0165

# Teaching Interaction Design: Matters, Materials and Means

**S Lundgren1**, O Torgersson1, L Hallnäs1, P Ljungstrand2, E Eriksson1
IDC | Interaction Design Collegium, Gothenburg, Sweden 1; Chalmers University of Technology, Gothenburg, Sweden 2 | jan.lundberg@ltu.se

On Interaction Design

Interaction design is a multidisciplinary subject bordering to areas like product design, industrial design and software design, cognitive science and many more. As the name states, it deals with the *design of actions between* humans, software and technology, together with all the other materials that build the things we use. This includes any kind of product containing software, be it a computer supported tool for cooperation and coordination within a world-wide company, a mobile phone, the interface between the driver of a car and the complex software that serves as aid, or a robotic toy.

However drawing from industrial design and interface design when it comes to designing actual products or interfaces, the interaction designer is highly concerned with, yes, the *interaction* with the product, including social and cultural aspects. That interaction designer does not only consider what happens when one user uses the product/interface, but what happens if many users use it; how this affect users, society and culture. When comparing the industrial designer and the interaction designer, the latter deals with the interaction with objects that can be said to have some kind of complex interaction behavior, due to their embedded software. To put it bluntly one can say that designing a hammer, or even a lawn mower is mostly industrial and engineering design, whereas designing an interactive toy such as for instance Sony's robot dog Aibo (Sony, 2006) is interaction design, simply because the robot dog is capable of a superior amount of complex behaviors and reactions.

Interaction design is also related to product design in the sense that these two disciplines not necessarily must focus upon making products whose most important property is its usability. Instead, the design can be targeted towards a product that is fun, entertaining, beautiful, puzzling or whatever other wanted quality, rather than useful in a strict sense. Thus, the interaction designer's goal can be said to be: To create an interactive object that with its responses affect the use, the user and the context in the intended way.

## Interaction Design as a New Discipline

Since computer-enhanced products and systems become more and more important in daily life, there is a growing need, for interaction designers, and obviously this new *corps de design* needs to be educated first. However, this is a bit tricky, since the interaction designer needs to be familiar with so diverse subjects such as for instance programming, material science, graphic design, human cognition, human computer interaction, electrical engineering, interface design and/or product design and a dozen more subjects, and needs to have substantial knowledge in at least two of these fields.

## Computational Technology as a Design Material

The design material common for all interaction designers is computational technology, which can be seen as a design material among other design materials (Dunne 1999, Hallnäs et al 2001, Hallnäs & Redström 2002b, Löwgren & Stolterman 2004). It isn't just dull technology implementing neutral technical specifications; it is – or can be! – an expressive material that can be used to create expressive behavior.

The computational material is simultaneously both abstract and concrete, both imaginary and material, both software and hardware. It manifests its form and expressions in the spatial, physical realm through displays of various sorts, but the true nature of this material is primarily temporal, as the dynamic motion of executing program code is its essence. This material allows for precisely controlled dynamic behaviours, communication, as well as adaptation to new or local conditions. This material is central to the field of interaction design. However, it is not sufficient for successful interaction design on its own. Other, more well-known materials are needed to form and shape the things we will come to use and live with. These "other" materials are primarily physical, such as plastic, metal, glass, wood, textile, etc.

By viewing computational technology as a material for design, we primarily take a perspective of use, rather than of function. Instead of primarily defining, modelling and understanding the properties of the system in technical terms that can be measured using physical measures and described with mathematical models, as in engineering, we are more interested in what we can do with it from a use perspective. Still, this understanding of use must be rooted in a deep and thorough understanding of the technical characteristics of the component in question, which to some degree requires an engineer's viewpoint, but is essentially something different, as it is more abstract and requires you to go outside of the technological system view and to look at the system in the context of its (imagined) use. Of course nothing really changes with the actual material; it is all a matter of perspective.

