Mikkonen, Jussi

On The Teachers Role in Interactive Prototyping

Published in:
Design for Next

DOI:
10.1080/14606925.2017.1352651

Published: 14/04/2017

Document Version
Publisher's PDF, also known as Version of record

Routledge
Taylor & Francis Group

The Design Journal
An International Journal for All Aspects of Design

ISSN: 1460-6925 (Print) 1756-3062 (Online) Journal homepage: http://www.tandfonline.com/loi/rfdj20

On The Teachers Role in Interactive Prototyping

Jussi Mikkonen

To cite this article: Jussi Mikkonen (2017) On The Teachers Role in Interactive Prototyping, The Design Journal, 20:sup1, S1212-S1223, DOI: 10.1080/14606925.2017.1352651

To link to this article: http://dx.doi.org/10.1080/14606925.2017.1352651

Published online: 06 Sep 2017.

DESIGN FOR NEXT
EAD12-ROME

Design for Next

12th EAD Conference
Sapienza University of Rome
12-14 April 2017

doi: 10.1080/14606925.2017.1352651

On The Teachers Role in Interactive Prototyping.

Jussi Mikkonen*

Aalto University, School of Arts, Design and Architecture
*Corresponding author e-mail: jussi.mikkonen@aalto.fi

Abstract: The industrial design students are increasingly involved with the development of functional interactable prototypes. There is complexity in both the concept generation and the implementation, something that the design students need to be able to cope with. We approach the prototyping in the light of representing a design idea, moving from concept towards a physical manifestation, and discuss the concept-imposed minimum towards the complexity of a prototype. While a combination of methods, such as cardboard-mockups, and wizard-of-oz can be used to evaluate simpler design ideas, they simply fall short with a complex device, as there are too many things happening. Thus, an approach is needed for developing complex prototypes. We examine five different prototypes, and discuss the role of the teacher in each. Drawing from the concept, prototyping practice, and the required complexity, we suggest stages of complexity for the development, and a more active participation from the teacher.

Keywords: interaction design, prototyping, complexity, education, teacher

1. Introduction

The role of the industrial designer is rapidly changing, having far evolved from the tradition of form giving. This paper discusses the development of physical prototypes in interaction design education, with a focus on the development of functional prototypes. Our goal is to develop students, who are able to communicate and operate in a multidisciplinary setting. Prototyping tools such as Arduino have become common, and the students are tasked with the development of prototypes, instead of just concepts. Prototyping as a practice in design is well known (Koskinen et al., 2011), in both teaching and in research. In their research, Wensveen and Matthews (Wensveen & Matthews, 2015) describe four roles for a prototype in design research context, derived through an analysis of several different prototypes. The first role is an experimental component, where the prototype is an integral part of an experiment towards testing specific hypotheses. The second role is as means of inquiry, where a prototype is used for open-ended inquiry, followed by a research archetype, useable as illustration or demonstration. In the fourth role, the prototype is used as a vehicle for an inquiry, where it drives the research direction.

A device being developed can be described with varied levels of detail, and explored through different complexities of implementation. The most rudimental difference is between “a prototype” and “a sketch”, as explained by Buxton (Buxton, 1998). Both are representations of a design idea, however a prototype is something which has been invested in, and a sketch can be easily discarded, as it is merely a suggestion for a representation of an idea.

There are several methods for the generative phase to help understand the issues and try to extract the knowledge in what may be an unknown and a complex situation. Experience prototyping (Buchenau & Suri, 2007) draws from using ones own bodily intuition to distill ideas, where we see the body as a filter for gaining information about the act of use, or the situation of use. On the other end of the scale, participatory design (Schuler & Namioka, eds., 1993) uses the stakeholders as a direct access to the information, to provide experience-based knowledge which could be otherwise difficult, or even impossible, to gain. Thus, these methods can be seen as bringing simplicity to the concept generation, as the information is already processed by either own experience or through others.

Hummels has explored prototyping through design concepts and prototypes at various levels (Hummels, 2000). There was a strong emphasis on division into low-fidelity prototypes, high-fidelity prototypes and working prototypes. Low-fidelity prototypes are more prevalent in the early stages of product development, while high-fidelity prototypes appear towards the end. They are also clearly separated from a spatial model, which is a physical model of the design concept or sketch. A working prototype is a fully functioning design idea, as Hummels explains: “A fully functioning prototype with the desired appearance approaches the experience of the product made and evokes a different experience than a low-fidelity cardboard mockup made quickly to test the interaction”.

