The Design Process Continues

Attending Experiential Values up to Version 1.0

Rikard Lindell

Mälardalen University, Box 883, SE-721 23 Västerås Sweden
rikard.lindell@mdh.se

Abstract. How to attend experiential values of a design throughout the implementation is still an open issue. The interplay between experience design and software engineering is problematic because of the different epistemologies of design and engineering. Interaction design is a design practice, whereas software engineering describes itself as engineering and science. There is a long tradition in design of discussing materials and the craft of making artefacts. Thus, if we have a material, it is reasonable to say that we have a craft. If programming language code is a design material, then, making a finished artefact is the shaping of material. The development process can thus continue as a design process up to version 1.0. This paper presents a design case up to version 1.0 of a music creativity app, utilising design through programming. The app design validity was evaluated in a field study at an electronica music festival. Material consciousness of code, and an open-ended, and quality-driven design process allow attention to the experiential qualities of the design.

Keywords: Experience Design, Interaction Design, Music, Creativity, Craft, Artisanship, Material, Materiality.

1 Introduction

In the experience design field we devote ourselves to create methods, practices, and knowledge to design valuable, aesthetically pleasing, and usable digital artefacts. However when our designs become finished products, their experiential qualities often get lost in the process. Buxton [1] argues that version 1.0 is critical in the software life cycle for the user experience of a digital artefact. Thus, bridging the gap between designing digital artefacts and implementing them is important. Still, how to bridge this gap remains an open issue [2-4].

This paper delineates the design process from a research prototype up to version 1.0 of an iOS music creativity app. The case portrays the relationship between design decisions, code snippets, and the resulting appearance and behaviour.

2 Background

Boehm [4] shows that software engineering has started to acknowledge usability, and that requirements of interactive artefacts cannot be defined a priori. Stakeholders

K. Chorianopoulos et al. (Eds.): ICEC 2015, LNCS 9353, pp. 315–328, 2015.
DOI: 10.1007/978-3-319-24589-8_24

cannot articulate their needs to be transformed into a well-defined requirements specification. Nonetheless, models and methods in software engineering focus on solving problems and thus entail commitments to well-defined requirements [5, 6]. Engineers are trained to solve well-defined specific problems [7, 8]. Engineering focuses on convergent processes to determine one solution to one problem in a sequential refining order in an objective manner [8].

Schön [9] introduces the concept of *technical rationality* to offer an explanation of the engineer's epistemology. “Technical rationality depends on agreement about ends”. Schön discusses how faith in rational, scientific, and technological solutions became dominant. These approaches were successfully applied during World War II, where the solution to a problem was to supply more resources [9]. This epistemology is part of the historical heritage in software engineering, where the metaphor of engineering is used to describe programming. Bennington [10] introduces a “top-down” engineering development model for software in 1956. This top-down instrumental approach was named "The Waterfall Model" in the 70's [5]. The Waterfall Model is still important in the development of large projects [5]. Over the last decades so called agile techniques have developed to attend use cases and features, such as the Spiral Model, Rational Unified Process, Extreme Programming and Scrum [5]. Although Scrum is designed to handle chaos and change [11], it stipulates agreements to end in so called *sprints.* In each sprint a development team commits to implement as set of usable features. These must not change during the sprint that may last up to a month. Despite the emphasis on features, Lárusdóttir et al. [3] have shown that scrum teams often fail to attend user experience values of a design.

Schön points out that technical rationality cannot solve confused and conflicting situations: “When ends are fixed and clear, the decisions to act can present themselves as an instrumental problem. But when ends are confused and conflicting, there is as yet no problem to solve. A conflict of ends cannot be resolved by the use of techniques derived from applied research; it is rather through the non-technical process of framing the problematic situation that we may organise and clarify both the ends to be achieved and the possible means of achieving them” [9]. This quote suggests that problem-setting is crucial to understand a situation to design for. Framing the problem space of the context, and cut a search tree of plentiful design proposition to reach the right user experience design of a future artefact [7, 12]. Design is the exploratory use of malleable tangible materials and provides suggestions for possible future solutions [7, 13]. The design process is tightly connected to the material and the materiality of the design [14-16]. Information technology can be regarded as a material with no recognisable features [16, 17]. However Bertelsen et al. [18] introduced materiality as a concept describing, among other digital artefacts, electronic music artefacts developed with the MaxMSP programming environment. Thus, experience designers can learn from more traditional design disciplines to be attentive to the designs’ materiality [19]. Furthermore, a previous study has shown that the metaphor of material is applicable to program language code [15]. In this study the informants, users of programming languages, were concerned with the material’s internal malleability. In that the language can be processed and transformed according to desire and needs.

