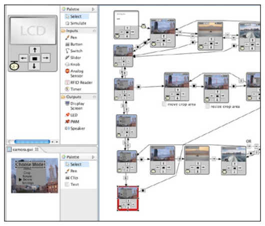
BJÖRN HARTMANN RESEARCH STATEMENT

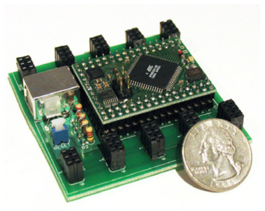
The progress of any creative discipline changes significantly with the quality of the tools available. As the diversity of user interfaces multiplies in the shift away from personal desktop computing, yesterday's tools and concepts are insufficient to serve the designers of tomorrow's interfaces. My research in human-computer interaction focuses on the design, implementation, and evaluation of authoring environments for novel user interfaces. My dissertation research is concerned with the earliest stages in UI creation — activities that take a novel idea and transform it into a concrete, interactive artifact that can be experienced, tested, and compared against other ideas. The dissertation addresses two research questions: How can tools enable a wider range of designers to *create functional prototypes* of ubiquitous computing interfaces? And how can design tools support the larger process of *learning from these prototypes*?

IMPROVING HOW PROTOTYPES ARE BUILT

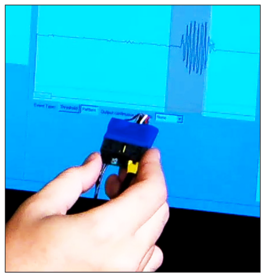
Prototyping is the fundamental activity that structures innovation in design. While prototyping tools are now common for graphical user interfaces on personal computers, prototyping interactions off the desktop remains out of reach for interaction designers. Our fieldwork at professional design companies showed that design generalists lack the tools to fluently experiment with interactions for sensor-based interfaces and information appliances. The first contribution of my dissertation research is a set of methods, embodied in authoring tools, that lower the expertise threshold required to author such novel interfaces. These tools *enable more designers to author a wider range of interfaces, faster.*


The *d.tools* visual authoring environment enables rapid construction of UI logic.

d.tools is a software and hardware toolkit that embodies an iterative-design-centered approach to prototyping information appliances [1]. d.tools enables non-programmers to work with the bits and the atoms of physical user interfaces in concert. Supporting early-stage prototyping through a visual, statechart-based approach, d.tools extends designers' existing storyboarding practices. As designers move from early-stage prototypes to higher fidelity prototypes, d.tools augments visual authoring with scripting. d.tools offers a plug-and-play hardware platform based on smart components that communicate on a shared bus. The architecture exposes extension points for experts to grow the library of supported electronic components.


The *d.tools* hardware interface offers a plug-and-play bus for smart components.

d.tools provides software abstractions for hardware and offers rapid authoring of interaction logic. An additional barrier for practitioners became apparent when we deployed d.tools to an HCI class: students often struggled to transform raw, noisy sensor data into useful high-level events for interaction design. *Exemplar*, an extension to d.tools, bridges the conceptual gap between conceiving of a sensor-based interaction and formally specifying that interaction through programming-by-demonstration [2]. With Exemplar, a designer first demonstrates a sensor-based interaction to the system (*e.g.*, she shakes an accelerometer). The system graphically displays the resulting sensor signals. The designer then marks up


Exemplar combines programming-by-demonstration with direct manipulation to author sensor-based interactions.

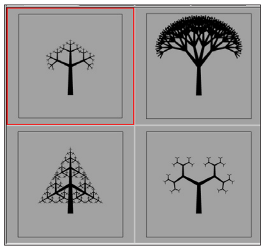
the part of the visualization that corresponds to the action—Exemplar learns appropriate thresholds and patterns from these markups. The designer can review the learned actions through real-time visual feedback and modify recognition parameters through direct manipulation of the visualization.

This evaluation participant used *Exemplar* to control 2D aiming in a game with an accelerometer, and shooting with a flick of a bend sensor.