The choice of the technology behind a product matters, and matters a lot. Compare the regular phone with a mobile phone. Same type of product, same area of use, but…! Only the laws of programming and computational behavior set the limits of what can be done. It's a flexible, magical material, but also fragile (Landin 2005). To its nature it is temporal, and abstract. It can best be compared with music, since it "builds" things when programs are run in a certain context, just as music only exists while played. They are both only present whilst in action, so to speak. In this, computational power differs from more static materials that are always present, always have the same properties, always have the same expression, once designed. Knowing this, the obvious conclusion is that no interaction design can be created without an extensive feeling for the possibilities, advantages, problems and limitations of code in combination with other technology – we need to teach the designers more on programming and the programmers more on design.

## Send in the Programmers: Bringing Design Education to Computer Science

The rest of this paper describes the methodology applied in the Masters program in Interaction Design entitled Human-Computer Interaction – Interaction design (HCI|ID) run by yy at xx University of Technology / The IT University of xx, in xx, yy. Interestingly, this means that the education belongs to the Computer Science and Engineering department, placing this design-intense subject in a tech-oriented environment. Therefore, the education is an interesting mix on how to teach design to non-designers (mostly computer engineers), and although it lends many classic ideas from traditional design education, new elements have been added in order to suit the multidisciplinary nature of both the subject and the students.

The HCI|ID program runs for three semesters where the last semester is spent on thesis work. The academic year is split into four periods and two courses are taken in parallel during each period. Courses with a more theoretical approach are: the two *HCI courses*; *Design Methods* where the focus is issues related to aesthetic topics; *Methods of Analysis* dealing with identifying needs and setting requirements as well as evaluating existing systems; and finally *User Centered Design*. The two courses *Graphical User Interfaces* and *Ubiquitous Computing* have a strong focus on practical work and high-fidelity prototyping, and so has the *Interaction Design Project* – in the latter all skills taught during the first semester are integrated and called upon in a fairly free project with full implementation requirements. For more information about the HCI|ID program, see (website x, 2006).

Of the circa 45 students that enrol each year, about 60% come from an engineering background, but many other subject areas are accepted, ranging from cognitive science to art via psychology etc. Since a majority of the students do have a technical background, involving elements of technology into courses is both feasible and desirable. This leads to an exploration of practical functionality and how to realize it, but also to the study of basic aspects of the appearance of this material as it is used in design, such as its expressiveness and aesthetics (Dunne 1999, Hallnäs & Redström 2002a, Hallnäs & Redström 2002b). In addition, there is a strong influence from problem-based learning as well as the ideas of reflection in action expressed by Schön (1983). Several courses, but not all, have strong influences of learning guided by ideas about constructionism as proposed by Seymour Papert, i.e. the construction of knowledge in the context of building personally meaningful artefacts (Kafai & Resnick, 1996).

Thus, the main focus is upon practical design – building fully working prototypes for reasons soon to be described. Even if practice makes perfect, theoretical knowledge is necessary for the understanding of various design principles. Accordingly, 40 to 60% of the courses (depending on individual students' choices) focus on subjects such as human cognition and perception, requirements analysis, evaluation methods, design methods, and design aesthetics. The latter two are essential when dealing with engineering students; they need to be trained to combine and enforce their technologically oriented view on the world with the designer's more free and unpredictable way to approach a problem. Vice versa the non-engineer students need to get an understanding for what technology can provide in the practical courses.

Practical work is supported and promoted by providing design studios where students can leave their work from day to day; they do not need to clear away their work every evening. The studios are equipped with an electronics lab, materials of various kinds and numerous tools for prototyping.

## The Art of Prototyping

Since computational power is only present in action, it is hard to design for it, without making extensive prototypes. We have to "test" in order to explore the expressions we are designing, and this can hardly be done without implementing the behavior to some extent. To compare with music once again, we can say that anticipating what a program will do and how it will express itself is pretty much the same as looking at a score and anticipate what the music will sound like. The more complex, the harder to predict of course, especially if users can interfere and steer the flow of actions.

