ABSTRACT
In online communities makers share and give feedback on DIY projects. Such feedback could also help novices who get stuck in their projects. However, documenting work in progress is little considered in current tools. We therefore developed a How-To related web platform for documenting work in progress and studied how children (aged 13-18) used it to document their physical computing projects during workshops. The evaluation outcome questions the appropriateness of our web platform and reveals the benefits of visual programming environments for documenting physical computing artifacts. Suggestions are given how to extend visual programming environments into minimalistic documentation tools that provide ways for children to successfully share their work in progress with other makers.

Categories and Subject Descriptors
K.3.2 [Computers and Education]: Computer and Information Science Education

Keywords
Documentation; Visual Programming; Physical Computing; DIY; Children

1. INTRODUCTION
The maker movement brings about easily accessible resources to a wide audience, comprising hardware kits as well as informal learning material such as How-To tutorials and videos created and shared by amateurs. They offer informal learning approaches to sometimes complex issues such as physical computing. However, most of the work presented on maker platforms (e.g. [7, 5]) shows successfully finished projects. We presume, that learning from each other can benefit from sharing unfinished projects, including projects that got stuck and failures. This seems especially relevant when young novices want to start making and learn something new, but face a lot of obstacles.

To benefit from other makers' help and to contribute to this maker movement, the state of a project needs to be communicated, including visions and/or current problems. To be able to communicate and share a work in progress in an online maker community, or to resume own projects later, requires having documentation at hand. It needs to be comprehensible and give a comprehensive picture, including the artifacts appearance, functionality, its authors' motivation, problems, attempts undertaken etc. But documenting work you are engaged in is laborious and not much fun because it interrupts hands-on construction activities. We are interested in how to support children (mostly teenagers), who are beginners in physical computing, to document their work in progress. Initially, we wanted to find out how a documentation tool could be designed and used for this purpose. To investigate this we developed a web platform that follows the common "How-to" format and evaluated how children used it to document their physical computing projects.

After giving some background on documentation tools and physical computing in maker and IDC contexts, we introduce the web platform, its functionality and design. The evaluation setting and methods are described. The outcome of the evaluation is presented, shifting the focus of this paper towards another kind of tool—visual programming environments—that seem more suitable for children's documentation of physical computing constructions. This is followed by a further look into work related to the outcomes, before we discuss the results and conclude the paper.

2. BACKGROUND
2.1 Documenting and Sharing
The maker movement creates a vast amount of informal learning resources. Many come from DIYers who document and share their projects online to learn from each other. They usually create this documentation in a very concrete and informal way, e.g. by means of How-To instructions which consist of pictures with text (similar to storyboarding) and demonstrate hands-on "how to" construct an artifact [19]. The How-To format is very popular, and is used on platforms like Instructables [7]. Variations of this picture-text format can be found. Diy.org [5] wants to encourage children to make and share DIY projects. Projects can be documented in a gallery style with pictures or links to videos and a description. The platform is targeted at presenting finished projects. The mobile app How.do [6] transfers the
How-To principle to mobile devices. Users can upload photos or videos of the project steps. Instead of typing text, descriptions can be audio recorded. Using a handheld device for documenting projects seems well integrated into the making process.

These very general formats are suitable for documenting diverse DIY projects and are not explicitly designed for documenting physical computing projects. They focus on documenting outcomes, not work in progress.

2.2 Constructing Physical Computing Artifacts

Technology development has not only contributed to the maker culture in the form of Web 2.0 communities: physical computing technologies such as Arduino [2] have become accessible to people not trained as engineers or programmers in recent years, and many educational Construction Kits [10] for children are based on Arduino components. Such kits are used in various educational settings with children who engage in hands-on experiences, inventing personally meaningful computing artifacts (e.g.,[11, 4, 3]).

2.2.1 Visual Programming Languages

Physical computing construction kits require programming. Most tools that target children and novices use visual programming languages (VPLs). The basic elements of a VPL are graphic symbols which represent programming commands. The symbols can be composed into an executable program. The graphical symbols simplify the syntax [12]. Only certain combinations of symbols are allowed by the programming software, so that syntactic rules are inherent. A prominent example of a visual programming environment (VPE) with a VPL for kids is Scratch [16], including releases for programming Arduino [9].

