ABSTRACT
Developing and maintaining HCI toolkits is a challenging task. In this position paper, we present three types of challenges that potentially turn toolkits into abandonware: organizational, institutional, and technological challenges. We derive the challenges from our own experiences in developing HCI toolkits and they have been consolidated based on the three sample toolkits Squidy, HuddleLamp, and Webstrates. We describe the overall goals of the toolkits and their application areas, report on their uses and the current state of development, and link them to the challenges. We conclude with questions for the HCITools workshop.

Author Keywords
Abandonware; toolkits; frameworks; libraries; enabling technology.

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

INTRODUCTION
Third-party software like libraries, frameworks, and toolkits often make the foundation of research prototypes used in HCI research. They enable researchers to explore new ways and ideas of computing systems to improve interaction between humans and computers. More general, in the fast-advancing field of computer science it is nearly impossible nowadays to have a complete understanding of inner-workings of computing hardware or software algorithms. A third-party software thus can provide a level of abstraction that allows researchers to build prototypes with complex computations – like 3d transformation, server/client or peer-to-peer communication, image- and video-processing, artificial intelligence, or machine learning – even with only basic programming skills.

For example, OpenCV is a framework that implements hundreds of computer vision algorithms in C/C++. It provides them to the community in a comparatively low-effort application programming interface (API) and for real-time computer vision processing. In the past, the OpenCV library was mainly for vision-based processing, hence the name open source computer vision. Nowadays, the library also implements neural networks for machine learning. Programmers can use the library without prior or extensive knowledge in computer vision and i.e. apply image processing algorithms to an input image such as canny edges. Beyond the C/C++ API, various open source libraries bind to OpenCV and make it accessible in other programming languages like Java or C# (e.g., Emgu CV).

OpenCV is a great example of a successful project that went from research to industry, and myriads of commercial, open source or research projects use it. A large community maintains it, continuously develops new algorithms, and publishes stable releases on a public website.

Large communities maintaining software, however, is very unusual for projects originating from research. This entails a risk, especially when toolkits are part of a research paper contribution. From our experience, it happens very often that building a project from source code or running it fails due to “out-of-date” third-party libraries, libraries that are not available for download anymore, or projects do not support the latest version of a runtime environment like the Common Language Runtime (CLR for .NET), Java Runtime Environment (JRE), or use deprecated Web standards. Such toolkits developed in research projects are deemed to end as abandonware where the authors stop working on it and reported bugs will not get fixed after the associated paper was accepted.

Despite this risk, HCI toolkits have many potentials and play an essential role in advancing HCI. They are often a great source of inspiration and help researchers to think outside the box. Duval argues that “[...] innovation in general – is that [...] sometimes just showing somebody a concept is all that you have to do to start an evolutionary path. And once people get the idea of 'hey we can do that', then somebody does something, somebody does something better, that just keeps developing.” (Duval 2011 as cited in [1])

In retrospect, much of our research has been inspired by HCI toolkits. The Proximity Toolkit by Marquardt et al. [7] is an excellent example (the accompanying paper has around 70 citations at ACM DL). We have used it for a study to investigate the effect of body movements on users’ spatial memory while navigating large virtual information spaces [9]. It al-
allowed as to rapidly test various interaction techniques during a focus group and leverage the group to discuss pros and cons of each interaction. The toolkit helped to choose an interaction technique appropriate to answer the research question.

In the remainder of this position paper, we will introduce three challenges that are potential causes for abandonware, report our experience in creating software toolkits for HCI, and conclude with questions that we would like to discuss with participants during the workshop.

**CHALLENGES AND OPPORTUNITIES**

Resonating with Duval’s statement above, we believe that HCI toolkits are catalysts for innovation and spark new ideas. However, in this position paper, we also want to emphasize on three challenges that HCI toolkits face before they eventually become abandonware.

