INTERACTIVE PROTOTYPING: TEACHING INTERACTION TO INDUSTRIAL DESIGNERS

Ilpo Koskinen[1] and Jussi Mikkonen[2]

[1]School of Design, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, ikkoski@polyu.edu.hk
[2]Aalto University, Helsinki, Finland, jussi.mikkonen@aalto.fi

ABSTRACT:

This paper describes the history and pedagogy of Interactive Prototyping, a MA level research class in which students of industrial design have eight weeks to design a functioning electronic prototype and evaluate it with users. The class was first conceived in Helsinki in 2007, and it has run continuously ever since, making 2014 the 8th edition. In the beginning, the instructors were a sociologist and an engineer, but since then, the class has been co-taught also by computer scientists and industrial designers. The class has been run sometimes as an academic class, but it has also cooperated with several companies, including Microsoft Cambridge, and Nokia. The class is neutral to materials and technologies as long as the design solution builds on a system of sensors, microcontroller, and actuators.

Keywords: interaction design, industrial design, prototyping, user-centered design

1. INTRODUCTION

Industrial design has been in a forerunner in design education since the seventies. Many new initiatives in design have come to design through it, including design management in the eighties, CAD/CAM in the early nineties, user-centered methods in mid-nineties, new breed of constructive research around 2000, and service design in the second half of the noughties. One of the changes industrial design has been able to incorporate has been interaction design. Interaction for industrial designers went from industrial user interfaces to the Web and mobile devices in the nineties, and then branched out to interactive devices of various sorts, before turning into service and community design with social media.

All these changes have set many new requirements to industrial design education, which has traditionally been focused on the material interface of technology. In particular, it has raised the question of how to teach designers to think in terms of technology, and how to translate these insights into physical products that have at their core a concept of interaction. This paper describes how we dealt with these issues in an industrial design program in former University of Art and Design Helsinki. It was evident that industrial designers could not only be content with creating concepts, but had to take a step forward towards constructing technology as well. The question was, how to do it best?

When thinking about how to respond to this question, we had several hypotheses around. One way stressed theory and learning (for instance, Friedman 2003), but we discarded this hypothesis as historical evidence told us that design has never been successfully been build on theory (Maldonado 1984: Alexander 1971; Jones 1977; McIntyre 1995). A more targeted and practical alternative came from design-oriented engineering programs. Our response was influenced by places like TU/Eindhoven, which applied constructive pedagogy to design, IDEO's process, which we learned from Jane Fulton Suri, empathic design, and the Interactive Telecommunications Program of Tisch School of the Arts in New York. There were flourishing scientific communities behind this tack, which we knew also worked in practice, as it had been working for decades in design teaching.

The problem with this hypothesis was technology: few design students have skills that could enable them to easily delve into programming at levels good enough to be competitive with these partners. How could we give a crash course into sophisticated technology that was unfamiliar to design students without sacrificing the empathic foundations of design education?

2. PRELIMINARIES

Our response to this question was a class we called Interactive Prototyping or, for brief, IP. It was piloted first in Spring 2007 with doctoral students and run for the first time for MA students in 2008. After these pilots, the format was ready for MA students in 2009, and the class has been running annually ever since. The class takes 12 students due to space limitations. Several instructors have taught it with a variety of backgrounds ranging from sociology to computer science and industrial design. The main constant feature in the class has been an electric engineer (Mikkonen). The class has had several customers, like the Nordic Innovation, Microsoft Cambridge and Nokia. The topics of the class have covered areas such as co-experience while driving, Internet search distributed to the physical environment, and honest signals in user interaction. So far, the class has led to several patents and a few start-ups, and at least two design consultancies build directly on its approach to design, one in Berlin, and another in Shanghai. In 2014, places for the class were reserved in ~~7~~.4 seconds after the application system opened. It is currently being taught in Berlin, with a focus on wearable technology in collaboration between Aalto University, Universität der Künste Berlin and Högskolan i Borås. By all measures, the class has been successful.