Of course prototyping can be made on different levels of complexity. Sketches, scenarios, scenario acting, fake prototypes (for instance with a human carrying out the computer's tasks, commonly known as Wizard of Oz) can give more or less useful hints on how the product will behave – often valid enough to trigger redesign – but only an almost fully working prototype can provide a true sense of the code and the interaction in action. Therefore, it is our belief that the interaction designer has to take this last step and experiment with the material in concreto through working prototypes, applying high-fidelity prototyping. While programming and software is essential to interaction design, it is not sufficient in itself. Any piece of software still needs some hardware to run on, and interface components (sensor and actuators, e.g. keyboard, mice and screen) to interact with human users and present the result of the executing program code in some way. However, as computational technology becomes embedded in almost everything around us and the interface components vary greatly, we see it as essential that the students also familiarize themselves with alternative means of interaction with computational systems. This includes hands-on working with embedded electronics and sensor and actuator technologies, but even more so to develop skills in selecting and combining various materials into a meaningful whole. In this sense, we are similar to product or industrial design, with the major difference that we primarily work with the interactive components (software, algorithms) and their physical extensions (sensors, actuators) .We often push the projects into fully working prototypes, mainly because the intended interaction can otherwise never be fully experienced and understood.

Our teaching approach is to have the students perform several projects, ranging from two to ten weeks. The projects are developed iteratively from idea to prototype with various degrees of implementation. In this way the students gain both theoretical and practical knowledge in prototype development and evaluation of prototypes.

## Working in Heterogeneous Groups

Working together in projects is a fundamental part of the curriculum, and in almost all courses the groups work together on a larger project resulting in a working prototype. The reasons for this are at least threefold:
1) The students need to get an understanding of the gap between the initial idea and the implemented artefact. Throughout these design processes students make numerous design choices of which many are not answered by literature nor user studies; they have to be implemented in order to be analyzed.

2) The students come from different backgrounds, meaning that they can learn a lot from another by working side by side; knowledge that cannot be conveyed with an ordinary approach to teaching within engineering.

3) Working with students from other disciplines enables projects which are less like prepared school exercises, allowing students to include a wider range of skills in the process, and give the practical knowledge of working with people from other professional backgrounds.

Due to the varying educational profiles of our students, we can deliberately put together groups with different skill sets. In particular, all of the most prominent background skills (graphical design, programming, and microelectronics) are possessed by one or two *but not all* members of any given group. The students are not free to create their own groups in the early courses of the education. Rather, groups are put together based upon the students' background and self-expressed skills. However, the students are not informed of their assigned role within the group to avoid problems with prejudice meanings. Hence, the working roles are not externally cemented from the beginning of a course. Note that a mismatch will not affect the entire education, since students are re-grouped for each course and they can create groups on their own in the later courses. The rationale for this freedom in later courses is that students will then have learned to work in heterogeneous groups and should be able to form groups around collectively created ideas that engage, even if different participants favour different aspects.

## Open and Given Problems

In order to train students in seeing that functionality is an abstraction expressed and that these expressions have their roots in technical functionality, project work goes back and forth between given and open project problems. In industrial design education this shifting is often standard procedure, and so we have included this approach.

*Given problems* are rather straightforward, e.g. "design and implement a web browser on a PDA display", which allow students to sense if they have solved the problem by comparing to some external requirements. These problems evolve around *how something can be done*, and explore general questions about materials and techniques. The given problem is a tool, a framework for exercises, and the students will discover the general more abstract "hidden" problems bit by bit during project work.

*Open problems* on the other hand, e.g. "design and construct a computational thing with focus on its expressiveness", do not provide students with specifications to measure completeness. These problems are initially abstract, often vague and focus on expressional aspects of interaction design, asking *what something would be like*. Here, students have to discover functionality as something inherent in the expression of things in use. As opposed to *given problems* the more concrete problems are discovered by and by. In relation to each other, two types of problems also allow students to explore the double perspectives of functionality, expressiveness and aesthetics (as related to computational technology).