How should the implementation then be approached? There are different levels of prototypical functionality, which can be used for the evaluation. For the low-end prototypes, there are e.g. paper prototypes (Ehn & Kyng, 1991) and non-functional mock-ups. These are suitable for exploring and communicating the basic idea, using imagination to fill in the gaps, in order to support the further development of the prototype. In his thesis, Simo Säde (Säde, 2001) described a cardboard mockup of a prototype. It was a three-dimensional mockup made from paper, building on top of paper prototypes (Ehn & Kyng, 1991). Taking a similar approach, a paper mockup was made, with the designer acting as the ‘machine’ behind the mockup. It was intended for user testing, without too expensive investment in materials or the development of technology towards what appears otherwise as a working prototype. This kind of prototype does not need technical expertise to achieve a representation of the final product. This is similar to creating the effect of a working prototype without actually having to program the intended behaviour, known as wizard of oz (Dahlbäck, Jönsson & Ahrenberg, 1993). However, it becomes increasingly difficult to create a fluid experience for the evaluation of complex systems: there are simply too many things happening at the same time.

While these are all useful approaches for generating ideas and concepts with the intent of creating a prototype, and evaluating the intended functionality, there is a gap in bridging the technology and the concept in the design education. As there are different levels of complexity for concepts, there appears to be a concept-imposed minimum towards the complexity of the intended prototype. We first approach the complexity towards the design education setting. Then, we examine the role of the teacher through interactive prototypes of different technological levels, and propose approaches for achieving higher level of complexity for prototypes in design education. Finally, we suggest a “stages of complexity” -approach towards developing complex prototypes.

2. Problem of Complexity

2.1 On Complexity

Norman discusses the complexity from a wide perspective, discussing different approaches through conceptual models and practical situations (Norman, 2010). One key aspect raised is the use of signifiers, i.e. how a complex situation can benefit from cues, indicating the state of the system, or suggestions towards the recommended action. The requirement engineers are trying to address complexity, with a problem being the management of the implementation of a large or otherwise complex embedded system (Sikora, Tenbergen & Pohl, 2012). In our opinion, both Norman and Sikora essentially describe the need for understanding and managing the information, where the focus is on how that information is represented. How, then, can a student be trained to work in a multidisciplinary team, and what is the level of complexity they need to be able to manage independently?

What do we mean by complexity? Consider the difference between Scratch (Resnick et al.2009) and C (Kernighan & Ritchie, 1988). Both can be considered programming languages, with almost identical basic operators. However, the implementations are vastly different. Scratch offers ready-made IDE with graphical programming, designed to minimise possibilities for syntactical and logical errors related to the language. C, on the other hand, is all manual. Even a simplest typo with one character can change the resulting behaviour, or simply just make the code invalid, uncompilable. The mental load is different, as a person coding with C needs to manage both the language, i.e. remember the syntax, variables and even the visual structure of the code, as well as the concept functionality. Scratch hides most of this complexity, allowing the user to focus only on the concept-related behaviour. As Scratch is similar to structured flowcharts, which have been proven to be beneficial for learning programming (Crews, & Ziegler, 1998)(Watts, 2004), it is a good example for managing the initial difficulties.

The complexity, as discussed here, is defined as *the apparent difficulty of managing the connected hardware, towards the functionality implied by the concept*. This can be divided to the electronic components required for the functionality of the concept, the firmware required for managing such electrical connections, and the operational software, which binds together the aforementioned to the behaviour as described by the concept. A fourth element could be argued to be the non-locality, such as devices being remote controlled by the prototype, or when the prototype behaviour would be influenced by information or interactions from the internet, or other non-local, intangible source. Thus, the minimum complexity attempts to describe the most straightforward resources required by a concept, which, consequently, begin to set real-world limits to the prototype implementation. We define the simplest resource as being the easiest to play around and understand, and acknowledge, that it depends on the person using the resource.