Fig. 1. The c3n play app version 1.0 running on an iPad 3 device. The photograph shows a zoomed in view of an arrangement of audio loops called a performance. Each performance consists of seven scenes indicated by the green circle segments around the performance. Each loop can be attached to any of the seven scenes. The current playing scene indicated by a fully saturated green colour.

There is a tradition of discussing material and craft and how they relate to each other [20], for instance, in the nineteenth century the discussion on handicraft and material in public education [21]. According to Adamson, "craft entail and encounter the properties of a specific material [22]." Artisans are, according to Sennett, conscious of the material and quality-driven, bordering onto the manic. They are busy perfecting their work with commitment to perform good artisanship for its own sake [23].

"Every good craftsman conducts a dialogue between hand and head. Every good craftsman conducts a dialogue between concrete practices and thinking; this dialogue evolves into sustaining habits, and these habits establish a rhythm between problem-solving and problem-finding. The relationship between hand and head appears in domains seemingly as different as bricklaying, cooking, designing a playground, or playing the cello..." [23]. The artisans are thus characterised by an ability to see and solve problems simultaneously in a dialogue between the hand and the mind. In this dialog talkbacks from the material tells the artisans what it wants to become [9]. Designers get talkbacks for their initial design ideas from design material such as sketches, storyboards, and mock-up prototypes. The goal of the design process is to frame, as much as possible, the problem for an engineering process to solve. In the ideal case, every problem is well defined and known. However, there are still design problems left unattended because the material of the design process is different from the material of the implementation process. Material consciousness of code and simultaneous problem-setting and problem-solving allow software artisans to be attentive to designs' experiential qualities in the making of the artefact [15].

3 Design Case

This design case presents the design process from a research prototype to version 1.0 of an iOS app for music creativity called *c3n play*. The development of the app was, eventually, conduced as a design project, carefully crafting the artefact in C programming language code. The design allows artists to play and collect loops, and create and edit performance arrangements. The content is presented on an infinitely large zoomable surface. The number of elements on the surface is however limited by the physical storage size of the device. The first version contains 285 loops. The user navigates with zoom and pan. Fig. 1 presents a photograph of an iPad running the app zoomed to a performance playing an arrangement of seven loops.

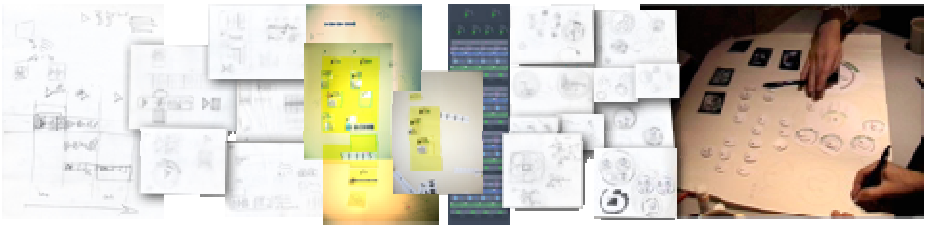

Fig. 2. Sketches and paper-prototypes exploring the user experience design for the research prototype.