## Exhibitions

Each semester in the first year ends with an exhibition showing fully functioning prototypes. In the first semester, the projects from the *Ubiquitous Computing* course are demonstrated. These are oriented towards exploring sensors and electronics to design interfaces that are not based on a traditional screen and keyboard, in an approach described as viewing computer technology as a design material (Redström 2001).
In the second semester, the results from the *Interaction Design Project* course are shown to the public at a one-week exhibition. The main theme of the exhibitions and the projects vary, but the focal point is always on expressiveness and gestalt.

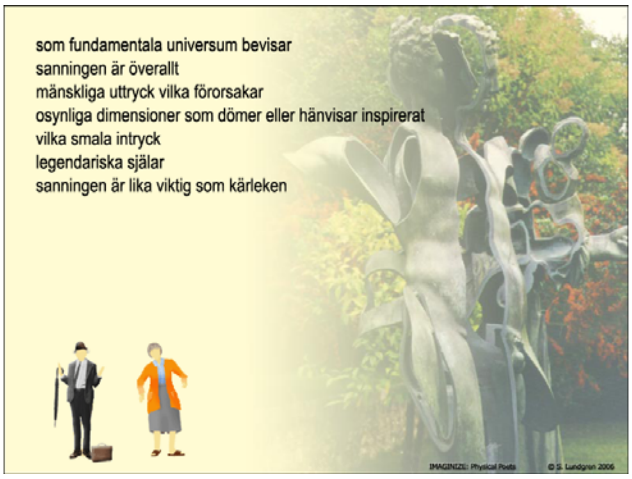
## Case: Physical Poets

One example of a typical, extremely multidisciplinary project is one just recently run in the Interaction Design Project course. It is called Physical Poets and was a collaborative project for the entire class in the sense that each group had to contribute a Poet to the system.
Physical Poets is an application that creates poems. These are shown on a screen with an appropriate, theme-related, background, but the input to the system is not screen-based; it is made placing a number of physical impersonations of the poets (wooden dolls) on a podium, thus triggering the creation of a poem. The poets can speak about several different themes (i.e. loneliness, nature and love), and they also react on each others personalities, adapting their own behavior and mood to the people present in the group of active poets. This affects how often, eagerly and long they speak.

The main purpose of this project was to have the students train how to create a rich persona[EE1] (i.e. a poet), and how to express this personality in several different ways; in the code, that expressed the behavior (i.e. with programming), in the words and sentence structures assigned to him or her (i.e. by understanding how the code worked and combine it with words and phrasings typical for that persona), in the physical appearance of the doll impersonating the persona (i.e. by creating clothes and choosing a body posture), etc. All of this had to rest on a firm foundation of a written persona description. In order to come up with a persona several special design methods were used. Each doll had an embedded RFID tag at the base, and inside the podium there was three antenna coils connected to an RFID reader and a PC running the poetry generation software.

Students Wolfgang, Thomas and Karin working in the design studio

The outcome was poets like Carl Henric, a misunderstood teenage idealist strongly involved in the environmental movement; Roffe, a middle-aged redneck loving fast cars and to impersonate Elvis Presley; Ulla-Britt, a caring, generous, semi-religious mother-like character; and Ture, a rigid old man wondering what has gone wrong with the youth and today's society. In all, there were eight poets, and together they created gems like this one (on the theme "deep thoughts"):



som fundamentala universum bevisar
sanningen är överallt
mänskliga uttryck vilka förorsakar
osynliga dimensioner som dömer eller hänvisar inspirerat
vilka smala intryck
legendariska själar
sanningen är lika viktig som kärleken

*as fundamental universes prove*
*the truth is everywhere*
*human impressions that cause*
*invisible dimensions which judge or refer inspired*
*what narrow impressions*
*legendary souls*
*truth is as important as love*

## Result: Learning by Burning

The combination of heterogeneous groups, high fidelity prototyping (as opposed to concept oriented design) and open problems together creates a real-life-like learning style, in order to prepare our senior students for work life. We jokingly call this style *learning by burning*; burning as in being burnt by making mistakes, but also as in being on fire; to passionately engage oneself in projects and course work. As any other approach it has its pros and cons; stress being one of the downsides, the insight that one actually *can* conquer, manage and master new technologies one of the advantages.