2.2 The Complexity of a Prototype is not Absolute

To extrapolate how the complexity depends on the person, we look at an imaginary example. In our concept, a capacitive touch-slider is used incrementally, i.e. not by absolute values, but by the direction of the movement. The slider can be implemented by having e.g. two parallel conductive triangles with width inversely proportional to the distance from the slider ends, or by having a minimum of three discrete touch areas in a sequence. Both implementations can be implemented in at least two ways in hardware: by building the capacitance measurement circuit directly to the processor pins, using components and considerable coding, or by using IC:s that measure the touch independently. The IC:s can be connected to the processor in different ways, such as e.g. by utilising

a communication bus, or by the IC(:s) sending direct signals, one for each touch-area in the slider structure.

Even for such a basic element for interaction, it is difficult to see what is the simplest, as people interpret them through their own experiences. A person who has programming experience, but knows little electronics, might pick the one with a communication bus, as most parameters can be controlled through the communication bus. A design-student might not care about the communication bus IC:s, if there are no libraries available for the communication protocol, and prefer the ones with a direct outputs. Someone with electronics background might prefer building their own capacitive circuit, as it can be done with just two resistors, requiring some low-level coding. It seems evident that the complexity is not absolute, as the simplest solution depends on the person and the intent of the prototype.

Thus, the minimum complexity regarding the component (sensor, actuator, etc.) depends on the ability of a given student to play around with. On the other end, as a concept will set the outline for the prototype behaviour, it thus dictates the extent of *the developer*-influenced requirements for the implementable functionality (as a separate from the user-influenced). As an example, Joep Frens simplified a user interface with rich interactions (Frens, 2006), where the mental model of digital camera was enhanced through physical actions supporting the use. He developed cardboard mockups for the evaluation, and then created the final functional prototype with the help of studio personnel, who were experts in the construction methods. His work shows simplification in two aspects of prototyping: in the simplified process, where he did't have to focus on learning metalworking, and the simplification of use (mental model of the prototype), by providing rich interaction cues. Therefore, the complexity can be in the implementation, in the concept, or the combination of both, i.e. the prototype.

The minimum implementation will then depend on the “critical aspect” of the prototype being evaluated, i.e. what is the novelty, or the core idea, under development. Should the prototype focus on new way to charge a mobile device, then e.g. the actual functionality could be provided as ready-made, so as to focus only on the act of charging. On the other hand, if the prototypes under development would focus on a new interface to control music, the students could be provided with the underlying technology, such as amplifiers and mp3-players, and the development could focus on the actual user interface. Therefore, it is necessary to look at prototypes and how this has been achieved.

3. Prototypes

All of the prototypes presented here are created using user-centred design process, each prototype having been developed towards a situation that is well known. The needs have been identified through interviews, on-site excursions and observations, if the developers themselves were not at the core user group. The functionality was developed using Flowcards-tool (Mikkonen, 2012). The first two weeks of the course were spent on learning the basics of programming and physical prototyping. The background work and concept development were during the first three weeks, overlapping with the technical studies. The next five weeks were spent on the development of the prototypes. The teacher was available during all hours, and the course was implemented as full-time study.

3.1 Magic Cabinet

The prototype was a cabinet, with gesture-controlled transparency on the glass-doors, intended for speeding up searching through the cabinet. Each door had a separately controlled glass, controlled with an individual transformer. The transparency-control transformer-units were on/off controlled with an Arduino, which also interfaced to the IR-LEDs, and an IR-receiver. This enabled a simple gesture detection, based on the blocked signals. Thus, if an IR-signal was missing, the source would be known immediately, turning the corresponding door transparent. If the user did the gesture over any two doors simultaneously, all four would become transparent. The cabinet is shown in Figure 1., with a functional early prototype, and with the use of the cabinet-version.