At the heart of the class is a dialect of C language and a variety of hardware components. The latter consist of sensors, microcontroller, and various sorts of actuators. In the class, the students have to identify a need for an interactive device, create a concept for it, build it, and evaluate it with a user test. In terms of teaching, the class begins with a brief and goes on to a short user-centered study with grounding on methodology. This part of followed by an introduction to the technology of the class, with both an introduction to programming and its grounding in hardware. The rest of the class goes through concept creation to construction and testing. Students spend time not only in the design studio, but also in the electronics lab, soldering components to the microcontroller, and debugging systems in case they do not work. What was left to the engineer were the truly difficult parts in programming, debugging, and testing the devices in cases something went wrong. His work, naturally, also covered maintaining the lab, finding and sourcing hardware components, and making sure every design was electronically safe.

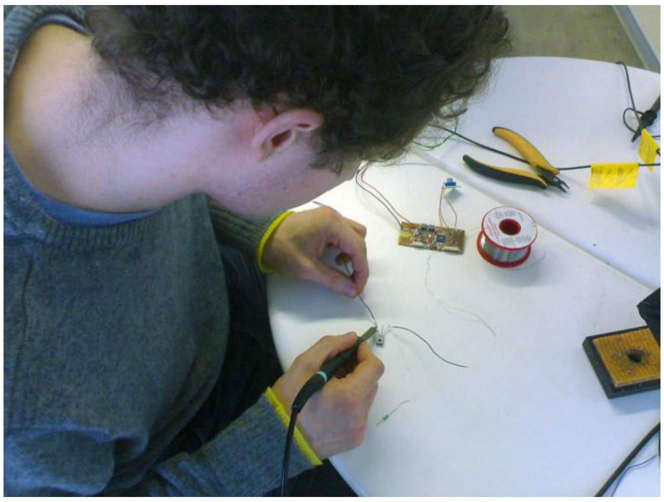

Figure 1. Soldering in the classroom

The class had many components unfamiliar to most industrial design students. In particular, C language is notoriously difficult to learn. Also, its connections to hardware through machine language are beyond the capabilities of but a few design students. Lacking expertise in electronics, it was suspected, would lead to a loss of focus in the classroom. The loss of focus would happen in overall terms, with students being unable to plan their use of time properly. It would, however, happen also in details, where new possibilities would either attract students too much, or rely on ill advice.

The pedagogy of the class was designed with this tension in mind. In pedagogical terms, the class was consistent with David Kolb's theory of constructive learning, but instead of his pragmatist philosophy building on Dewey, it built on ethnomethodology (Garfinkel 1967). From the latter, the class picked up two convictions: designers have their own methods of doing things; and these methods make things understandable and accountable to them. From these two premises follows a third: if a class builds on methods design students are not familiar with, they are puzzled, at loss, and feel that the class plays games on them. To make sure design students would focus on learning what the class wanted to teach, it was built at every step on design methods that were fitted into a design process.

Interactive Prototyping, then, became a balancing act between the familiar and the new. The central challenge of the pedagogy was to find a way to push as much newness to the students as possible without pushing them so far that the class would have become just a probe into technological possibilities. This challenge resulted in three central decisions about the class, all meant to keep students focused on their design.

3. DESIGN REDUNDANCY

The first decision was about the overall structure of the class. The aim was to build the class so that the students could plan their activities and workload without a second thought. Ethnomethodological thinking behind the class suggested that this was best to be done by building the class in the form of a design project.

This was done in three ways. First, there was the process. The class was constructive, and organized as a design project to make it more familiar with industrial design students. The project ran through briefing, research, learning electronics, concept creation, prototyping, and finally evaluation. Figure 2 shows the organization of the class in 2008.