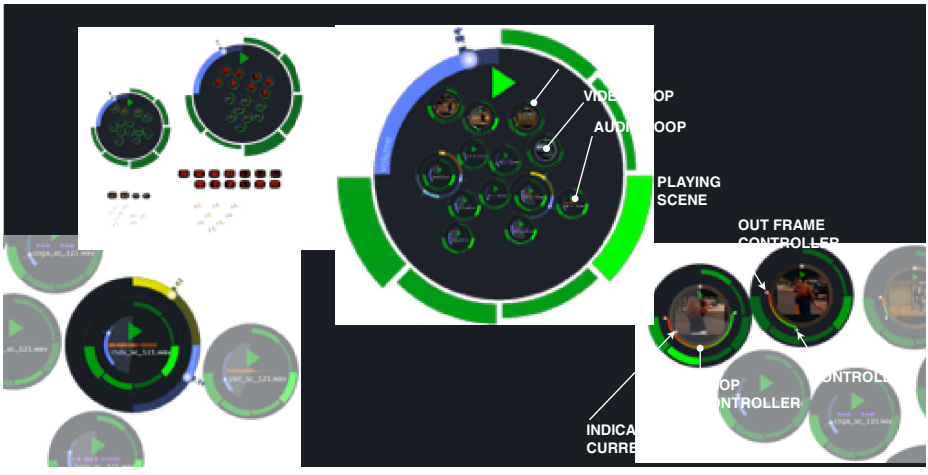

Fig. 3. This figure presents the research prototype's interface. The top left image shows an overview. The top middle image shows a performance containing audio and video loops. Each loop has scene tags connecting it to any of the six scenes. Bottom left image shows audio loops. The bottom right image shows video loops and their controls for selecting a sub-loop, beginning at IN FRAME and ending at OUT FRAME. The artist moves the sub-loop selection with the LOOP CONTROLLER to dynamically play different parts of the underlying video stream.

The starting point for the implementation of the interactive research prototype was the result of design process working with sketches, moodboards, paper prototypes

[24], see fig. 2. Fig. 3 shows the appearance of a few aspects of the research prototype's user interface. I did exploratory coding in the dynamic programming language for the design work. The coding was a conversation with the material; thus, many design problems emerged and were solved in this process. The goal was to make a sufficiently reliable artefact for field studies. Two music artists and a video artist evaluated the prototype in a multimedia performance at a festival playing electronic dance music. The collaborative design allowed the video artist to be more involved in the live performance. All artists reported this as a major advantage. Furthermore, they reported that the design in combination with touch screens gave the prototype the experiential quality of a music instrument. A video of the prototype field study can be found here: http://youtu.be/xslEtVnBnEo

4 The Making of Version 1.0

Encouraged by the results from the field study and with the advent of the iPad, I decided to make an app with this design. However, the design process had not answered all design questions. For instance, the text labels used in the prototype felt inconsistent with the design idiom. Instead of labels, we based the design on icon symbols. Inspiration for symbols came from Maya signs, electric symbols, and signs in astronomy, fig. 4 and 5. Eventually, crop circles showed to have interesting characteristics, fig. 6 (left). One can recognise a crop circle and distinguish them from each other. Fig. 6 (right) shows examples of symbols. The underlying design idea is that users will learn the meaning of the symbols from the feedback they get in interaction with the app, evoking hedonic attributes [24] of being in control.

Fig. 4. Three proposed designs for a symbolic language.

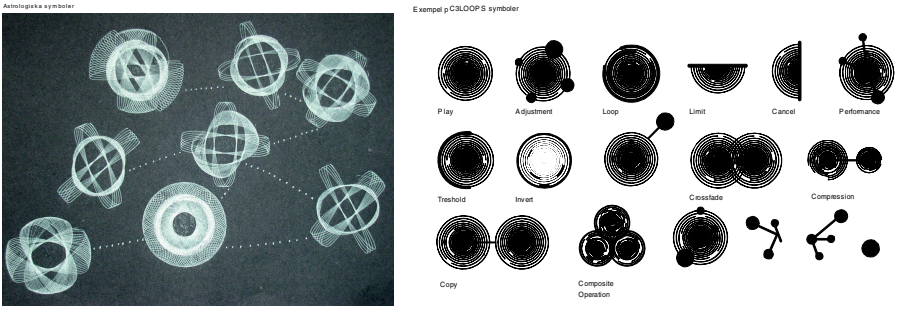

Fig. 5. Suggestion for a symbolic language based on symbols from astronomy.

Fig. 6. Crop circles as inspiration for a symbol language (left), Corn sign inspired symbols. Each symbol can be described from a string of data (right).

