The Future of Things
How Industrial Designers can Team with Software Designers to Prove the Value of Objects

Tod Corlett, IDSA and Mike Leonard, IDSA
Philadelphia University

Industrial design education is currently presented with an unprecedented challenge. For decades, educators could comfortably assume that the interface of even the most advanced system was primarily a physical interaction; this truism guaranteed industrial designers an important role in the design of that interaction.

With the recent advent of interfaces that are primarily digital and hardware-independent, things have changed. Bits are cheaper than things, and easier to change, and quicker to distribute. To enable a dizzying array of special-purpose software, interaction models like the iPhone/App Store demand objects which don't get in the software's way—ones which are literally as generic as possible. The perfect object, under this model, is one with no features at all, or at least none that aren't determined by software. This puts industrial designers, who generally believe that users benefit from well-designed objects with features that support specific interactions, in a position where they need to earn a place at the table. If industrial designers wish to participate fully in the design of intelligent electronic objects, they have to find new ways to collaborate with software interface designers- ways which can show the true value inherent in physical interfaces.

For five years, collaborative teams of Philadelphia University graduate digital media and industrial design students have been making such proposals in an ongoing series of collaborative projects, finding new ways that objects—and the cognitive, tactile and structural properties that only objects have—can add dimension, nuance and meaning to interaction with information.

This, however, turns out to be a difficult thing to teach people to do, and the initial years of this interdisciplinary project were less than successful. One major problem, interestingly, was "professionalism." Student designers, mostly to their credit, try to be professional. Unfortunately, many of the standard behaviors that define professionalism—privileging efficiency in time, materials and interaction—end up working against the needs of end-users and leading the design team into short-sighted or plagiaristic decisions. These "professionalism traps" were repeatedly encountered during several years of interdisciplinary projects at Philadelphia University. It was eventually understood that the problems weren't going away by themselves, and radical steps had to be taken to counter them. Over several years, a series of corrective strategies were put in place. In response, the project has gone from copying commercial interaction design to leading it in new directions. While the projects themselves are certainly more fascinating than commercial, the process of designing them now equips the students with strategies they can use when trying to keep from repeating the same old tired experience-design clichés.

The rebuilding strategies that have been put in place fall into two general categories:

Strategy 1: Build a focus on users into the project.
The first step here was to be very clear at the outset of the project about the definition of professionalism in the two fields involved, so that attempts at professionalism could take students in a productive direction. Faculty in both fields let the students know, and reiterated often, that the ID and digital design fields are humanistic disciplines. They are primarily about people, and improving their fraught relations with the things and information they use, rather than about making efficient business or engineering decisions.

Having said this, faculty then had to prove their seriousness and give the students a way to act on their principles by building user focus into the project brief itself.

During the unsuccessful early interdisciplinary projects, the project brief was left relatively open. For instance, students were asked to design an innovative voice and text communication system.

To inform their designs, they used standard design-research tools, such as observation of users of current systems, mind maps and the construction of personas. Although the research was generally competent, the results were disappointing; rather than creating something really interesting, the students created things that were extremely real. Smartphone after smartphone was designed, each driven by scrolling menus, each menu controlled by up, down and select buttons or, later, touch-sensitive scroll wheels. It was difficult for the students, having grown up with these conventions, to understand that "desktops," "files" and "menus" were more or less arbitrary metaphors for manipulations of information, and, like other metaphors subject to overuse, they had become clichés. Eventually, it was decided that the project had to begin with a defined design task that took the students outside the realm of anything they might copy, and the project brief was pushed beyond the simply practical. In 2008, for instance, students were told to create transmissible "avatars" of the team members' personalities. For 2009, the students had to design "intelligent surfaces" which could react and respond to human actions. In addition to creating a sense of esprit de corps in battling the shared "enemy" of the brief, the new-style brief puts people, and testing with people, at the center of the team activity, by building this goal into the project itself. There's no more wondering if a design is a good solution; if it performs well in testing with people, it's clearly good, and if it does not it is bad. This makes team decisions easier. Questions that can be tested can be resolved quickly. Questions that hinge only on personal opinion can be wrangled over forever, at great cost to morale and project time.