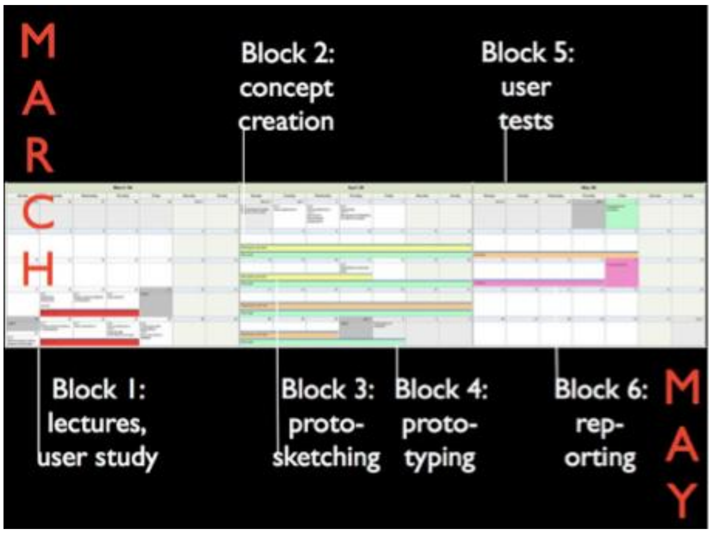

Figure 2: Building the class on design process

Second, the class was organized physically as a studio. This meant that although the main activity of the class took place in conversations and laptops, the students were collected together to studio spaces and workshops, and required to alternate between these. For instance, for the class of 2008, which worked on an issue related to cars, the instructors bought a car for the class and kept it in studio for the entire duration of the class. Overall, the class was to function as a design project, the aim being giving students a sense of control over their work, and a map. Third, the students were taught to work using methods familiar with them from other design classes. For example, instead of "analysis," the class talked about concept creation. For example, the early stage research methods built on cultural probes (Presence Project 2001), and immersive exercises with roots in inclusive design (Moore 1985; Herwig 2008) rather than ethnography, which was industrial gold standard at that time.

Figure 3 shows first a critique session in one of the open studio spaces of the university, and then a workshop, which was equipped with a car for the duration of the class. The reasoning for situating the class to these "experience-near" (a term by the psychoanalyst Heinz Kohut 1977: 302-303) environments came from ethnomethodology, but also from participatory design, which has stressed the need for building design on language games familiar to the people designed for (Ehn 1988).

