PARTICIPATORY SYSTEM DESIGN AS A TOOL FOR LEARNING

Claudia Roda
American University of Paris
31, Avenue Bosquet, 75007 Paris France
croda@ac.aup.fr

ABSTRACT

This paper suggests that participatory system design methodologies may be used as a learning tool in academic environments. It reports from a successful experiment in this direction, and discusses how participatory design characteristics and techniques are particularly well suited for interdisciplinary learning experiences.

KEYWORDS

Participatory design, project based learning, interdisciplinary academic work, image database.

1. INTRODUCTION

> *[…] one must expect that I will not be able to tell you my idea of constructionism. Doing so is bound to trivialize it. Instead, I must confine myself to engage you in experiences (including verbal ones) liable to encourage your own personal construction of something in some sense like it. Only in this way will there be something rich enough in your mind to be worth talking about.* *Papert and Harel (1991)*

The majority of the literature, research, and practice on IT enhanced learning concentrates on three types of support to teaching and learning.

First, Learning Management Systems are designed for teachers and course administrators who want to create Internet based learning tools. These tools normally allow teachers to distribute content stressing availability anytime, anywhere, and to manage virtual communities or at least synchronous and asynchronous communication between students and teachers and amongst peers. Other facilities may include: online gradebooks, class lists, course calendars, surveys and exams, and the tracking of students web site use.

Second, a plethora of systems have been designed that support learning of specific subjects. Such systems strive to present instructional material in an adaptive, flexible, interactive, engaging, and often playful manner. Some of them implement radically new, technology enabled approaches. See for example the Multi-User Programming Pedagogy for Enhancing Traditional Study (Phelps, Bierre and Parks 2003).

Third, simulations have been developed to allow learners to experiment with methodologies, concepts, and skills in safe and proactive environments. Some of these simulations "let students modify the model […] or create their own simulations, thus engaging in what educational researchers call constructive modeling" (Hwang and Esquembre 2003).

In constructivist teaching approaches, technology (ranging from digital cameras to complex simulations) is used as a tool supporting students in the completion of projects designed to help them "learn how to recognize and solve problems, comprehend new phenomena, construct mental models of those phenomena, and given a new situation, set goals and regulate their own learning" (Jonassen, Peck and Wilson 1999), the focus is on providing the individual with "tools to think with" and the freedom to pursue self-selected goals (Kafai and Resnick 1996).

Information technologies however, not only have brought us many tools that may support learning such as the ones described above. It has also brought a set of methodologies that computer scientists use to build and analyse the tools they develop. The goal of this paper is to discuss how these methodologies may be used as a learning tool for several disciplines and to initiate a debate on best practices of pedagogical use of system design.

The application of methodologies of one discipline to another is by no means unusual and efforts in this direction often stimulate debates that enhance the understanding of the roots of the various disciplines and encourage interdisciplinary work and fertilisation. For example, several attempts have already been made to import methodologies from other disciplines to enhance our understanding of computer science and programming in particular. These include Schou's and Nord's (1988) suggestion to use literary criticism as a programming pedagogy, theories of program comprehension based on text comprehension theory (Burkhardt, De Tienne and Wiedenbeck 1997, Pennington 1987), and the use of mathematics education research in computer science education (Almstrum, Ginat, Hazzan and Morley 2002).

2. SOFTWARE DESIGN AS A PEDAGOGICAL TOOL

One of the most fascinating and controversial fields of computer science is software design. Software design has been originally formalised as a field of study by Mitchell Kapor who, in his Software Design Manifesto (1991), states that "design disciplines are concerned with making artifacts for human use" and that "the critical role of design, as a counterpart to programming, in the creation of computer artifacts" should be recognised. In Kapor's view software designers stand to software engineers in creating software systems, as architects stand to construction engineers in creating buildings.

Following Papert's tradition and the experiences with Logo, many have argued that writing a program that implements a certain task implies the acquisition of knowledge about the task (Corritore and Wiedenbeck 1999, Roda 1984). David Scherer states: "If you can write a computer program to do something, you understand how to do it. If you write a computer program to model something, you might reach new insights about it." (Scherer 2000)

The same can be said for software design.

In her experiments with children designing software games Kafai (2003) indicated that the "software design projects provided a pedagogical model of how to integrate the learning of programming with the learning of science and math content, and an in-depth look at children's collaborative and planning practices in long-term design projects". Papert and Harel (1991) advocate the same approach in their analysis of system design as a learning environment.