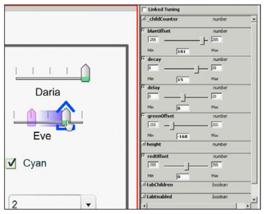
Both d.tools and Exemplar have been evaluated through individual laboratory studies and deployment to interaction design courses and to industry. In a first-use evaluation of Exemplar, participants with little or no prior experience with sensing systems were able to design new motion-based controllers for games in less than 30 minutes. In our collaboration with educational toy company Leapfrog , we provided d.tools hardware schematics and software to Leapfrog's advanced development group. In return, Leapfrog manufactured a complete set of hardware toolkits for us to distribute to a second year of Stanford HCI students. In collaboration with Nokia, we also extended d.tools to author prototype interfaces for mobile devices. I look forward to deepen and grow such partnerships with industry and professional designers in the future.

SUPPORTING WHY PROTOTYPES ARE BUILT

While enabling the construction of prototypes is an important function of design tools, it should not be the only goal. Prototypes are just a means to an end—they are built to elicit feedback about design choices. Today's design tools are largely ignorant of this larger objective. My dissertation research contributes systems that explicitly acknowledge and support the context in which prototypes are built. These accelerate *gaining insight* from prototypes.


Side-by-side execution in *Juxtapose* allows rapid comparison of alternatives.

Exploring alternatives: Designers often create multiple alternative prototypes prior to committing to a direction, because these alternatives provide them with a more complete understanding of a design space; facilitate comparative reasoning; and scaffold stakeholder communication. How might interaction design tools explicitly support creation and management of multiple user interface alternatives? We investigated this question with *Juxtapose,* a code editor and runtime environment for designing multiple alternatives of interaction designs in parallel [3]. Juxtapose offers a textual editor for UIs authored in ActionScript in which interaction designers can define multiple program alternatives through linked editing, a technique to selectively modify source files simultaneously. The set of source alternatives are then compiled into a set of programs that are executed in parallel. Optimizing user experience also often requires trial-and-error search in the parameter space of application variables. To improve this *tuning* practice, Juxtapose generates a control interface for application parameters through source code analysis and language reflection. A summative study of Juxtapose with 18 participants demonstrated that parallel editing and execution are accessible to interaction designers and that designers can leverage these techniques to survey more options, faster. To demonstrate that general principles of working with alternatives carry over into other domavins, we also developed Juxtapose runtime environments for mobile phones and microcontrollers.



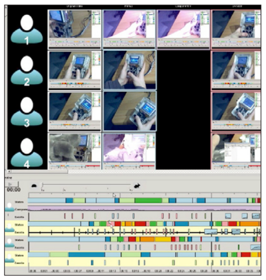
Automatic generation of control interfaces for application parameters enables experimentation with dynamic UI animations.

Juxtapose mobile extends working with alternatives to smart phone applications.

Managing feedback about prototypes: If prototypes are primarily learning vehicles, how can design tools help designers capture and manage feedback? We explored two methods for integrating feedback directly into tools. Many prototypes go through team discussions and reviews before being tested. In word processing, revision management algorithms and interactions techniques effectively enable annotation and asynchronous collaboration over text documents. But no equivalent functionality exists yet for revising interaction designs. *d.note* introduces a revision notation for expressing tentative design proposals within *d.tools* [4]. The tool comprises commands for insertion, deletion, modification and commenting on appearance and behavior of interface prototypes. Based on the insight that changes are often proposed on a higher level of abstraction and ambiguity than concrete logic, d.note realizes three benefits: it visually distinguishes tentative changes to retain design history, allows for Wizard of Oz simulation of proposed functionality, and manages display of alternative design choices to facilitate comparison.

In a first evaluation of *d.note*, participants revised d.tools interaction designs with freeform annotations and semantic revision actions on a pen-driven tablet display.

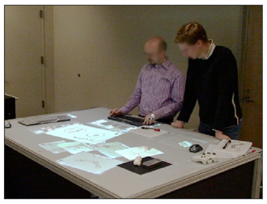
Contextual inquiry with designers showed that prototype test sessions are frequently videotaped, but the hours and days of work required for manual video analysis has limited the practical value of video. The d.tools video suite provides integrated support for testing prototypes with users and rapidly analyzing the results to inform subsequent iteration [1]. d.tools logs all user interactions with a prototype and records an event-synchronized video stream of the user's interaction. The video is automatically structured through state transitions and input events. d.tools provides synchronized video interactions that enable designers to view video and statechart interaction in parallel, visualize time line events as they appear in the statechart, and perform direct manipulation queries to quickly recall, for example, all of the video interactions in a particular state or with a particular control.