The normal procedure to include graphics in an iOS app is to put PNG image files in the app's resource bundle. We implemented the in an initial attempt to create a product. This project relied on hired software engineers, and we relied on the model view controller design pattern for iOS app development. The project failed for a couple of reasons. First, the carefully selected consultants focused more on state diagrams, sequence diagrams, and design patterns than on experiential qualities. Focus on technology turned out to be disastrous because of the remaining unanswered design questions. Second, the high level technologies in iOS prescribe a specific behaviour. For example, the implementation used CoreAnimation to display content. CoreAnimation is a technology designed to provide hardware-accelerated support for animated graphics and data visualisation. The documentation does not describe the CoreAnimation constraints. Thus, it is up to the developer to empirically evaluate its usefulness. We could maybe have avoided the failure with better management and better research of the technologies in iOS; thus, better engineering. However, design problems arise as talkback from the implementation. Then, focus needs to be on experiential qualities value, instead of on technical details. Finally, the project started over, this time with no outside programmers. Our focus was the experience design, and to implement a close coupling between storage, representation, presentation, and interaction with the content.

One of the first measures was to make a scene graph vector rendering system as replacement for CoreAnimation to present content on a zoomable surface. In this environment bitmap images did not feel consistent with the appearance of the design and the zoomable interface. It felt natural to rely on the scene graph code for the symbols too. The features of the symbols in figure 7 were analysed and transformed into data descriptions for vector graphics. The symbols consist of arcs, circles, and lines from the origin. Each feature has four attributes, shell (1, 2, 3, or 4), starting angle, length (in angle, or shells), and origin offset.

A software engineering approach would have been to define a file format or use an established format, for instance SVG (scalable vector graphics) or JSON (JavaScript Object Notation). The later is a data descriptive attribute–value pairs format expressed in human-readable text that can be parsed by the JavaScript interpreter in a web browser. The Lua scripting language, used for the research prototype, also has data

descriptive attribute–value pairs features. However, we carved the app in C, which is tedious and cumbersome, nonetheless we transferred the acquired skills and practice of Lua scripting to C programming. Instead of engineering a so-called *content pipeline* relying on file formats and shared libraries, the features of the symbols were described directly in C. Thus, using C as a data descriptive language.

```
void C3Symbol_createSymbols(){
   const unsigned short performanceComponents[][5] = {
   // type                 shell    angle    length offset
      {kC3SymbolShapeArc,  4,       0,       360,     0},
      {kC3SymbolShapeArc,  3,       2,       41,      0},
      {kC3SymbolShapeArc,  3,       47,      41,      0},
      {kC3SymbolShapeArc,  3,       137,     41,      0},
      {kC3SymbolShapeArc,  3,       182,     41,      0},
      {kC3SymbolShapeArc,  3,       227,     41,      0},
      {kC3SymbolShapeArc,  3,       272,     41,      0},
      {kC3SymbolShapeArc,  3,       317,     41,      0},

      {kC3SymbolShapeNil, 0, 0, 0, 0} // terminator
   };
   const unsigned short addMediaToSceneComponents[][5] =
{
   // type                 shell    angle    length offset
      {kC3SymbolShapeArc,  4,       0,       360,     0},
      {kC3SymbolShapeArc,  3,       2,       41,      0},

      {kC3SymbolShapeNil, 0, 0, 0, 0} // terminator
   };

   ...
C3ZNodeRef symbol;
   symbol = C3Symbol_createSymbol(performanceComponents);
   _symbols[kC3SymbolPerformace] = symbol;

   symbol =
C3Symbol_createSymbol(addMediaToSceneComponents);
   _symbols[kC3SymbolAddMediaToScene] = symbol;

   ...
}
```

The dataset for each symbol is parsed by the function C3Symbol_createSymbol, and the resulting scene graph node is kept in a static array. This approach has a couple of advantages: the code is cleaner because there are no dependencies to file formats, and shared libraries, and the launch time of the app is shorter because the app binary

contains the compiled data. The top right of fig. 9 displays the resulting symbols for *performance* and *addMediaToScene.*

Fig. 7. In zoomable interfaces, position and scale may be arbitrary. To compensate for this, we sought a design for sliders with fixed interaction position regardless of the slider's value. Vintage telephone dials in combination with lids from McDonald's coffee cups provided inspiration for the slider design.