Every system designer, for example, knows that in order to design a system capable of supporting a certain process it is necessary to acquire a very good knowledge of the process. This knowledge is often partial (for example, one does not learn how to fly an airplane simply by designing a flight simulator or a cockpit control system) however it is often very well structured and quite in-depth.

When designing a software system at least two types of knowledge are put into action and exercised: knowledge about software, and knowledge about the domain of the application.

Therefore, as the exercise of programming may serve to encourage an in-depth acquisition of knowledge about the task being implemented, also the exercise of designing a software system can be used as a tool to stimulate the acquisition of knowledge about the system's domain.

In terms of pedagogical use, one advantage of software system design over programming is that it does not require students, who normally wouldn't be computer scientist, to learn a specific programming language. It is necessary, however, for the students to have a quite good understanding of qualitative systems affordances. This means that they should know what software systems such as databases, web browsers, web servers, can do and cannot do.

If the design exercise is conducted in a team including members with some knowledge of software design and members with some knowledge of the domain, the two groups can feed each other information, stimulate each other in further inquiries, and collectively increase each other knowledge about both the system and the domain.

Amongst the many software design methodologies, "participatory design" methodologies seem best suited for this type of pedagogical design exercises because they call for multidisciplinary teams formed by design experts and users who normally bring considerable domain level knowledge.

The classic goals of participatory design are the development of usable systems, and users' empowerment with respect to the use of new technologies (Beyer and Holtzblatt 1995, Ehn 1992, Kuhn 1996). "Advocates of participatory design emphasize the importance of meaningful end-user participation and influence in all phases of the design process" (Kuhn 1996).

When applying participatory design methodologies, the users work together with designers to build systems that fit their needs, and the design process is an interdisciplinary, collaborative, learning process.

3. AN EXPERIMENT IN THE USE OF PARTICIPATORY DESIGN AS PEDAGOGICAL TOOL

This section reports on a "successful" experiment of the use of participatory design for the development of a system capable of preserving and securing access to the image collections of the American University of Paris (AUP) starting with the collection of the Art History Department. The project was, and it is - since only its first phase has been completed - successful in two ways. First of all, the system that has been so far developed was very well received by all the stakeholders at AUP. Second, and most important in the context of this paper, the project has been a successful learning experience for all the students and faculty involved in the project. The project, under the co-ordination of the Academic Resource Center, involves students and faculty from three departments: Art history, Communication, and Computer Science.

In the rest of this section, first we briefly introduce the essential elements of the "development" project then we discuss the "learning" project, how it took place and in particular the role of participatory design techniques.

3.1 The development project

The objectives of the first phase of the development project, which just came to a conclusion, included:

The creation of searchable databases, using appropriate scholarly standards and metadata entries

The preservation, by high-resolution digitization, of slides in the collections of the Art History Department

The research of relevant laws and policies, in order to make recommendations for an AUP Digital Assets Management system in the future (including digital rights management).

Metadata have been selected as a substantial subset of the Getty vocabulary (Getty 2004). They have been used to define both database entries and search fields for two different implementations of the system's database: a Relational DataBase, and an Object Oriented DataBase. The interface for data entry (to be used by the managers of the collection) was implemented for the RDB and the interface for the end user was implemented for the OODB. Procedures for scanning the images were defined ensuring to have copies both in formats for safe storage and printing, and for video display. Laws and policies relative to copyright management for several types of distributions (e.g. a class, AUP students, AUP community, anyone) were studied as well as techniques for preventing illegal reproduction of the images.

3.2 The learning project

The objectives of the learning project included both, knowledge and skills to be acquired by all participants and specific learning objectives for each department. The learning goals shared by everyone where:

- Developing the skills necessary to work in interdisciplinary teams.
- Appreciating the skill necessary to participate in, and manage a large project
- Learn about the affordances of some of the essential hardware and software components of information systems (e.g. databases, web browsers, development environments, web servers)
- Learn about the essential characteristics of an art piece (e.g. authorship, contributors, periods, attributions, schools, materials)
- Learn about the essential issues in digital copyrights

The learning objectives specific for each department included:

- For participants from the Communication department, to acquire an in-depth knowledge of copyright issues both for physical and digital documents, including images. Appreciate the diverse laws enforced in different countries, and how they apply to different environment (e.g. educational use, versus personal use, versus commercial use)
- For Art History participants to acquire the knowledge related to the management of art collections. Including standards for art pieces classification, definition of ownership, and collection control. Also to learn how to appropriately scan images for paper and video reproduction.
- Participants from the Computer Science department were the largest group (two professors and four students). They were all expected to acquire expertise on techniques for system design, and participatory design in particular. Three subgroups were also created with some members

participating in two subgroups. One subgroup concentrated on database design, another group on interface design and usability studies, and the last one on techniques for image protection from reproduction and system security.