## Conclusion: Lessons Learned

### Prototyping and Public Exhibitions

As discussed above we believe that high-fidelity prototyping is essential to provide students with a thorough understanding of what it means to work with computational technology as a design material. However, high-fidelity prototyping is also an important means for setting students on fire: the satisfaction achieved from seeing ones ideas, through hard work, turn into a working prototype that can be shown to friends, relatives, and the public at an exhibition must not be under-estimated. To be able to actually try out the intended interaction is also important since the focus on interaction is on of the things that set interaction design apart from, e.g., industrial design.

Here, we see a strength in the mix of students; their miscellaneous backgrounds all melt together seamlessly in order to create whole, rich, diverse prototypes, that could probably not have been made without such a wide range of competences, background-related views and ditto opinions.

However we must point out that even if the students are ever so competent in programming etc, they need very much support when creating their first physical interactive prototypes, since these normally involve sensors and microelectronics; electrical engineers are unfortunately conspicuous by their absence. Not only do they need a fully equipped electronics lab, but also very much supervision and support from teachers and assistants. This need would be even more urgent if all the students were design students. Similar issues has been noted at other universities with related educational approaches (Winograd & Klemmer 2005).

### Working in Heterogeneous Groups

Creating heterogeneous groups has numerous positive effects: firstly, the final design will become better since – again – many different views are applied upon it and secondly the students have to train how to communicate with people that have a whole different set of views and agendas when it comes to design. This is especially important since the average company or project only has one interaction designer, however working tightly together with programmers, database experts, interface designers, product designers, project leaders, customers, testers etc. Therefore this practice in how to explain the design and its main ideas to anyone is very useful.

Thirdly, the students practice discussing design, analyzing design, motivating design. This is especially important for those of the students that have a non-designer background; they need to learn words and expressions as well as pick up views and tools for communicating design.

Finally, students also learn a lot from each other; a customized learning process.

The outcome of our surveys shows that students value working in mixed groups. As an example, in the evaluation of our latest course 72% of the students acknowledged that the group together reached a better result than they would have been able to do by their own. Throughout the four years the education has been run, a rough estimate shows that there are quite serious cooperation problems – or worse - in circa 10% of the groups. In the mentioned evaluation only 5 students out of 52 rated the group's cooperation as being "bad" or "lousy". The risk for disagreements is increased due to the fact that misunderstandings depending on different backgrounds and views are common. In most cases the group solves these problems internally, and learns from the process. This disharmony is a price the students pay for the possibility to learn how to work with people from other backgrounds. So yes, students may get burnt by a dissatisfying group experience, but these heated discussions also thicken their skin, making them more ready to face reality.

## Open Problems

The diverse background of the students sometimes makes it difficult to find the right balance in design exercises. Investigating the materiality of computational technology in actual construction work is a difficult task for students with a weak background in computer science or electrical engineering. Working with expressional aspects of computational technology is just as difficult for students lacking design experience. Providing extensive supervision is a way to work around this. Particularly, open problems tend to frustrate some students, especially those from a strong engineering background. Initially they ask for borders and limits, but when they finally understand that the projects are free, they fly. This freedom spurs enthusiasm; suddenly the students are on fire, working hard to fulfil their very own fantastic ideas. Of course this energy, combined with time-pressure and wishes to present well at the exhibitions, sometimes lead to meltdowns.

## Success Rate?

Our aim is to ignite the students' passion for interaction design. To measure to what extent this has succeeded is of course not trivial but one quantifiable measure is very clear: we have essentially no drop-outs. Of the students admitted to the program each year at most one or two do not finish the first year. A few more take longer than the planned semester to finish their Master thesis work, but compared to other comparable educations at the IT-university of xx we have a very high turn-around of students. Furthermore, many students have taken the initiative by their own to get their work published at various international research conferences and journals (website y, 2006).