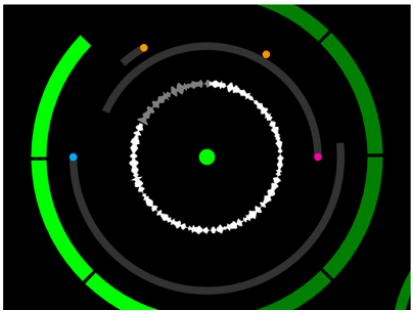

Fig. 8. A loop in a performance. The seven green tags indicate which scenes the loop is attached to. The artist tap the scene tags to attached or detach the loop to the corresponding scene. The green dot adds or removes the loop to or from the current playing scene. The blue, orange, and pink dots are slider heads at a fixed. The length of the slider's tail conveys its value.

Another example of an unanswered design problem from the research prototype was the design of sliders. In fig. 3 the sliders were arcs with various positions for the head. In a zoomable interface position and scale varies. Thus, we sought for a design with fixed positions of slider heads to make the design more consistent. In fig. 7 old dials and the layout of a plastic coffee cup lid provided inspiration for the design of sliders portrayed in fig. 8.

Fig. 9 shows an overview of c3n play. We used a spiral design language for the organisation of content. The prototype organised content in a grid, this was, however, inconsistent with the design idiom. Sub-spirals organise collections of loops in a spiral. The spiral expands with new performances and new content. The symbol for creating a new performance from the current playing loops or the symbol for appending these loops to the current playing scene creates a flow between navigating, creating arrangements and performing.

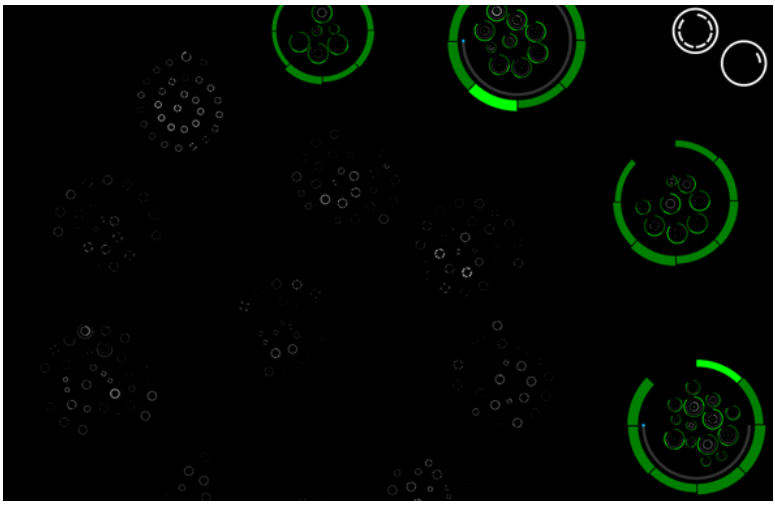

Fig. 9. An overview of the c3n play app version 1.0. A spiral of audio loops collected in sub-spirals is shown in the middle. Arrangements of loops collected in of performances are shown in the continuation of the spiral. Bright green indicates a playing scene of a performance. A tap in the left top right symbol collects the playing loops and creates a new performance. A tap on the right top right symbol adds the current playing loops to the last playing scene.

In the development of version 1.0 we used the Kanban development model. It has attracted attention by providing freedom for adaptation [24]. "Kanban leaves almost everything open. The only constraints are Visualize Your Workflow and Limit Your WIP [work in progress]. Just inches from Do Whatever, but still surprisingly powerful." [24]. The goal of a work in progress can change during the process; thus, it allows open-endedness and simultaneous problem-setting and problem-solving. The model allows the concrete material from the design process – mood boards, sketches, storyboards, videomatics etc – to be used to describe functionality. Hence, Kanban is a radically different approach than the earlier development models and allows an artisan approach in direct engagement with the features of the materials [21].