All the above goals were achieved. In fact several students learned more than expected in fields that were not their own. For example computer science students learned most aspects of digital copyrights since they needed the knowledge to define the various access levels to be implemented for the system. Everyone learned about client / server architectures due to several discussions about "where the images actually are". Experiences, not foreseen at the beginning of the project, but that allowed the acquisition of specific new skills, added more learning occasions to the project. These included the request by the administration to present the project to the University community at large with the consequent work for preparation and delivery of the presentation; the offer by the computing services department to acquire a new server to host the system with the consequent work for the server specification, and pricing; the decision to experiment with an object oriented database and the consequent search for the database management software that could integrate with our development environment, and the contact process to obtain an educational license.

3.3 The learning experience and its relation with participatory system design

The learning experience that has taken place was characterised by the fact that participants built knowledge structures "by making" something, by "constructing a public entity", by "working with concrete materials rather than abstract propositions". In this sense, the learning experience can be described as constructionism (Papert and Harel 1991) in an environment of project based learning. Furthermore, in this case, the project has the specific characteristics of being a "software design" project, and in particular to apply participatory design techniques.

We believe that participatory software design projects have several distinguishing features that make them particularly well suited to stimulate constructionist type of learning in an interdisciplinary environment.

In the rest of this section we analyse these features. First we consider software design project in general and their fitness to be used for the implementation of constructionism based learning. Then we turn to participatory software design projects and the characteristics that make them well suited for constructionist approaches in multidisciplinary teams.

System design projects in general match the type of projects called for in the constructionism approach, both in the sense of fostering reflective and reactive learning and in the sense of creating a dialectic between the "artifact being built" and the learners.

Constructionism aims at supporting learning both in a "bottom-up" practical way by solving a specific problem, or building a specific artifact, and in a "top-down" reflective manner applying abstractions to the specific problem. Papert and Harel (1991) say: "some people prefer ways of thinking that keep them close to physical things, while others use abstract and formal means to distance themselves form concrete material". We have seen all this happening in our project. Participants with a more reflective approach tended to bring more "academic" types of input (e.g. research papers on HCI, or on the advantages of using open source rather than proprietary software, or on the social consequences of certain digital rights management policies). Participants relying more on reactive approaches referred more frequently to their practical experiments.

Papert and Harel (ibid.) also discuss a "negotiation between the programmer and the work in progress". This type of dialectic matches Schon's "conversation with the materials" in design projects: "sometimes, the designer's judgements have the intimacy of a conversational relationship, where she is getting some response back from the medium, she is seeking what is happening - what it is that she has created - and she is making judgements about it at that level" (Schon and Bennett 1996). What makes the match so perfect? Why a design project is more likely to support this dialectics than any other project? In order for a project to allow learners to actually have a conversation with what they are creating it should have at least two characteristics. First, the project should not have a "solution", a final "correct or wrong" judgement should not be relevant. Second, it must be possible to evaluate the project's outcomes on some, possibly complex and project specific, parameters. In fact a project that has a correct/wrong result does not allow the same level of dialectics than a project that may have efficient/inefficient, beautiful/ugly, comfortable/uncomfortable, types of results. The latter is exactly the case for design projects where the details of the expected results are defined by refinement of its specifications as the project develops.

Here we turn to the relevance of participatory design in the case of multidisciplinary learning experiences. For an heterogeneous team it is not easy to find projects where all the participants have a

real chance to assess partial and final results, and to interact with the project product. With a participatory design approach, the outcomes of the project can be evaluated by the all team, irrespective of member's expertise. Some participants will evaluate the project as designers and some as users but everyone can have a strong saying in the results that are defined, and agreed upon, by all members in the form of specifications and revised specifications as necessary.

Another aspect of participatory design that makes it particularly well suited for constructionist approaches is the facts that it heavily relies on the creation of prototypes (these being paper prototypes, or digital ones). These prototypes enable users to evaluate the project as it develops allowing them to distinguish between good decisions and bad decisions, choices that significantly advance the project and blind alleys.

Three problems are often reported related to learning in interdisciplinary environments. First the general problem of the management of group dynamics, second the problem of heterogeneous approaches to learning, third the lack of a common vocabulary, or understanding of basic subjects.

There are obviously no easy solutions to the first problem. The facilitation of team work, and the timely resolution of conflicts are two essential tasks that teachers will have to undertake with the help of team members. We have noticed however, that in the phases when the group was working at the actual building of the system (as opposed, for example, to situations in which we were discussing procedures) the "team spirit" prevailed over conflicts. Also the self-allocation of specific tasks and responsibilities to each member of the team helped resolving basically all initial fears and conflicts.