3. THE WEB PLATFORM

In order to provide a flexible tool that facilitates documenting work in progress as well as project processes and not just project outcomes, we developed a web platform (referred to as WP hereafter). It was designed as a web community (accessible only by workshop participants) and implemented some principles derived from the How-To format. E.g., each WP item (similar to a step in a How-To) can contain one or more pictures, descriptions and file (e.g., program code) or video attachments. Pictures can be uploaded or directly captured from a webcam and annotated. Items can be commented by other users. How many items one creates to document the current state of a project is optional. On the project overview page, thumbnails of single steps can be aligned or grouped arbitrarily (fig. 1). Instead of displaying single steps linearly like common How-Tos do, individual ways (with dead end branches) of a project can be visualized. Therefore, non-linear development processes can also be documented, different solutions can be shown as different branches, and several items can be grouped as a project step.

The tool offers specific features for physical computing projects. When creating an item, standard pictures of hardware components (e.g., LEDs, sensors) can be selected from an image gallery, and file attachments allow for uploading program code. Items can be assigned to categories according to their focus (hardware development, programming, project idea, etc.). The assigned category determines the frame color of the step thumbnail shown on the project overview page. The design of the tool was influenced by insights gained through running physical computing workshops for children for more than six years [18] following Constructionist [15] principles. Children did not actively participate in the design phase, but the workshops described below were meant to generate feedback for a redesign of the WP.

4. EVALUATION

The evaluation aimed to find out how children make use of the tool, whether the chosen format and features were suitable and perceived as useful, and how the artifacts were represented. The latter issue was examined in a content analysis of project documentations created by kids, while the previous issues were investigated in-depth in a case study.

4.1 Context and Setting

The tool was used and evaluated in the context of three TechKreativ Arduino workshops [18, 4]: a weekly 2-hour workshop during five weeks run as an after-school activity with 7 tenth-graders (3f/4m) and a tutor, and two four-day-long full-day workshops at a summer camp with 15 children (aged 13-18) with two tutors each. After an initial brainstorming about smart objects and a brief introduction to the technology and programming, groups of two to four participants set their project goals and worked on them for the rest of the workshop (project examples in fig. 3). The tutors tried to intervene as little as possible, and assisted the participants only when needed. The material included Arduino and Arduino LilyPad boards, switches, sensors and actuators, wires, resistors, crafting material and computers with a VPE with access to the WP. The children were encouraged to use the WP for their own needs, e.g., to create a documentation for others and for themselves for resuming projects later, or to look up solutions from others’ projects.

The children used the visual programming environment (VPE) Amici [1] to program their artifacts. Amici offers a graphical interface to program Arduino boards (fig. 2). It is based on a block metaphor where graphical blocks represent programming commands. The block syntax is translated into textual Arduino programming code automatically. When choosing blocks to program sensors and actuators, the user is asked to select graphical icons representing the hardware sensors or actuators and pins, which are then visible...
4.2 Case Study

At the weekly after-school workshop an in-depth case study was conducted with a project group of four pupils (2m/2f). They were inventing, constructing and programming a "smart cup" that showed the right drinking temperature (fig. 3 left). This project group was selected for the case study because this workshop was run on a weekly basis, making documentation appropriate for remembering the current state. When compared to other kids, the members of this group attended all sessions, were equally mixed in terms of gender and at an average age.

During and after all sessions brief field notes were taken by a tutor. One week after the last session, a semi-structured group interview was conducted by the tutor with the case study project members in an informal gathering. Its purpose was to find out how the participants perceived the tool and its features in general, why they chose to document certain artifact characteristics and use certain media types, and how they would redesign it. Screenshots of the WP features and the interviewees’ project documentation were provided to facilitate the discussion and to scribble suggestions for redesign.

4.3 Methods for Content Analysis

The interview and field notes were analyzed with a content-structured qualitative content analysis [13]. The interview of the case study was transcribed and coded together with the field notes by one coder in two rounds. The findings were summarized according to coding categories.

Additionally, all eight project documentations created with the How-To were analyzed with scaling-structuring content analysis [13]. The purpose of this analysis was to gain insights about which specific media types were used to document different characteristics of the artifacts. The coding was done as described above. The final categories represented different characteristics (in/formal, static, dynamic) of the artifact, pragmatic aspects (such as context, authors, intention), and media types. E.g., a photograph of the hardware setup was coded as "hardware" and "iconic graphics". Results were summarized quantitatively on a percentage basis by documentation and item type. They, together with the field notes, served as contextual material for the interview evaluation.

5. OUTCOME

Unfortunately, use of the WP to document project states was very limited (5 project documentations with 8 items in total). E.g., at the summer camp only two documentations were created. Using the tool imposed additional effort, as it was not naturally integrated in the construction process.