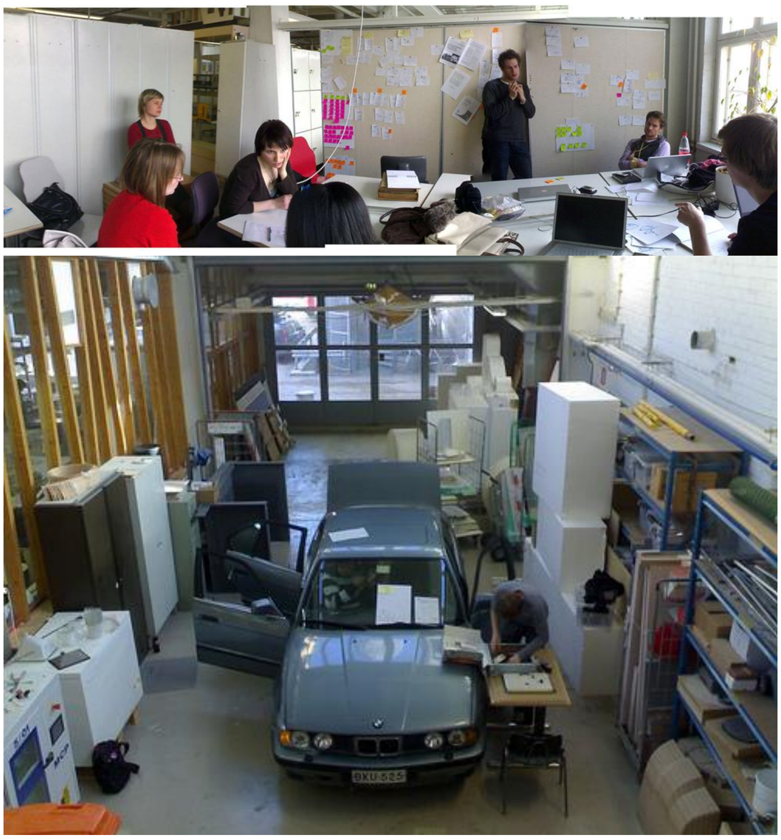

Figure 3: Snapshots of the main working environments of IP: studio and workshops

4. IMMERSIVE USER RESEARCH AS GLUE

Information technology poses several challenges to design students. They may fall in love with technology and go on to exploring details while losing the big picture of the aim of their exercise. Not having critical knowledge of their own, they may also be led astray by advise from well-meaning friends and colleagues on the Web. Building the class on a design process only gave students an overall direction, timeline, and aim, but did not help them to manage more specific design decisions. The class needed stronger glue that would give students a backrest to fall back upon when in doubt.

This glue was a snap user research early in the class. In the beginning of the class, students have to conduct a short user study and process its findings through design methods such as scenarios, use cases, and mock-ups. In later stages of the process, instructors always ask students to refer to user studies that come to function as the glue that keeps students in focus by giving a joint reference point to their conversations. The reasoning was that if these references were kept prominent in the minds of the students, they would function much better in the classroom than the then-prevailing methods like personas (Cooper 1999), which were too designer-led and confounded reality and designers' imaginations. The methods of user research have varied quite a bit over the years depending on the instructors, but the user research component early in the class has remained the same.

In the beginning of the class, instruction was relentlessly focused on this brief user research. Whenever students were in doubt about what to do, the instructors refused to give them advice other than the question "what does your fieldwork say." The tactic proved to be useful in achieving the dual aims of creating a team spirit, and "sealing" the design process from entropy that easily stems from working in an unfamiliar territory. Obviously, the tactic was not always successful; throughout the years, there have been student pairs and teams that have not been working well, for a variety of reasons not of interest here.

The way in which user research was conducted early on in the class, however, was different from the ways in which user research is typically taught in the classroom. As the aim of research was to use research as a tool for giving students a vivid experience of their design content, the class was not content with interviews and observations. Instead, we took our inspiration from universal design and told the users to develop ways to immerse into the world of people they designed for (the inspiration was Fulton Suri et al. 2005).

For example, in 2008, where the topic was co-experience while driving (specifically, how drivers interact with children in the backseat), the instructors sent the students out immediately after the brief (for co-experience, see Battarbee 2004). The students were told to find people who have children and who drive, and to describe their interactions with kids on the backseat. They were told not only to ask questions, however. They had to do the interviews *in the car* with both parents and children. While interviewing and observing, they had to ask people to sit in their positions, buckled up, and show situations in which they had been with the kids. Then the students were told to take the roles of the drivers and the children to gain a better grasp of the experiences from both sides. When students came back, they had a host of stories to tell, but they also had gained an insight into movements, distances, materials, and activities that define co-experience.

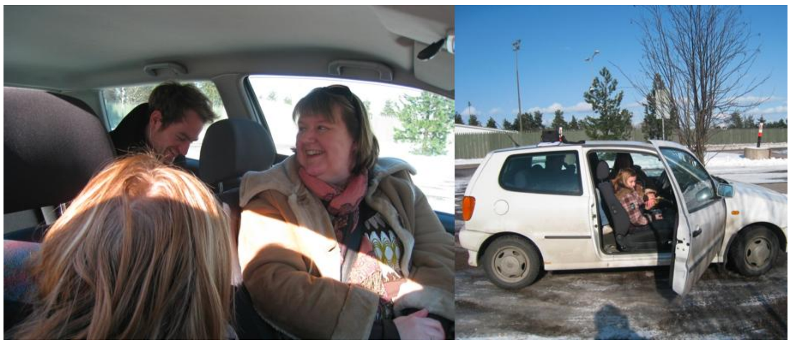
Figure 4: User research in Winter 2008

Immersive exercises continued throughout the class. When students were sketching their first interactive devices, for instance, the instructors told them not to work in talk and on paper only, but to use whatever ready-mades they could come up with to mock-up their concepts. Inspired by IDEO's work on bodystorming (Buchenau and Fulton Suri 2000).