As far as the second and third problems they are specifically addressed by participatory design techniques. In particular we have used user interviews and "contextual enquiry" (Holtzblatt and Jones 1993) so that non art history members could understand how the system would integrate in an art history class or how it would help accessing the images. These experiences enriched Computer Science members with an understanding of the art history vocabulary and learning approaches. Similar experiences allowed these types of exchanges for example during discussions on copyrights and on system configuration. Another technique that significantly facilitated the acquisition of a common understanding of fundamental issues was the comparative study undertaken at the beginning of the project. The discussion on the characteristics, advantages, and disadvantages of the systems analysed - which included (Cartography Associates 2004, Luna Imaging 2004, New York Public Library 2004, Virginia Tech 2004) - helped clarifying participant's vocabulary, approaches to the problem, and expectations.

In general, all learning that happened around domain level issues was specifically motivated and reinforced by the design experience. In particular, the investigation of classification and metadata fields - which included reviewing several de-facto or official standards (DCMI 2004, Getty 2004, Library of Congress 2004, National Information Standard Organization 2004) - was guided by the art history members and motivated by the need to define the system's database fields and search facilities. The research in all fair use/copyright issues and digital rights management in academe and in cyberspace - see for example (Educause 2004, World Intellectual Property Organization 2004) - was guided by the communication members and motivated by the need to define access rights and visibility within the system. This part of the work also raised the problem, taken in charge by the art history experts, of defining current locations and rights-holders of physical art pieces of which we had images.

Those briefly discussed here are only a few of the most formal issues that were raised during the project design. Many others, less formal, issues were raised and investigated by the team. For example the team analysed the contributions and requirements of the many stakeholders at AUP and the multiform aspects of art history image preservation and use in various disciplines and contexts.

The students, who acquired significant knowledge related to the classification of art pieces, information systems, and copyright legislation, were extremely demanding with themselves and with each other and covered, within the span of a single semester, a lot of content. Perhaps more importantly they all reported having more confidence in their ability to take part in a large, "real-life" project; to have a better understanding of the problems involved with working with people with a background different from theirs; to have a better appreciation of the importance of clear problem definition in a team and the role played by written reports of their work; to understand the relevance of communicating with stakeholders who may not have a direct involvement in the project but may greatly facilitate it or hinder it.

4. CONCLUSIONS

The system design process served as a methodology to exchange knowledge between students and faculty of different departments and allowed to apply "learning by doing" and constructivist type of

learning to an art history subject. Although this was a single experiment we believe that the same type of approach could be applied to many other subjects and disciplines. Most of the learning happened at the level of the system design, in fact many of the participants in the project did not get into the details of the implementation. They were simply prompted by questions coming from the computer science students, and they were asked to go through scenarios and prototypes. The first phase of the project was articulated over a semester, however smaller projects may be envisioned to last only a few weeks. Beyer and Holtzblatt (1995) state that "the fundamental problem in the relationship between customers and designers is that of enabling learning" in our case, the students are both customers and designers. They can learn from exchanging and formalising their own knowledge of the subject and the knowledge of their colleagues.

Wilson (1995) states that "organizing instruction around problems and cases should not mask the importance of perception, reflection, and metacognitive activity. Indeed, these two aspects of human performance (problem solving and perception) can be seen as inherently complementary and equally necessary.". Participatory design works also at the level of perception, reflection, and metacognitive activity. During our project this was further emphasised by the existence of a shared "note book", implemented as an online community where members of the project could annotate project advancement, references discovered, but also reflections about the work we were doing and the learning involved.

ACKNOWLEDGEMENT

The project described in this paper was founded by a grant of the Mellon Foundation. It was run under the supervision of the director of the Academic Resource Center, Ann Borel. The project team for the spring semester 2004 was formed by the students: Dharit Anjaria, Paul Cociuba, Joumana Hassan, Sarah Rozelle, Nathania Stambouli, Isabelle Ulfsdotter, and the Professors Evgeni Gentchev, Julie Thomas, Kathleen Wilson-Chevalier and myself. Technical support was provided by Eric Cisternas, and Pierre-Yves Vasener. The author also wishes to thank AUP's Vice President for Academic and Grant Planning Celeste Schenck for her continuous support and encouragement, and Professor Jim Clayson for his contagious curiosity about human learning and trust in constructionism.