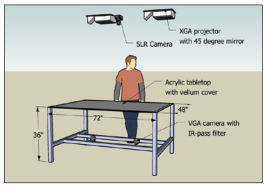

The *d.tools* test & analysis functions connect video clips of test sessions to event traces of the tested prototype.

RESEARCH AGENDA

In future research, I will continue my focus on HCI systems in general and authoring tools in particular. In addition, I am interested in contributing methodologies for tool evaluation. I will continue collaboration with professional designers, and expand my network of academic collaborators across departments.

What will the design studio of the future look like? Digital and physical tools coexist in today's design spaces, but they are largely unaware of each other. What would an enhanced design studio look like that acknowledges the co-presence of digital and physical artifacts and aids designers in working fluidly with both types of material? As a first step toward the design of room-scale environments, I recently constructed a large, multi-person workbench with overhead image capture with collaborators at Microsoft Research [6]. This early work has already generated many opportunities for continued research.





FourBySix is a design workbench constructed with collaborators at Microsoft Research [6].

What would a textual programming language for early prototyping look like? Prototyping tools for experts often take the shape of libraries for a general purpose language such as Python or C++. If prototypes are distinct from 'polished' software, how can a programming language reflect and exploit that difference? How can language design facilitate gathering feedback, modifying applications at runtime, or leaving behaviors partially undefined?

How might people author user interfaces on mobile devices? Tools exist to author applications for mobile devices. But how would one program *on* a phone? For the majority of people outside the western world, a cell phone is the only computing device they are likely to use or own. Given constraints of screen real estate and input technologies, what are the boundaries of authoring on mobile devices? Can we enable millions of interested amateurs to experience the empowerment many of us felt programming our first applications on desktop PCs? I will research how careful editor design, combined with programming by example modification [5] can overcome the limitations of mobile devices.

How should the HCI community evaluate tools research? Close-ended experiments that are appropriate for the evaluation of specific interaction techniques are not easily transferable to design tools. But the summative usability evaluation of systems that is prevalent today often does not contribute to building a theory of design tools. I will investigate methods to evaluate complex authoring environments that lead to more confident generalization of findings beyond the particular interface artifact that was tested.

REFERENCES

[1] **Hartmann, Björn**, Scott R. Klemmer, Michael Bernstein, Leith Abdulla, Brandon Burr, Avi Robinson-Mosher, and Jennifer Gee. Reflective physical prototyping through integrated design, test, and analysis. In Proceedings of UIST 2006: *ACM Symposium on User Interface Software and Technology*. Montreux, Switzerland, 2006. (BEST PAPER AWARD)

[2] **Hartmann, Björn**, Leith Abdulla, Manas Mittal and Scott R. Klemmer. Authoring Sensor Based Interactions Through Direct Manipulation and Pattern Matching. In Proceedings of CHI 2007: *ACM Conference on Human Factors in Computing Systems*. San Jose, CA, 2007. (BEST PAPER AWARD)

[3] **Hartmann, Björn**, Loren Yu, Abel Allison, Yeonsoo Yang, and Scott R. Klemmer. Design as Exploration: Creating Interface Alternatives through Parallel Authoring and Runtime Tuning. In Proceedings of UIST 2008: *ACM Symposium on User Interface Software and Technology*. Monterey, CA, 2008. (BEST STUDENT PAPER AWARD)

[4] **Hartmann, Björn**, Sean Follmer, Anthony Ricciardi, Timothy Cardenas, and Scott R. Klemmer. d.note: Tracking Revisions, Alternatives, and Annotations in Interaction Design Prototypes. *In preparation.*

[5] **Hartmann, Björn**, Leslie Wu, Kevin Collins, and Scott R. Klemmer. Programming by a Sample: Rapidly Creating Web Applications with d.mix. In Proceedings of UIST 2007: *ACM Symposium on User Interface Software and Technology*. Newport, RI, 2007.

[6] **Hartmann, Björn**, Merrie Morris, Hrvoje Benko, Andrew Wilson. Imaging for Imagination: Enabling Design across Physical and Digital Artifacts on an Interactive Workbench. *In submission*.