5. INTO THE MIND OF THE MACHINE: FLOW CHART AS ANALYTIC

As the class ultimately relies on C programming language, notorious for its complexity, the instructors had to find another tool for learning to think in terms of the system they are conceiving. Again, the problem is one of focus: if students are unfamiliar with information technology, they easily get lost in the details of subprograms and subcomponents, and rely on invalid advice instead of building systems that are solid and safe. The problem for the class was how to find a method that would keep students focused on designing a system rather than getting lost in unproductive sideways.

The students are typically familiar with the creation of storyboards, however, which visually describe the use of the prototype, prompting the further use of visual means for the development of the more specific behavior of the prototype. Even though there are many ways to graphically describe an algorithm or functionality of a device (Yoder and Schrag, 1978; Newman, 1968; Lindsey, 1977; Resnick 2009), the structured flowcharts (Nassi and Shneiderman, 1973) have been shown to be very useful in the early stages of learning programming. They have been described as being very efficient in displaying functionality of an algorithm, as opposed to a pseudocode (Scanlan, 1989; Watts, 2004).

We have found out that the act of physically writing a card and attaching it to a wall helps the students slow down and think about the intended functionality, instead of seemingly achieving much by quickly writing or copy-pasting code, or ready-made structures, on a computer program. It also allows for the group of students to discuss and have a full visualization of the overall behavior, as the whole algorithm can be put up to a wall to be seen by everyone (Figure 5).

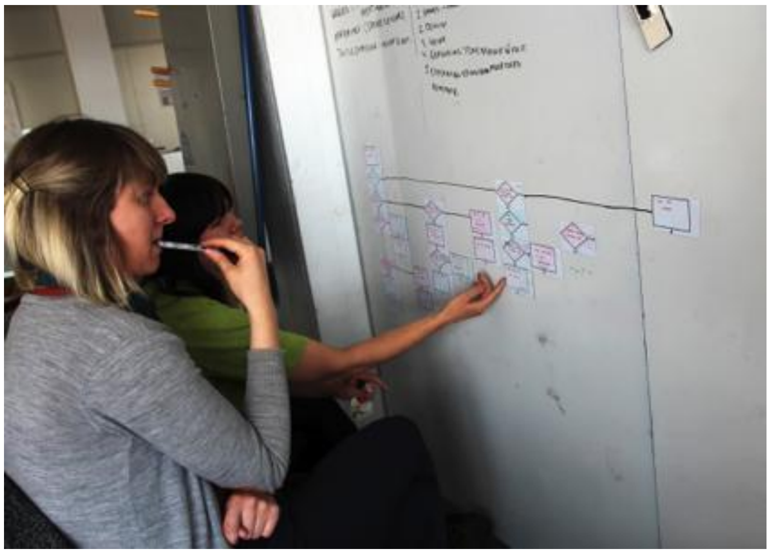

Figure 5. Working with the cards in 2013

The focus in using flowcharts is to improve the problem solving skills, and not focus on the syntax of the code. We have adapted them to introduce some ground rules for using flowchart symbols in a consistent and efficient manner, by creating and utilizing the 'Flowcards' (Mikkonen, 2012). Cards were created to help the design process and to simplify the prototyping, to provide means for generating code from the description of the functionality. The cards have been in their final form since 2012.