REFERENCES

Almstrum V. L., Ginat D., Hazzan O. and Morley T., 2002. Import and export to/from computing science education: the case of mathematics education research. *7th annual conference on Innovation and technology in computer science education*. 193 - 194

Beyer H. R. and Holtzblatt K., 1995. Apprenticing with the customer. *Communications of the ACM*, 38(5), pp. 45 - 52

Burkhardt J.-M., De Tienne F. and Wiedenbeck S., 1997. Mental representations constucted by experts and novices in object-oriented program comprehension. *INTERACT97: 6th IFIP International Conference on HumanoComputer Interaction*. Amsterdam: North-Holland, 339-346

Cartography Associates, 2004. http://www.davidrumsey.com/amico/preview.html. Accessed: 5.5.2004

Corritore C. L. and Wiedenbeck S., 1999. Mental Representations of Expert Procedural and Object-Oriented Programmers in a Software Maintenance Task. *International Journal of Human-Computer Studies*, 50, pp. 61-83

DCMI, 2004. http://www.dublincore.org/. Accessed: 5.5.2004

Educause, 2004. http://www.educause.edu/issues/issue.asp?Issue=DMCA. Accessed: 5.5.2004

Ehn P., 1992. Scandinavian Design - on skill and participation. Usability - Turning technologies into tools. P. Adler and T. Winograd

Getty, 2004. http://www.getty.edu/research/conducting_research/vocabularies/. Accessed: 5.5.2004

Holtzblatt K. and Jones S., 1993. Contextual Inquiry: A Participatory Technique for System Design. Participatory Design: Principles and Practice. D. Schuler and A. Namioka Ed., Hillsdale, NJ, 180-193

Hwang F.-K. and Esquembre F., 2003. Easy Java Simulations: An Interactive Science Learning Tool. *Interactive Mulitmedia Electronic Journal of Computer Enhanced Learning*, 5(2)

Jonassen D., Peck K. and Wilson B. G., 1999. *Learning with Technology: A constructivist perspective*. Prentice-Hall, Inc, Upper Saddle River, NJ

Kafai Y. B., 2003. Children designing software for children: what can we learn? *Interaction design and children.* Preston, Engalnd, 11-12

Kafai Y. B. and Resnick M., 1996. Introduction. Constructionism in Practice. Y. B. Kafai and M. Resnick Ed., Mahwah, NJ, 1- 8

Kapor M., 1991. A software design manifesto: time for a change. *Dr. Dobb's Journal*, 172, pp. 62 - 68

Kuhn S., 1996. Design for people at work. Bringing design to software. T. Winograd Ed., New York, 273-289

Library of Congress, 2004. http://www.loc.gov/standards/standard.html#ead. Accessed: 5.5.2004

Luna Imaging, 2004. http://www.lunaimaging.com/insight/index.html. Accessed: 5.5.2004

National Information Standard Organization, 2004. http://www.niso.org/. Accessed: 5.5.2004

New York Public Library, 2004. http://digital.nypl.org/schomburg/images_aa19/. Accessed: 5.5.2004

Papert S. and Harel I., 1991. Situating Constructionism - Chapter 1. Constructionism. S. Papert and I. Harel

Papert S. and Harel I., 1991. System design as learning environment - Chapter 4. *Constructionism*. S. Papert and I. Harel

Pennington N., 1987. Stimulus structures and mental representations in expert comprehension of computer programs. *Cognitive Psychology*, 19, pp. 295-341

Phelps A. M., Bierre K. J. and Parks D. M., 2003. MUPPETS: multi-user programming pedagogy for enhancing traditional study. *4th conference on Information technology education.* Lafayette, Indiana, USA

Roda C., 1984. Inserimento dell' Informatica nella Scuola, Seconda parte. *Insieme*, 3/4, pp. 14-18

Scherer D., 2000. *Pedagogy, programming environments, and readings*. EDU-SIG discussion list

Schon D. and Bennett J., 1996. Reflective conversations with material. Bringing design to software. T. Winograd, 171 - 184

Schou C. D. and Nord R., 1988. Literary criticism and programming pedagogy. *ACM sixteenth annual conference on Computer science Atlanta*. Georgia, United States, 67 - 71

Virginia Tech, 2004. http://imagebase.lib.vt.edu/. Accessed: 5.5.2004

Wilson B. G., 1995. Situated instructional design: Blurring the distinctions between theory and practice, design and implementation, curriculum and instruction. Proceedings of selected research and development presentations. M. Simonson Ed., Washington D. C.

World Intellectual Property Organization, 2004. http://www.wipo.org/. Accessed: 5.5.2004