The document analysis also showed that the image upload and webcam function of the WP was the documentation feature most used (5 items) besides program code upload (all items). Very few descriptive texts were added (2 items). Formulating descriptions seemed to be challenging especially for the younger children (according to the field notes). Features like image annotation and categorization of steps were not used. Also the project overview page was hardly considered. Some just updated the latest WP item to the current project state instead of creating a new item.

The kids of the after-school case study uploaded visual program code after each session and sometimes webcam captures of the built-up hardware configuration, a sketch, or of themselves. In the interview they told that they had taken pictures of the wired hardware components (like in fig. 3 left) to be able to build it up again, but that they had not used nor needed it at the next workshop session. Instead, they realized that they had found all the necessary information in the visual program code immediately: "Actually [...] the program was sufficient where you could see what was connected where" one boy concluded. Asked what other information they considered relevant for external makers to reproduce the project, they argued that, besides a detailed picture, they would say what the artifact was supposed to be and describe incidents that occurred.

Due to these rather unexpected findings, we analyzed 16 Amici programs (documented in the WP plus others created by kids in similar workshops, see also fig. 2). The same analyzing procedure and categories used for the WP document analysis were applied. It was found that about one third of the analyzed programs did not only contain characteristics of the program and hardware setup. By means of method declarations and program code file names, information about the appearance, behavior or purpose and context of the artifact (e.g., "future", "blink", colors) were mentioned. The
outcomes also confirmed us about design decisions made for Amici previously [18].

6. RELATED WORK ON VPEs

Attempts to implement additional value into VPEs for children and novices have been made by others. Some VPEs for Arduino have features which are suitable for documenting hardware setups, similar to the ones of the Amici VPE. Examples are ModKit [14], Minibloq [8] and S4A [9]. S4A builds on Scratch, which is connected to a community website. Programs can be uploaded directly from the IDE [16]. Extensions have been suggested for encouraging Scratch programmers to reflect on their intentions [17]. However, these tools are not comprehensive in the sense that they allow users to document all facets of their artifacts, including “soft” characteristics such as appearance, purpose or functionality.

7. DISCUSSION

The WP proved to be less suitable for documenting the artifacts than expected. This was probably due to the fact that they required extra effort and because using it interrupts the construction process. Surprisingly, the evaluation pointed to VPEs potential not only for programming, but also for documenting. The children found the program files sufficient to rebuild their project configuration, especially because they could read the hardware configuration at a glance from the VPE program. Direct access to a webcam proved to be very suitable to facilitate the integration of pragmatic aspects (e.g., context and authors) and informal characteristics like the artifact’s appearance or sketches.

A VPE is well integrated into the construction process, since its usage is essential for activating a programmable artifact. With a VPE like Amici, mostly formal characteristics of the artifact are represented, although some hints about its appearance or the authors intentions were indicated by method and file names. We suggest that documentation tools for physical computing targeting children and novices start from VPEs. Short text fields and image fields with webcam access can enhance a VPE to facilitate the documentation of artifact attributes such as appearance or the authors’ intentions. Also a version control panel could be useful to go back to earlier programs and track changes, but still focus on the current state.

The tool was evaluated at tutored technology workshops. Being a case study, only one project group was interviewed and looked at in-depth. We did not verify whether the documentations are comprehensible for young makers who have not participated in such a workshop. More research is needed to verify the results with such makers, including their motivation to document projects.

8. CONCLUSION

This paper has presented a study of how children use a How-To related web platform to document their physical computing projects in the context of three workshops. Surprisingly, the evaluation shifted the focus from the initial WP to the VPE the children used to program their artifacts. A VPE has inherent potential for project documentation since its programs were perceived as sufficient and practicable by the kids. The visual programs created with such a VPE contain adequate information to rebuild the current state of a project. Creating programs is an essential part of the construction process and therefore such a tool is already part of the artifact construction.

We argue for VPEs as basic documentation tools for children constructing physical computing artifacts. To enable a comprehensive picture of the artifact covering its informal and pragmatic characteristics, we suggest integrating image fields with webcam access and small text fields.

This work paves the way for tools that allow them to more conveniently document on-going physical computing projects and eventually share them in maker communities. We have started implementing the recommended features in Amici. For future work, it needs be evaluated. Further, export and upload features (e.g., export into common web formats) are required to make the VPE a useful tool that helps children to share projects in a maker culture.

9. REFERENCES