**Organizational**

Open source projects need a large community to maintain code, and ideally more than one responsible person (maintainers) who take the lead on the project and divide the workload equally among each other. They are responsible for accepting pull requests, continuously check the quality of the source code, make sure the project builds correctly, put new releases online, and define goals for future developments. An important and often criticized aspect of open source software is the lack of proper documentation or working examples. In the case of poor or no documentation, programmers who use the toolkit have to dig through the source code manually to decode and understand how it should be used.

**Institutional**

The reusability of research toolkits is often limited for scenarios presented in their accompanying research papers, which impedes the possibility of using them in new and meaningful ways. We believe there are a number of institutional factors that impede the continued development of toolkits. Firstly, time is a precious resource in academia and scientists who pursue an academic career are often confronted with the balancing act between community service by offering toolkits to be used freely and advancing the career by increasing publication count and boosting the h-index. Secondly, paying developers to maintain a toolkit or renting proper build infrastructure (e.g., continuous integration server) is costly and may be difficult to budget on research grants. Thirdly, continued work on an already published system or toolkit may be discouraged as it is considered incremental rather than novel work.

**Technological**

The technological challenges also often impede with the reuse of toolkits. For example, runtime environments depreciate or special hardware is required to run it but is not available for purchase anymore. Or infrastructure like code repositories disappear. Changes in the technological landscape can also impact the reuse of toolkits when technology emerges, and suddenly previously popular technology fades away. A few years ago TCL/TK was fashionable, but nowadays it is rarely used. This can likewise happen to current mainstream technology (e.g., HTML5 or Unity3d).

**EXPERIENCE WITH HCI TOOLKIT DEVELOPMENT**

We have worked on several research toolkits that reached various levels of maturity, which also led to a consolidation of challenges as mentioned earlier. We present three of our toolkits by describing their overall goal, report on their uses and the current state of development, and link them back to the challenges.

**Squidy – Open Source but Concluded**

Squidy [6] is an interaction library which eases the design of post-WIMP or “natural user interfaces” by unifying various device drivers, frameworks, and tracking toolkits in a common toolkit. It provides a visual design environment for visually connecting input devices to filters (e.g., a Kalman filter or a gesture recognizer) and map them to an output (e.g., Microsoft PowerPoint) (see Figure 1). Squidy offers diverse input modalities such as multi-touch input, pen interaction, speech recognition, laser pointer-, eye- and gesture-tracking. The visual user interface hides the complexity of the technical implementation from the user by providing a simple visual language based on high-level visual data flow programming combined with zoomable user interface concepts. Furthermore, Squidy offers a collection of ready-to-use devices, signal processing filters, and interaction techniques. The trade-off between functionality and simplicity of the user interface is especially addressed by utilizing the concept of semantic zooming which enables dynamic access to more advanced functionality on demand. Thus, developers, as well as interaction designers, are able to adjust the complexity of the Squidy user interface to their current need and knowledge.

![Figure 1. Squidy’s user interface with a pipe-and-filter metaphor to visually connect nodes (filters and input and output devices) (left), a property view per node to change settings during runtime (top-right), and a debug view to inspect current data flow between nodes (bottom-right).](image-url)
exposed from May 17th to May 25th, and around 290,000 people were visiting the Ideenpark. It was used to allow a single user to control a world map application. This application was projected onto a 360-degree panoramic display where users could navigate to particular locations all over the world.

Squidy was also employed during several university classes and courses to allow non-programmers to design everyday widgets (see Figure 2). For example, an advising key holder (top) reports on weather and outside lighting conditions or a TakeCare flower pot (bottom) gives agency to a flower, which expresses feelings like “It is too dark” or “I’m hot.”

A summative evaluation of Squidy showed the applicability also for programmers with little programming experience. It offers a low barrier to entry for beginners with its visual pipe-and-filter metaphor (low threshold [8]), but also enables experienced programmers to implement advanced interaction techniques (high ceiling [8]).

However, the project is no longer maintained. Keeping up-to-date with third-party libraries of supported input and output devices was tedious (technological). Also, build- and continuous integration servers had to be maintained (organizational). The project ended with the end of the research funding and authors had to move on with other research projects (institutional).