To give the students space to innovate in service of these new design briefs, they had to be constrained away from easy, prefabricated solutions. To this end, a rule was put in place that mice, keyboards, display screens, and later even video projectors (basically all standard computer input-output devices) were off limits for the project. The students actually have to design and build the entire experience. Although this initially strikes many students as draconian and unfair, the vast majority eventually see the value in having only those pieces of hardware present which actually support the experience.

Strategy 2: Use open-source tools, to show rather than tell.
In the early projects, there were no appropriate tools on hand to actually prototype the systems the students designed, and no engineers to collaborate with. Faculty expected that this would result in some of the designs being unattainable with current technology, as students proposed solutions that were slightly speculative. This was seen as a good thing, in small doses, identifying technologies that could be developed further for near-future products. Instead, even with good technological research by the students, this "paper design" situation resulted in designs that were more, rather than less, technologically conservative than expected. Professionalism, once again, intervened. Can the circuit board fit? Maybe not, make the case larger just in case. Do they make round LCD screens? We've never seen one, so you'd better make it square. Time after time, students professionalized themselves into designs that weren't even reasonably state-of-the-art, much less innovative.

Gradually, it was realized that the students that were the most comfortable with technology were the most willing to innovate and speculate, so it was decided to make everyone more technologically literate. This process began with short lectures on available technologies and predictive strategies such as Moore's Law, but it was the advent of the Arduino development platform and Processing computing language that really gave the students the tools they needed, and Arduino has been used with increasing success for the past three years in this project.

This open-source software-hardware system allows students to actually create their own input and output devices and program their behavior. It's easy to learn, and cheap enough that students don't need to worry excessively about equipment-damaging mistakes. Students are now taught to build, interface and program eight types of sensors and output devices. More are available and acceptable in the project, but students are told that excellent, innovative solutions can be created with only the techniques demonstrated in class. A lot of class time goes into these workshops teaching the principles of soldering, programming and basic electronics, but the rewards have been commensurate. Student teams who have just built, in a few hours, something

they previously thought was science fiction, are much more willing to extrapolate and give technology the benefit of the doubt than those who think technology is “hard to do.”

In addition, Arduino is a widely-used platform worldwide. This allows students to “shop” for successful projects and programming code on the internet (YouTube is full of public demos), and to adapt, re-use and expand successful strategies used elsewhere. This, in turn, lets the students work fast. Because the software and hardware are open-source, the students can borrow and modify at will without worrying about licensing or infringement, although they are urged to be responsible in acknowledging sources.

In addition to giving the students a new perspective on technology, the most important advantage of open-source tools has been the conversion of this project from an exercise in speculation to one that results in real, built, practical solutions. The two strategies of open-source and user focus support each other, because only quick, effective interactive prototyping tools can allow the creation of designs that users can actually test, and only genuine testing with real users can validate what would otherwise be a guess or a piece of speculative fiction rather than a design. In a project to propose new ways of interacting with machines, the only way to prove out the proposal is to build some new machines and let them interact with people, preferably with non-designers.

In the earlier, pre-open-source years of the project, the final graded event was a presentation by the student teams to faculty and outside critics. This audience of experts was expected to evaluate the quality of the proposed solution, which did not and could not come anywhere near the real world.

In the revised project, every time a team proposes a new solution, or evolves one under development, the students are expected to demonstrate the design with working hardware, and to test its effectiveness with people. The project now ends with an end-of- semester public gallery show in the student center, in which six autonomous, intelligent systems are expected to interact successfully all afternoon with people from across the campus. The student teams are present to observe and, occasionally, to repair, but the projects themselves are the stars, and the judges are the ordinary users who welcome them into the real world.