This is a link to a tutorial video for the app: http://youtu.be/gOdJwlvMOFA

5 Evaluating the Validity of the Design

We carried out the design process with professional music and video artists to perform music and video live on stage. Evaluations of prototypes have shown that the design is useful for this purpose and context. We wanted to make a design that was challenging at the first glance, but that attracts and evoke interaction, and that shows its semantics and functionality through interaction without instructions and descriptions. Our aim was to create conditions for a sense of skill among users, creating enjoyment of the design [25]. However, a new electronic musical instrument must have a low learning threshold while providing room for virtuosity [26]. The design needed to be balanced between these qualities.

Krippendorf [12] discusses the validity of a design, specifically relevant here is pragmatic validity and experimental validity. “Pragmatic validity. If the stakeholders of a design are committed to support, promote, realise, or use it, this surely is a kind of evidence that no one could ignore. Pragmatic validity lies in the hands of its stakeholders, as it should. In some sense, pragmatic validity is the ideal of a self-evident proposal that requires no further explanations.” In this quote Krippendorf [27] means that users’ statements about a design in various forums, reviews in online stores, and the count of followers or *likes* in social media constitute evidence to the validity of the design. In comparison with the research prototype, c3n play is a crippled app with only a few of the features necessary for professional performance use. Yet, the version 1.0 conveys the design idiom of the user experience from the research prototype. Here follows some positive user reviews of the app:

“Awsome I found it very Easy and amusing. It doesn’t lag and you have a large play area, I made good beat in 6 min Max Its Worth it! Try out its an awesome app!”

“Newthinking!! Great idea and interesting interface, easy to get a groove going but still takes a while to grasp.”

“great interface ,simple yet genius layout”

“Greate stuff I have tried dozens of music programs but no one is better than this app. Easy and cool to use!”

“Awesome app! I love how easy it is to play around with different sounds and compositions. The UI is interesting and solves a complex problem in a fun way!”

In an update of the iOS operating system, we missed a bug that resulted in a lower rating. “Nice app (3 of 5) I really like the app, but its not working well at the moment.pls fix ”

The reviews indicated the pragmatic validity of the design. However, the 19 reviews in AppStore are a fraction of all the 4,070 downloads. The reviews are encouraging, but they do not provide a rich picture of the design's validity.

In the following quote Krippendorf [28] delineates experimental validity: “Experiments with prototypes … [a]nswers to such questions as to whether people can recognise the prototype for what it affords, how many disruptions they experience while interfacing with it, the characters traits they would attribute to it, and which user identities the are believed to support define statistical distributions. Experimental validity … allowing subjects to interact with a prototype, which can yield unexpected evidence in support of or against the claims advanced by its designer.”

A rigorous design process still needs to be evaluated to show what meaning users attribute to the design. Evaluations show if the design fulfils the intended experience qualities.

To receive a complementary view of the design's validity, we conducted an evaluation at the Volt Festival 2014, an annual electronica music festival. We set up two iPads tablets connected to a mixer. Each tablet also had a pair of headphones. Festival visitors were invited to play. There was always music in the speakers, but if users felt unsafe, they could listen to the music they played only in their headphones. A video camera by each tablet collected data. The evening generated in total four hours of video data for 38 users. I analysed the material on users’ engagement and learning. How long users played, if they made positive comments, or made suggestions for

improvements to the design indicated engagement. If users seem to understand what they do, and if they asked for help, shows learning.

The average time was over eight minutes and median time was five minutes. Seven users played for 18 minutes or longer. Three of these users learned the design without introduction and one of them played entertainingly (my subjective assessment). Four of the seven needed an introduction to the design. One of these suggested improvements of the design (a button that stops all playing loops). Another user spontaneously comments: "its quite fun", "its a good one", "ah, its really cool." Two drunken users played in the late night for almost half an hour, afterwards one of them said: "this was the evening's best dance floor." This comment is somewhat narcissistic in the sense that you played better than all the artists at the festival; however, the comment indicates engagement.

Two users played for 12 or 13 minutes. These two men were intoxicated. They did not ask for help. They seemed not to learn the design.

One user played for ten minutes, she received an introduction to the system and then explored the design on her own. After six minutes, she began to dance to the music she created, an indication of engagement.