HuddleLamp – Open Source and Ongoing

Another example is HuddleLamp [10], which is a desk lamp with an integrated low-cost RGB-D camera that detects and identifies mobile displays (e.g., smartphones or tablets) on tables and tracks their positions and orientations with sub-centimeter precision. Users can add or remove off-the-shelf, web-enabled devices in an ad-hoc fashion without prior installation of custom hardware (e.g., radio modules, IR markers) or software. Because of HuddleLamp’s web-based pairing, adding a new device is simply done by opening an URL on the device and putting it on the table so that it is visible to the camera.

HuddleLamp was demonstrated at ITS 2014 and has been used for research studies (e.g., [11]). Apart from the hybrid sensing pipeline presented in the research paper, HuddleLamp contributes a visual editor to rapidly change the pipeline and test and try out different settings for each processing step (see Figure 3).

Webstrates – Open Source and Highly Active

Webstrates [4] is the most recent toolkit and under active development. It has reached a state beyond prototypical use. Webstrates presents an alternative take on the future of web use and development. In Webstrates, changes to the Document Object Model (DOM) of webpages (called webstrates) are persisted across reloads and synchronized to other clients of the same webstrate. This includes changes to embedded JavaScript and CSS. Webstrates addresses the challenge that while web technologies have become increasingly powerful, technologies such as server-side scripting have turned the web into a sophisticated application platform rather than a user-driven hypermedia system [2] as envisioned by Tim-Berners Lee.

This project is still ongoing and used in research projects, but the public source code is currently not maintained. Unfortunately, Creative does no long manufacture the supported Senz3D camera (technological), which requires implementing a new input node to allow tracking with an alternative camera (e.g., Microsoft Kinect v2).
Webstrates was originally developed to prototype a reiteration of Kay and Goldberg’s vision of interaction with computers as being interaction with personal dynamic media [3], but with an emphasis on shareability. We therefore referred to this reiteration as shareable dynamic media (SDM). The core principles of SDM are “Malleability: users can appropriate their tools and documents in personal and idiosyncratic ways; Shareability: users can collaborate seamlessly on multiple types of data within a document, using their own personalized views and tools; and Distributability: tools and documents can move easily across different devices and platforms.” [4]

Since the original publication Webstrates has gone from being a proof-of-concept to a robust web framework used in multiple research projects internationally and by the paper authors for their daily activities (e.g., note taking, lectures, presentations, teaching material, and rapid prototyping).

To become more than a one-time affair, a full-time developer implements new features and maintains the quality of the code. The development is covered by research funding and costs approximately $70,000 per year. The paper authors also dedicate a significant portion of their time to advance Webstrates and to define future directions together with the full-time developer.

Potential Measure of HCI Toolkits

In contrast to Squidy and HuddleLamp, the public interest in Webstrates and its community of users is growing. At the time of writing, the GitHub repository has eight forks, and more than 50 people bookmarked it. Of course, GitHub forks and bookmarks should not be stressed too much as a reliable measure of the success of Webstrates. But unlike commercially sold hardware and software and their success measured by a company’s revenues, it is difficult to quantify the success of an HCI toolkit. Measuring a toolkit’s success could be based on a jury assessing it according to pre-defined metrics (e.g., generalizability to other application areas) or ranking it by how often it is used in research prototypes.

CONCLUSION

As argued in this position paper, we believe that the viability of HCI toolkits developed in research is largely constrained by the three presented challenges: organizational, institutional, and technological. However, we also believe in the power of HCI toolkits. They serve as factual manifestations of the advancement of socio-technical systems and help the HCI community (and industry) to crystallize a shared vision for HCI, and herewith we are emphasizing on Duval’s statement at the beginning of this position paper.

In the workshop, we would like to share our experiences in building HCI toolkits and discuss the presented challenges with participants. We would further like to spark discussion around the legacy of HCI toolkits. How can we, as HCI research community, preserve the genealogy of HCI toolkits? Does it make sense to start an encyclopedia of HCI toolkits that answer questions like (i) What does a particular toolkit do?, (ii) What did the authors inspire to create it?, and (iii) How did it push the field forward?

REFERENCES