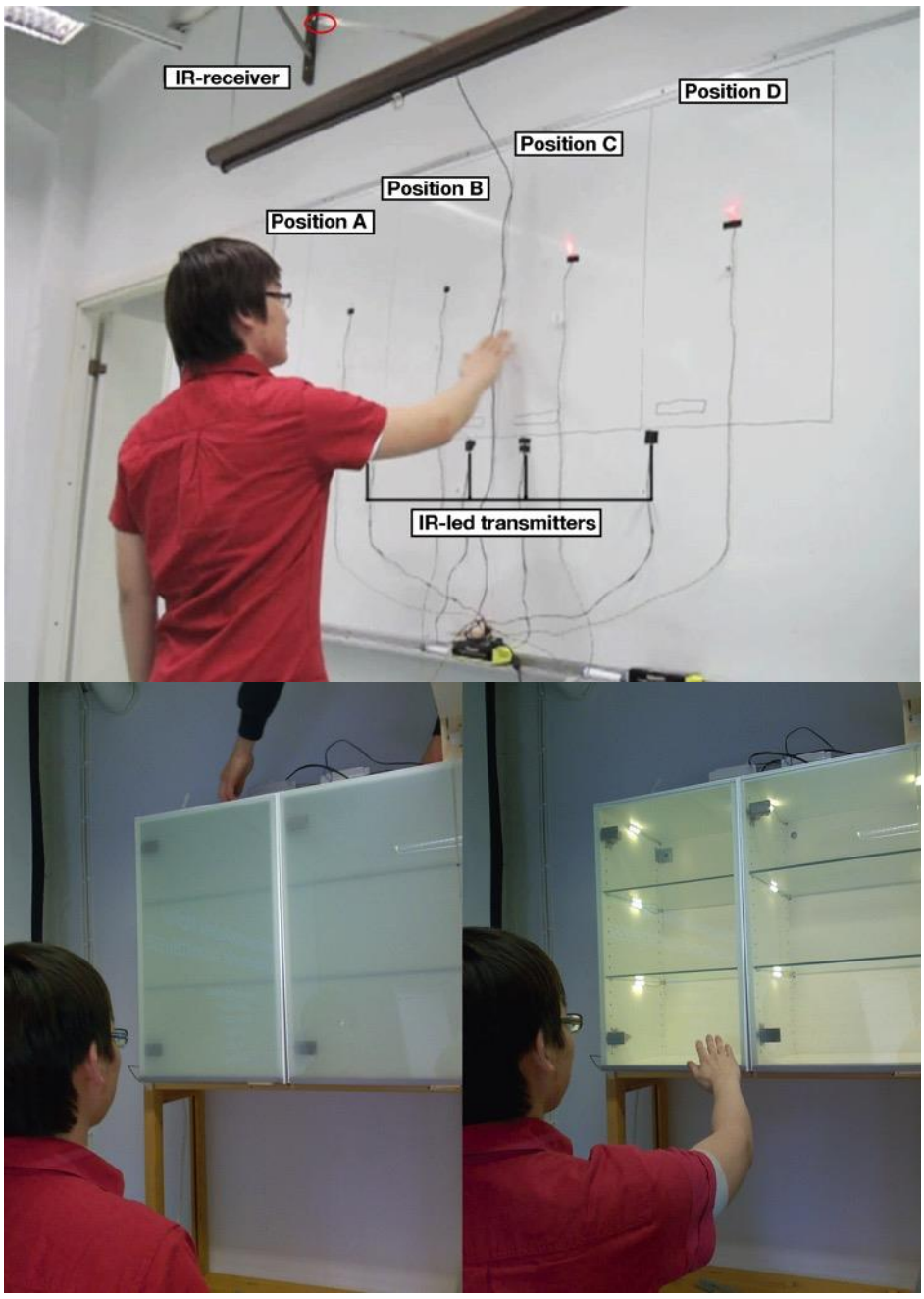

Figure 1. Magic cabinet functional mock-up (t), inactive prototype (bl), and activated prototype (br).

To focus on the interaction, the system was first developed without the glass-doors. The system was simplified with one red LED acting as a door, and reproduced for four doors. The setup was wired for

a classroom whiteboard, where a sketch of the cabinet was drawn. This facilitated testing with the sensor/receiver locations, as well as coding and debugging. Once the interactions were satisfactory, the final version was built. In addition to the aforementioned, final cabinet consisted of four LCD-laminated glass-doors, and lights for each shelf. The teacher had the biggest influence on the selection of the transformer-units for the doors, by directly selecting an interface supporting the development using the red LEDs. Otherwise, the discussions focused mainly on the verification of the students work.