## Conclusion: What's In it for You?

We hope that this paper has inspired other design educators. We would especially like to advocate more cooperation between design students and engineering students everywhere, by hosting such multidisciplinary projects as those within interaction design. When staging such a project it is important to plan it carefully in order to avoid the havoc that arises when people and disciplines collide. Make sure every group has members possessing all competences needed to carry out the project. If possible, have a 50/50 female/male ratio. Give groups a soft start by giving them a smaller assignment to solve first, so that they can get to know each other, each others views, approaches and opinions before the real project starts. Point out that they *are* different and that they need to deal with this. Preferably this first assignment can cover a topic that is new for most or all of the students; they will still go about solving the problem in their traditional approach, and they will still have to sort out their misunderstandings. Remember: Engineers can be frustrated by open problems; they tend to want clear boundaries. Remember: Designers tend to be frustrated by, or scared of, or prone to avoid, thinking in terms of technical solutions and possibilities, which can cripple their design process. Remember: all will need support – from supervisors or each other – in the areas they do not master.

We also hope that we have inspired some designers to try and start exploring this wonderful new design material of ours: computational power.

Björn demonstrating the student project "Wake up get up" – a bed with an integrated light and sound system, in order to wake people up in a good and lenient way (Bernholdt Olsen et al 2006).

**Acknowledgements**

We would like to thank our fellow educators, past and present: x,y,z,s,t,f,g, and u… and most of all we would like to thank the students who work and grow, and who intrigue, inspire and challenge us.

## References

**Books**

Dunne, A. Hertzian Tales; Electronic products, aesthetic experience and critical design. RCA CRD Research publications; London, UK, 1999.

Löwgren, J. and Stolterman, E. (2004) Thoughtful Interaction Design: A Design Perspective on Information Technology, MIT Press.

Kafai, Y and Resnick, M. (Eds.) (1996) Constructionism in Practice. Designing, thinking and learning in a digital world. Lawrence Erlbaum Associates, New Jersey.

Redström, J. (2001) Designing Everyday Computational Things. Ph.D. thesis. Gothenburg Studies in Informatics, No. 20, May 2001. Dept. of Informatics, Göteborg University.

Schön, D.A. (1983). The Reflective Practitioner: How Professionals Think in Action. Basic books.

Articles & Conference Papers

Bernholdt Olsen, J., Stelling, S., Söderström, R., Wikman, T., Östlund, B. Wake Up Get Up: A better way to wake up. In Proceedings of Scandinavian Student Interaction Design Research Conference (SIDeR) 2006, Göteborg, Sweden.

Hallnäs, L., Jaksetic, P., Ljungstrand, P., Redström, J., & Skog, T. Expressions; Towards a Design Practice of Slow Technology. In: Proceedings of Interact 2001, IFIP TC.13, 2001, Tokyo, Japan.

Hallnäs, L. & Redström, J. From Use to Presence; On the Expressions and Aesthetics of Everyday Computational Things. In: ACM ToCHI,Vol. 9, No. 2, June 2002, pp. 106-124. ACM Press.

Hallnäs, L. & Redström, J. Abstract Information Appliances; Methodological Exercises in Conceptual Design of Computational Things. In: ACM SIGCHI DIS 2002, pp. 105-116. ACM Press.

Landin, H. (2005) Fragile and Magical – Materiality of Computational Technology as Design Material. In Proceedings of Critical Computing 2005, Århus, Denmark

Winograd, T and Klemmer, S. (2005). HCI at Stanford University. Interactions, Vol 12, No. 5, pp. 30-31. ACM Press.

**Websites**

Royal College of Art on interaction design, Irene McAra-McWilliam quote:
http://www.interaction.rca.ac.uk/overview/interaction.html. Last visited April 3, 2006.

Sony's robot dog the AIBO
http://www.sony.net/Products/aibo/ Last visited April 3, 2006.

Website x: The home page of the HCI|ID education:
http://www.......... Last visited April 3, 2006.

Website y: The home page of student's publications
http://www.......... Last visited April 3, 2006.