The remaining 28 users played in seven minutes or less. Ten of them had no introduction to the app, the two of these tried to use the app for a minute and did not learn how the design works. There were three of the 18 who receive an introduction to the design that did not learn the design either.

A total of seven users (including the two intoxicated) did not learn how the design works. They navigated primarily through panning, and played loops, the rest of the design's functionality remained hidden for these users.

5.1 Discussion of the Results

The evaluation was not a controlled experiment, but rather a field study without a prior stipulated hypothesis. The basis for the analysis of the data was the design's intended experiential values and to find indications if these were valid or not. Thus, one cannot say that the evaluation experiment was quantitative despite counting the number of minutes played, and how the subjects asked for help or were offered help. Since I counted and measured, I cannot say that the evaluation was qualitatively. However, the interpretation of the data was qualitative.

The app is designed for a niche. For that reason, I have prioritized interpretations from data of the users who played longer than 18 minutes. The time they devoted to play the app implies experimental design validity. However, four of the seven users needed or received an introduction to the design. This indicates that the design should be more accessible. Without a tutorial that describes the design, the first step seamed to big. It was also clear that it was difficult for users to understand the sliders. The vertical gesture manipulation that allows multiple sliders to be altered simultaneously, did not map to the sliders circular presentation.

The design of the sliders needs to be modified so that there is an affordance for vertical movement. We will also search for a design that makes it easier to get started with the app, without breaking its aesthetics, a design that leads users into the design

so that they gradually learn how it works. Pohlmeyer [25] shows that a challenging design leads to satisfaction that lasts longer. The learning approach to the design corresponds to excitement in Csíkszentmihályis [29] flow model, excitement of being able to cope with something above your ability.

6 Discussion

The design case in this paper shows that the design process does not stop when the implementation start. Design through programming and treating programming language code as a design material for digital artefacts helped the making of a version 1.0 that entailed the intended experiential values. Dourish and Mazmanian suggest that there is a materiality of digital representations, and digital technologies need to be studied on their own materiality and on their particular forms of practice [30]. Previous results have shown that program language code can be considered a material [15]. Thus, masons chopping letters from stone with a chisel and a hammer can be metaphor for the description of symbols in C code. For a more sophisticated approach, a deeper analysis of the symbols could provide several glyphs that combined from a string of characters present the symbols. Yet, the stone mason's craft affected the appearance of the Roman alphabet characters. Similarly, the descriptive C code affected the look of the symbols; compare fig. 6 (right) and 8. Using C is sound for responsive, highly interactive, and performance demanding digital artefacts, especially on computationally weak devices. However, designers have increasingly developed the capacity of programming themselves, instead of relying on software engineers, through more pliable tools and dynamic languages such as ActionScript, JavaScript, and Processing.

Lárusdóttir et al. have showed that the agile software engineering method scum fails to attend experiential qualities in development projects [3]. Buxton criticised the engineering approach in his open letter on Engineering and Design [7]. The engineering community itself struggle with the issue [5]. In the design case above a year of work was lost because of engineering. Instead of engineering, as suggested in the design case, the gap between interactive prototype and version 1.0 can be bridged through a quality-driven and open-ended artisan approach characterised by material consciousness of code and careful attention to the experiential qualities of the design.

References

1. Buxton, B.: Sketching User Experiences - getting the design right and the right design. Morgan Kaufmann (2007)
2. Lai-Chong Law, E., Abrahão, S.: Interplay between User Experience (UX) evaluation and system development. International Journal of Human-Computer Studies 72(6), 523–525 (2014)
3. Lárusdóttir, M.K., Cajander, Å., Gulliksen, J.: The Big Picture of UX is Missing in Scrum Projects. In: Proceedings of the 2nd International Workshop on the Interplay between User