3.2 Colour Connection

Figure 2. Colour Connection used in a test situation, placed so as to identify faces

The Colour Connection-prototypes were designed for an airport restaurant or food-bar, to help identify those who want to communicate, as well as to keep track of the food and the flight status. Each table has a computer with a camera, connected to an inside projector and an RFID-reader. The tickets contained an RFID-tag, which were associated with simulated flight-data and food service. The status of the food and flight were projected to the table surface from below, when the ticket was kept on top of the table. The tables were placed so, that the face-detecting cameras would primarily look at the other tables, shown in Figure 2. If faces were detected looking towards the user, and if you would also look at the same table, your table would slowly turn red to indicate common interest. It would also show if you were on the same flight.

The students were independent during the prototyping-phase, as they immediately decided which functionalities they wanted to use. After going through the examples of face detection, RFID-tag reading and communication with the teacher, further help was not needed. Since the OpenCV-library used for the face detection gave coordinates and face-sizes, the students were able to independently figure out the expected parameters from a person sitting at a specific table. The students explored and tested with numerical values, associating them with physically represented parameters, effects of which could be easily visualised.

3.3 Honest Shoes

Figure 3. Honest Shoes in a test situation.

The Honest Shoes, shown in Figure 3., were developed to help people become better presenters, with the concept stemming from research evaluating different postures and how people fidget their feet while presenting. The shoes use acceleration sensors to detect fidgeting, as well as force sensitive resistors on one foot to help differentiate walking during presentations. Both shoes were built using Arduino FIO, and communicate wirelessly. They were developed in three phases, starting with the evaluation of honest signals (Pentland, 2008) of nervousness, and then making a first simple prototype: it was used for recording user fidgeting, as well as to later develop the first version of a detection algorithm. The second prototype of quality leather shoes was built based on the first, and includes the sensors for detecting walking.

The students were fairly independent, building on top of the previous phases. The simplification of data communication, as well as how to temporally (i.e. to scale time) utilise the multi-dimensional acceleration data for detecting fidgeting, required more teacher input. Interestingly, the most difficult idea to implement was sending the data wirelessly. In the end, the teacher explained directly that the detected state from another shoe would not require complex coding, and could be represented in, and transmitted to, the primary shoe with only one variable.

