amount of technological detail.

We used various themes and metaphors to provide experience constraints. For example, in our master's course on multimodal interaction we used the "Power Trio" theme to inspire and unify students' activities. In our undergraduate course "Sketching Interactive Systems" we used the theme of sketching to encourage them to explore more diverse technologies.

Our experience constraints have a role similar to that of a primary generator in design used to "narrow down the space of possible solutions by providing an initial focus, i.e., by constraining and guiding the designer's development of a solution" [10].

Conclusion

Our framework has been developed and applied during a period of three years at the Department of Industrial Design at the Eindhoven University of Technology. We used it in three iterations of the undergraduate course "Sketching Interactive Systems" and three iterations of the postgraduate course "Multimodal Interaction." The first results are encouraging, and although it's still too early to make more specific claims, our initial findings suggest the following:

- The key element of our approach was to empower students to have relevant experiences with advanced computational technologies. Without such experience, the students do not have a basis to reflect and learn. This was a particularly successful element in the usage of our framework, especially in undergraduate education.
- Our tools enabled students to discover and learn a range of important properties of current computing technologies, as well as some basic computing abstractions, such as variables. Though such experience has its limits and cannot enable students to discover all relevant concepts, it provides a productive context to discuss such concepts and increases the general interest of students.
- Providing a structure and a set of plans that support informed exploration and reflective inquiry was crucial to enabling students to learn from their experience and from each other. Simply letting students explore computing technology and build computational systems will not necessarily help them learn computing concepts.
- Having themes and constraining students' experiences had a positive effect on the conceptual integrity of our educational activities and on student collaboration. However, the theme has to be introduced carefully and clearly to avoid confusion among students about its role.

Our initial results are encouraging, but more studies are necessary to get deeper insights into the learning process of students and to develop an empirically grounded theory of how interventions based on experiential learning of computing concepts work.

ENDNOTES:

1. Schön, D.A. *The Reflective Practitioner.* Basic Books, New York, 1983.

2. Kolb, D.A. *Experiential Learning: Experience as the Source of Learning and Development.* Prentice Hall, New Jersey, 1984.

3. Joplin, L. On defining experiential education. *Journal of Experiential Education 4,* 1 (1981), 17-20.

4. Kesselheim, A.D. A rationale for outdoor activity as experiential education: The reason for freezing. *Proc. 1st North American Conference on Outdoor Pursuits in Higher Education* (Boone, NC). 1974, 18-22.

5. Dewey, J. *Experience and Education.* Simon and Schuster, New York, 1938/1997.

6. Stehno, J.J. The application and integration of experiential education in higher education. Touch of Nature Environmental Center, Southern Illinois University, Carbondale, IL, 1986; (Eric Doc. Reproduction Service No ED-285-465).

7. Kirschner, P.A., Sweller, J., and Clark, R.E. Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist 41,* 2 (2006), 75-86.

8. Obrenovic, Ž. and Martens, J.B. Sketching interactive systems with Sketchify, *ACM Transactions on Computer Human Interaction 18,* 1 (March 2011), Article 4.

9. Obrenovic, Ž., Starcevic, D. and Abascal, J. Universal accessibility as a multimodal design issue. *Commun. ACM 50,* 5 (May 2007), 83-88.

10. Lawson, B. *How Designers Think: The Design Process Demystified (4th ed.).* Architectural Press, 2005.

ABOUT THE AUTHOR

Željko Obrenović (obren.info) is a best-practices evangelist at Backbase, Amsterdam. He conducted the work reported here while working as an assistant professor in the Department of Industrial Design at the Eindhoven University of Technology. His professional interests include design of interactive systems, end-user development, rapid prototyping, creativity support tools, software engineering, and universal accessibility.

DOI: 10.1145/2168931.2168945