Similarly, the usage of block diagrams has helped the students modularize the mental model of the prototype and thus help with the creation of the code. By being able to define the actual physical components, and start by building "driver" algorithms for them, has eased the access to otherwise complex hardware. Sometimes the class engineer has helped with the access to more complex sensors and actuators, but since the introduction of Arduino in 2009 and the use of Flowcards, this support has provided more fruitful results with less input.

One of the critical points in the development of a prototype is when the point of view changes from the users to the prototypes. This typically happens when the components that make up the prototype are already tested and known, well after the first version of the storyboard has been finalized. The students would write the key aspects of the storyboard to individual cards, and organize them to follow an optimal use case from the users point of view. By further arranging the cards to functional modes or similarities, they would be then changed to the machine point of view. This is a complex issue during the early stages of learning: the processor needs to be told exactly what kind of actions to take and respond to.

The implementation typically starts with the selection of sensors and actuators for the concept. During this time, the storyboard is finalized. The process continues with the individual testing of the sensors and components, resulting in a "taped-to-a-table" prototype. With this rough "sketch" of a prototype, the basic functionalities, i.e. reading sensors and writing to actuators, are implemented. Once those work, the actual prototype behavior is coded. This starts with the use of cards, to generate flowchart, which is then changed to a working code. Once the prototype performs satisfactorily in a table, the final version is built.

6. DISCUSSION

This paper has described the pedagogy of Interactive Prototyping, a class run in the industrial design program at the former University of Art and Design Helsinki (now Aalto University). The class first saw daylight in 2007, and has been running at the master’s level since 2008 until today. Its aim has been to teach industrial design students prototyping skills for interactive devices. The challenge is that few industrial designers have background in electronics and programming, which meant that the pedagogy of the class had to be built on methods familiar to designers rather than on the science of electronics. Inspired by ethnomethodology (Garfinkel 1967) and constructive education (Kolb 1984),[1] the class was built to make sure students could have a sense of control of their learning process, as unfamiliar as some of the contents were to them.

The paper has described three of the methods the class has used to guarantee a smooth learning experience. The class, first, is built on a design process. This gives students a sense of direction and control, and a way to schedule their work in the class. The class, second,

[1] With Kees Dorst’s work on expertise (Dorst 2003), constructive pedagogy was also one of the foundations of design education at the Technical University of Eindhoven, which was one of the inspirations of Interactive Prototyping. The management of TU/Eindhoven had visited Helsinki in 2002 in a workshop organized with Jane Fulton Suri (see Fulton Suri et al. 2005), and most likely took inspiration for its education from it.

has been built on a brief user study. This user study has been design-specific in that it has gone beyond interviews and observations, and has used a set of immersive techniques throughout the years. User study gives students references specific to the topic that function as glue that keeps the design process in focus. The third thing the class has introduced targets the hardest part of the class, learning programming the microcontroller with C language. Here, the problem was to find a way to teach students how to think from the perspective of an electronic system. The engineer running the class developed a flowcharting method to address this end. With it, students were able to construct a logical basis for the system they were designing, and keep its main structure in focus instead of getting lost in the details of subprograms and subcomponents.

The results of the class have been consistently good. It has led to a few start-ups, a few design companies build their methodology on the class, and it has produced solid, well-reputed researchers. It has also produced IPR for students and the university, and it has brought in customer companies of solid reputation, like Microsoft, VTI technologies, and Nokia. The class has also become one of the most sought-after classes in Aalto University – seats to the class being booked on average in less than 10 after the admission system opens annually.

REFERENCES:

Alexander, Christopher (1971) The state of the art in design methods. *DMG Newsletter*, *5*(3), 1–7.

Battarbee, Katja (2004) *Co-experience. User experience in interaction*. Helsinki: UIAH.

Buchenau, Marion and Jane Fulton Suri (2000) Experience prototyping. In: *Proceedings of DIS* (pp. 424–433). ACM.

Cooper, Adam (1999) The Inmates are Running the Asylum: Why high tech products drive us crazy and how to restore the sanity. Indianapolis, IN: Sams Publishing.