3.4 BeoSound Orbit

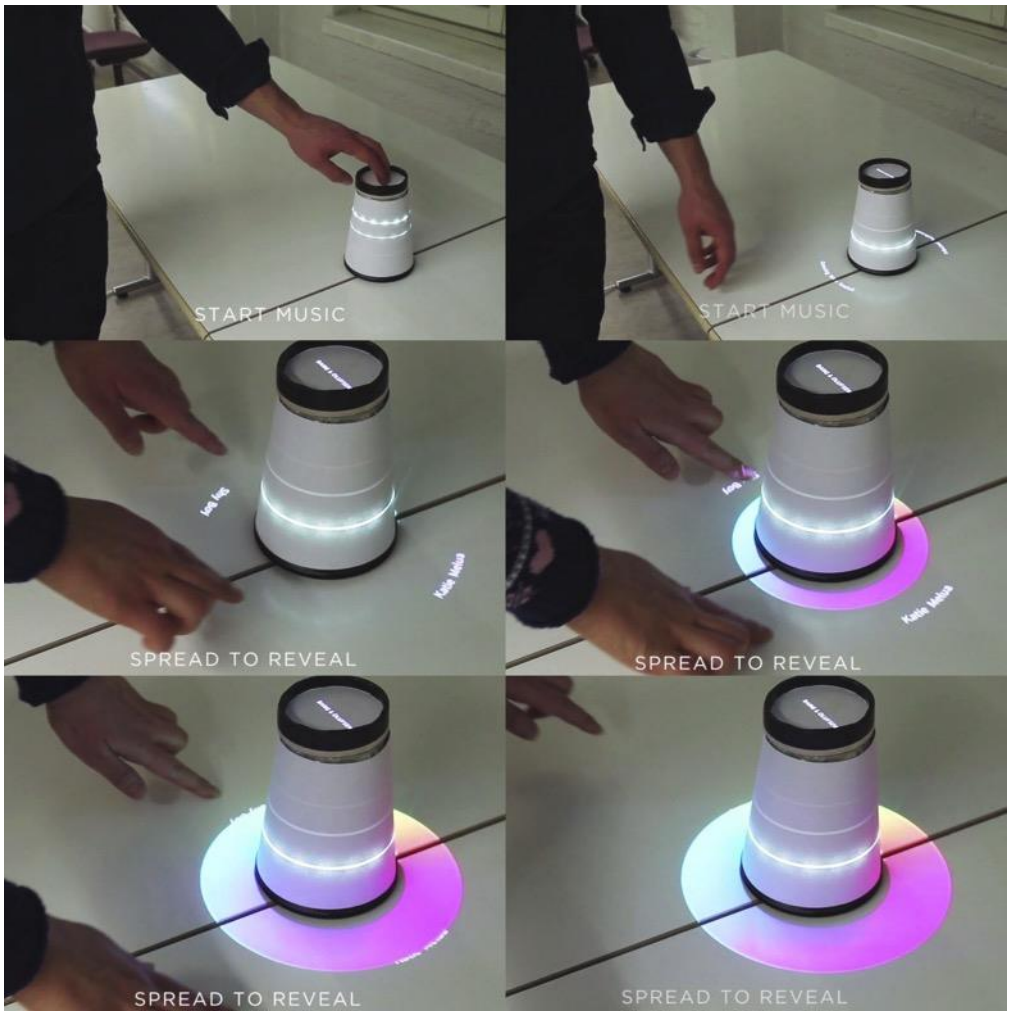

Figure 4. BeoSound Orbit usage, top-projection is used to create a user interface, which is combined with the physical interface at the top of the UI-device.

The BeoSound Orbit is a multi-modal user interface for controlling music streamed from spotify. It uses top-projection and IR-camera for augmenting the physical user interface, which has rotational free-rolling volume control and on-off -functionality built to the cardboard top ring. The paper-cone contains LEDs for both visual feedback, as well as for illuminating objects at the table surface-level using infrared. The key elements of the user interface are visible in the Figure 4. By touching, sweeping and pulling or pushing with fingers, user can interact with the spotify content. The spotify-connectivity, projection and camera-analysis are handled by a PC, using ready-made libraries. The LEDs and the physical user-interface components are managed through a connected Arduino.

The students were highly independent; however there were several discussions on how to implement and augment the tabletop touch detection. Therefore the teacher suggested improvements to the technical (physical) signal qualities, which the students then explored. The teacher directly suggested how to combine the data from the mechanical interface to the core software on PC, as the overall system was becoming complex to manage by the students.

3.5 MealWeaver

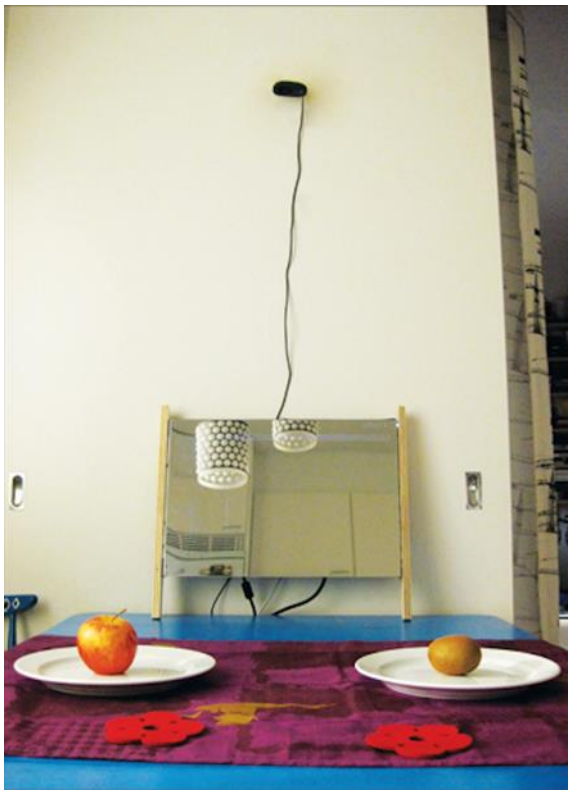

Figure 5. MealWeaver-setup in a test situation, using a webcam as the input, and the semi-transparent mirror as the output.