Experience Evaluation and Software Development. In Conjunction with the 7th Nordic Conference on Human-Computer Interaction (2012)
4. Memmel, T., Gundelsweiler, F., Reiterer, H.: Agile human-centered software engineering. In: Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI...but not as we know it, (BCS-HCI 2007), vol. 1, pp. 167–175. British Computer Society, Swinton (2007)
5. Boehm, B.: A view of 20th and 21st century software engineering. In: Proceedings of the 28th International Conference on Software Engineering (ICSE 2006), pp. 12–29. ACM, New York (2006)
6. Kroll, P., von Krüchten, P. The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP. Addison-Wesley Professiona (2003)
7. Buxton, B.: On Engineering and Design: An Open Letter. Businessweek, April 29 (2009), http://www.businessweek.com/innovate/content/apr2009/id20090429_083139.htm (accessed April 27, 2015)
8. Löwgren, J.: Applying design methodology to software development. In: Proceedings of Designing Interactive Systems, pp. 87–95 (1995)
9. Schön, D.A.: The Reflective Practitioner - how professionals think in action. Basic Books (1983). ISBN: 0-465-06878-2
10. Bennington, H.D.: Production of Large Computer Programs. Annals of the History of Computing, vol. 5(4) (October 1983)
11. Schwaber, K.: SCRUM Development Process. In: Workshop Report: Sutherland, Jeff. Business Object Design and Implementation of 10th Annual Conference on Object-Oriented Programming Systems, Languages, and Applications Addendum to the Proceedings, vol. 6(4), pp. 170–175 (1995)
12. Krippendorf, K.: The Semantic Turn. CRC Press, Taylor & Francis Group (2006)
13. Wiberg, M.: Methodology for materiality: interaction design research through a material lens. Personal and Ubiquitous Computing 18(3), 625–663 (2014)
14. Vallgårda, A.: Giving form to computational things: developing a practice of interaction design. Personal and Ubiquitous Computing 18(3), 577–592 (2014)
15. Lindell, R.: Crafting interaction: The epistemology of modern programming. Personal and Ubiquitous Computing 18(3), 613–624 (2014)
16. Löwgren, J., Stolterman, E.: Design av informationsteknik - materialet utan egenskaper. (Design of Information Technology - the material without properties) Studentlitteratur (2004)
17. Robles, E., Wiberg, M.: From materials to materiality: thinking of computation from within an Icehotel. Interactions 18, 32–37 (2011)
18. Bertelsen, O.W., Breinbjerg, M., Pold, S.: Instrumentness for creativity mediation, materiality & metonymy. In: The Proceedings of the 6th Conference on Creativity & Cognition, pp. 233–242 (2007)
19. Redström, J.: On Technology as Material in Design. Design Philosophy Papers: Collection Two: 31-42. Team D/E/S Publications (2005)
20. Salomon, O.: Introductory Remarks, from The Teachers' Handbook of Slöjd. In: Adamson, G. (ed.) The Craft Reader, Berg, Oxford. Silver, Burett & Co excerpted, Boston (1891) (2010)
21. Adamson, G.: Thinking Through Craft, Chapter 2 Material, Berg (2007)
22. Sennett, R.: The Crafsman. Penguin Books (2008)
23. Wallace, J., Press, M.: All This Useless Beauty: The Case for Craft Practice in Design For a Digital Age. The Design Journal 7(2), 42–53 (2004)

24. Kniberg, H., Skarin, H.: Kanban and Scrum - making the most of both. C4Media Inc. (2010)
25. Pohlmeyer, A.E.: Enjoying joy: a process-based approach to design for prolonged pleasure. In: Proceedings of the 8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational (NordiCHI 2014), pp. 871–876. ACM, New York (2014)
26. Wessel, D., Wright, M.: Problems and Prospects for Intimate Musical Control of Computers. In: Proceedings of New Interfaces for Musical Expression (NIME 2004), Shizuoka University of Art and Culture, Hamamatsu, Japan, 3-5 June (2004)
27. Krippendorf, K.: The Semantic Turn, p. 267. CRC Press, Taylor & Francis Group (2006)
28. Krippendorf, K.: The Semantic Turn, p. 264. CRC Press, Taylor & Francis Group (2006)
29. Csíkszentmihályi, M.: Flow: The psychology of optimal experience. Harper & Row, New York (1990)
30. Dourish, P., Mazmanian, M.: Media as Material: Information Representations as Material Foundations for Organizational Practice. In: Proceedings of the Third International Symposium on Process Organization Studies (2011)