Dorst, Kees 2003. *Understanding Design.* Amsterdam: BIS.

Ehn, Pelle (1988) Work-oriented design of computer artifacts. Stockholm: Arbetslivscentrum.

Frens, J. (2006) *Designing for Rich Interaction. Integrating Form, Interaction, and Function*. Eindhoven, the Netherlands: Department of Industrial Design.

Fulton Suri, Jane (2011) Poetic observation: What designers make of what they see. In A. Clarke (Ed.), *Design anthropology*. Springer: Vienna.

Fulton Suri, Jane, Katja Battarbee and Ilpo Koskinen (2005) Designing in the dark: Empathic exercises to inspire design for our non-visual senses. In *Proceesings of include 05*, Royal College of Art, London, England.

Garfinkel, Harold (1967) *Studies in ethnomethodology*. Englewood Cliffs, NJ: Prentice-Hall.

Herwig, Oliver (2008) *Universal design: solutions for a barrier-free living*. Basel: Birkhäuser.

Jones, J. Christopher (1977) How my Thoughts about Design Methods have Changes During the Years. Design Methods and Theories 11(1): pp. 50-62.

Kohut, Heinz (1977) *The Restoration of the Self*. Madison: International Universities Press, Inc.

Kolb, David (1984) Experiential Learning. Experience as the Source of Learning and Development. Prentice-Hall, Englewood Clifs, NJ.

Koskinen, Ilpo, Katja Battarbee and Tuuli Mattelmäki (eds.) (2003) *Empathic design*. Helsinki: IT Press.

Lindsey, C. H. (1977) Structure charts a structured alternative to flowcharts. SIGPLAN Not. 12, 11 (Nov. 1977), 36-49. DOI= http://doi.acm.org/10.1145/956641.956645

Maldonado, Tomas (1984) Ulm revisited. In *Rassegna, Anno 19/3*, settembre 1984. (Frank Sparado, Trans.)

McIntyre, Jean (1995) The Department of Design Research at the Royal College of Art: Its Origins and Legacy 1959-1988. In Frayling, Christopher and Claire Catterall (eds.) *Design of the Times: One Hundred Years of the Royal College of Art.* London: Richard Dennis Publications/Royal College of Art.

Moore, Patricia with Charles Paul Conn (1985) *Disguised: A True Story*. W Publishing Group.

Nassi, Isaac and Ben Shneiderman (1973) Flowchart techniques for structured programming. SIGPLAN Not. 8, 8 (Aug. 1973), 12-26. DOI= http://doi.acm.org/10.1145/953349.953350]

Newman, William M. (1968) A system for interactive graphical programming. In Proceedings of the April 30--May 2, 1968, Spring Joint Computer Conference (Atlantic City, New Jersey, April 30 - May 02, 1968). AFIPS '68 (Spring). ACM, New York, NY, 47-54. DOI= http://doi.acm.org/10.1145/1468075.1468083

Presence Project (2001) London, RCA: CRD Research Publications.

Resnick Mitchel, John Maloney, Andres Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai (2009) Scratch: programming for all. *Commun. ACM* 52, 11 (November 2009), 60-67. DOI=10.1145/1592761.1592779 http://doi.acm.org/10.1145/1592761.1592779

Scanlan, David A. (1989) Structured Flowcharts Outperform Pseudocode: An Experimental Comparison. IEEE Softw. 6, 5 (September 1989), 28-36. DOI=10.1109/52.35587 http://dx.doi.org/10.1109/52.35587

Watts, Tia (2004) The SFC editor a graphical tool for algorithm development. J. Comput. Small Coll. 20, 2, 73-85.

Yoder, Cornelia M. and Marilyn L. Schrag (1978). Nassi-Shneiderman charts an alternative to flowcharts for design. SIGMETRICS Perform. Eval. Rev. 7, 3-4 (Nov. 1978), 79-86. DOI= http://doi.acm.org/10.1145/1007775.811104