MealWeaver, shown in Figure 5., is a device for recording the colours of the food being eaten, and creating abstract “woven” pattern using it. It consists of a button on Arduino connected to a PC, and a webcam for detecting food on top of white plates. The idea was to help couples decide on what food to prepare, by showing the eating history as an abstract pattern consisting of colours in the food. The colours would be extracted by searching for white circles, and a patch would be formed using the colours (food) inside the circle, size determined by the eating duration. The user interface consisted of a semi-transparent mirror display, and a button, which was pressed to indicate the beginning and the end of eating. The students were required to describe the behaviour in much more detail, with the teacher acting similar to a subcontracted engineer.

4. Discussion

4.1 On the role of the teacher

The prototyping has involved the teacher in different roles: as a guide, an engineer, and a work manager. Regardless of the role, the teacher tried to always arrange time and be approachable. In all cases, the teacher would also independently think of at least one way to implement the student concept. In all cases, the teacher provided them early on with basic methods on managing the electronics and to the program the behaviour, and during the prototyping phase, acted as a discussive guide.

The teacher was relying on the engineering knowledge in all cases, however with the MealWeaver, the role was very direct: coding as an "outside engineer", the students were required to give very detailed specifications for the behaviour. The teacher would also discuss the intermediate results,

how and why they were obtained. With BeoSound Orbit, the engineering knowledge was needed to modify the camera for IR-detection, and combine that with the correct IR-LED:s. With the cabinet, the implications of different movement and touch detecting principles were discussed, due to the high voltage present in the glass-doors.

Even though mentioned only in the shoes, one critical aspect in all systems was the problem with time-scale differences. The prototypes operate at a very high speed, capable of millions of decisions per second. Scaling this to the user level has been such a problem, that the teacher has already created a set of libraries for managing them. It could be said that the teacher should be able to suggest methods, and approaches to implementations, and be prepared to actively go beyond basic setup. As with the capacitive slider earlier, each option has different benefits and downsides, and the technical aspects should be made easy to ignore: the teacher should be able to provide means for trivialising the usage, when necessary, so that the student can focus on the actual concept.

With the exception of the MealWeaver and the Colour Connection, the teacher has acted as an invisible work manager, keeping up with the prototype developments, having suggestions ready when needed. On the other hand, typical educational goals, requiring the students to give presentations at fixed times, setting schedules for ordering components etc., deadlines are being set. This is in effect very explicit work management, which can help the students considerably. By combining the deadlines with the engineering knowledge, i.e. times required for implementation, the teacher can discreetly manage the students work, suggesting changes, providing libraries, if things do not progress.

4.2 The Stages of Complexity

Complexity comes from the inability to manage the whole, requiring abstraction and simplification. The initial approach to the prototyping is critical, and keeping up the progress needs consideration. Even though the concept would be complex, it can be divided to modules and parts, an implementation approach suggested by Sikora et al. as well (Sikora, Tenbergen & Pohl, 2012). However, a complex implementation would imply less attention to the details through abstraction, it is necessary to understand the details. Simple prototypes can be addressed through teaching basics of interaction elements. Complex prototypes basically require either libraries or pre-made interfaces, requiring considerable preparation and reaction to the student progress. The first two weeks spent in learning the basics address the fears of implementation, and show that independent development is possible, helping kickstart the initial motivation.

As soon as the students have their concepts, identifying and separating key functionality is important. This sets the minimum complexity through the intent of the concept, and the identification of key components can be started. The students should be able to explain the purpose, in order to look for different ways, i.e. components with different functionality, to achieve it. The students would then approach the components, with the intent of learning to understand the implications, by playing with them to get first hand experience. Ideally, there would be several different options to explore: for example, touch could be detected using capacitive components, but also with pressure-sensitive components, such as force-sensitive resistors. Such sensors are electrically simple, but may require flexible casing.

The concept can be split to targets, which can be achieved through successive work. With the exception of MealWeaver and Magic cabinet, each prototype had several functional elements, augmenting the core concept. With BeoSound Orbit, the external casing was made from paper, as there was no time to finalise the casing. Thus, to develop the essence of the interaction concept, and then build additional goals keeps up the motivation, help deal with the complexity in incremental

steps. However, the key is in the teacher reactions and adaptability towards student progress, creating a prototyping-positive atmosphere.

5. Conclusions

We have discussed the role of the prototype and complexity, presented several prototypes, and how to manage prototyping in interaction design education. The teacher's role is critical - but how much of the work should rely on the teacher, and should the line be drawn on the lecture material? When the teacher independently develops at least one way to implement a prototype, it will also help in defining the minimum complexity. To address the complexity faced by the students, we argue for expanding the role of the teacher, to take a more active role in the development, augmented by prototyping through different stages of implementational complexity.

References

Buchenau, M., & Suri, J.F. (2000). Experience Prototyping. In ,. D. Boyarski and W.A. Kellogg (Eds.), Proceedings of the Third Conference on Designing interactive Systems: Processes, Practices, Methods, and Techniques (DIS '00), 17-19 August 2000, New York City. New York: ACM. 424-433

Buxton, B. (2007). Sketching User Experiences: Getting the Design Right and the Right Design. Morgan Kaufmann

Crews, T., & Ziegler, U. (1998). The Flowchart Interpreter for Introductory Programming Courses. Twenty-Eighth Annual Frontiers in Education Conference 04-07 November 1998 (FIE '98) 307-312

Dahlbäck, N., Jönsson, A., & Ahrenberg, L. (1993). Wizard of Oz Studies: Why and How. In Proceedings of the 1st International Conference on Intelligent User Interfaces (pp. 193–200). New York, NY, USA: ACM.

Ehn, P., & Kyng, M. (1991) Cardboard Computers: Mocking-it-up or Hands-on the Future. In Greenbaum, J., & Kyng, M. (Eds.) Design at Work: Cooperative Design of Computer Systems. Hillsdale, N.J.: Lawrence Erlbaum Associates. pp. 169-196

Frens, J. W. (2006). Designing for rich interaction: Integrating form, interaction and function. Eindhoven, the Netherlands: Eindhoven University of Technology.

Hummels, C. (2000). Gestural Design Tools: Prototypes, Experiments and Scenarios. Delft: Delft University of Technology

Kernighan, B.W., & Ritchie, D.M. (1988). The C Programming Language (2nd ed.). Englewood Cliffs, NJ: Prentice Hall. ISBN 0-13-110362-8.

Koskinen I. et al. (2011) *Design Research Through Practice: From the Lab, Field, and Showroom*. Waltham, MA: Morgan Kaufmann/Elsevier, 2011

Mikkonen, J. (2012). Flowcards - a communication tool, *Proceedings of Design Research Society Conference 2012*, Bangkok, Chulalongkorn University, Bangkok, Thailand, 1-4 July 2012

Norman, D. A. (2010). *Living with complexity*. MIT press.

Pentland, A. (2008). *Honest Signals: How They Shape Our World*. The MIT Press.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all. *Commun. ACM*, 52(11): 60-67.

Schuler, D., & Namioka, A. (Eds.). (1993). *Participatory design: Principles and practices*. CRC Press.

Sikora, E., Tenbergen, B., & Pohl, K. (2012). Industry needs and research directions in requirements engineering for embedded systems. Requirements Engineering, 17(1), 57–78.

Säde, S. (2001). Cardboard Mock-ups and Conversations, PhD thesis, UIAH

Wensveen, S. & Matthews, B. (2015). Prototypes and prototyping in design research. In Paul. A. Rodgers and Joyce Yee(Ed.), Routledge Companion to Design Research(pp.262-276) London, United Kingdom: Routledge.

Watts, T. (2004). The SFC Editor a Graphical Tool for Algorithm Development. J. Comput. Small Coll., 20(2): 73- 85

About the Author:

Mikkonen Jussi, D.Sc. (Tech.) Having worked and taught at both University of Technology, as well as Arts and Design, he has developed a positive perspective towards multidisciplinary prototyping. His doctoral work focused on prototyping interactions, drawing from both design research and electrical engineering. His research interests are on IoT, smart clothing and -textiles, physical computing and interaction design, as well as Art Nouveau using modern day technology.

Acknowledgements: The author wishes to thank the students of the Interactive Prototyping courses over the years, as well as the sponsors of the corresponding years: Iittala in 2009, Nokia Research in 2010, Microsoft Research Cambridge in 2011, and Bang & Olufsen in 2016.