The Inmates are Still Running the Asylum: How to share a design vision with engineers

Uday Gajendar, Principal Designer
Colin Johnson, UI Architect

Citrix Systems, Inc. 4988 Great America Parkway
Santa Clara, CA 95054 USA

Abstract. This paper presents an approach taken by Citrix to shape a balanced, shared product design effort with engineering. Key points include the rise of hybrid designers skilled in software programming, the use of standard UI components, and collaborative standards council activities. Action items are also noted for interested readers trying to build their own integrated design and development efforts for good software user interfaces.

Keywords: User interface design, user interface engineering, user interface technology

1 Introduction

In his seminal 1999 book *The Inmates are running the asylum*, Alan Cooper made a bold claim that shook the software industry. "The reason for bad software products," he argued, "is that engineers are in control." [1] Because engineers tend to craft interfaces the way they craft code—logically and mechanically—software was invariably designed *by* engineers, *for* engineers, who gave little thought to users, and what inspires and delights them.

Much has improved since the book's publication. Interaction designers, visual designers, and researchers are increasingly involved in developing software products at many companies. Businesses have noticed the commercial successes of Apple, Salesforce, Netflix, and the like – companies that feature design, and have design-driven product development processes. A cursory look at recent IxDA or CHI job postings reveals an increased demand for hiring designers. At conferences and in online discussion, user-centered, design-driven product-development approaches are more and more widely accepted.

But there is still a long way to go, partly because of the rapid spread of user interface (UI) technologies that enable a range of outcomes. Software development teams can now choose from a wide variety of technologies to use in building application interfaces. Each technology is good at enabling certain sorts of use experiences, on various devices, while doing specific tasks. For the most part, these decisions are owned by engineering teams, who, all too often, make choices for reasons that have nothing to do with giving users a good experience. Instead, they

select a UI technology because it enables them to showcase "cool" features, because they want to learn to use it, or because time is short and they already know how to use it. The result, too often, is that user experience is an afterthought, and suffers accordingly.

This is particularly true when, as too frequently happens, engineering teams switch UI technologies from project to project. Different technologies enable different sorts of UI behaviors, and different looks. Designers have to scramble to keep up. Often they have to migrate branded assets, and other standard UI elements, to a new, unfamiliar technology framework (from HTML to WPF, for example, or Silverlight to Objective C). Worse, designers sometimes have to re-design everything from scratch, consuming valuable time and resources due to color bit-depth differences or font support limits.

So not only has Cooper's problem not been solved—in some ways, it has become worse. As Cooper points out, engineers are smart, pragmatic, and territorial. It is one thing to persuade an engineer that she should not design a graphical user interface. It's quite another to persuade her that she should not choose the technology used to build a certain interface. Yet this is what software companies must do, to avoid the sorts of problems we have just outlined.

Rather, these and related decisions should be vested in a new sort of product team, one in which developers, designers, and business-side employees collaborate at every step. Moreover, these designers should have strong coding skills. Designers who can code can make informed contributions to team technology decisions, and produce not just static wireframes, but fully coded prototypes that can serve as the basis for development of final products.

This paper will showcase our efforts, at Citrix, to build product teams of this sort. It will also discuss the effects of our decision to staff the Citrix Product Design group with designers who have the hybrid skills needed to partner fully with engineers, to create products that are both technologically advanced, and deliver a top-notch user experience that is consistent across product lines.

2 Explosion of UI Technologies

Fifteen years ago the UI technology choice was simple. Write a Windows application or a simple HTML-based interface for the web browser. Now, product teams have a dizzying array of choices. Consider the list on this Wikipedia page, which shows the variety of frameworks available for building web-based applications. There are also myriad component toolkits available that are independent of the underlying application framework and commonly run on top of them. Examples include ExtGWT or SmartGWT on top of GWT, jQuery or Prototype on top of Ruby on Rails, and Infragistics or Telerik on top of WPF Silverlight.

Each such technology makes it relatively easy to enable a particular look and feel and behaviors, and difficult to enable others. Unfortunately, at many software companies, such as Oracle, Cisco, and Adobe, HCI-trained designers are rarely or never involved in choosing the UI technology. As a result, they have little or no

impact on a decision that radically constrains their ability to ensure that their designs, and their user experience vision, are realized in the final product.

3 Towards a New Kind of Product Design Team

At Citrix, the Product Design group currently supports a relatively narrow range of products, with most intended to be used by IT personnel. And most are used on Windows-based desktops. But this is changing. We are building more and more applications for consumer users, and for use on a variety of OS and device platforms, including smartphones and tablets.

To make sure these applications deliver a high-quality experience that is consistent across OSs and devices, it's essential for our group to cooperate with Citrix's engineering team at every step. A critical part of this is playing a role in selecting the appropriate UI technology to be used in each product. How do we do this?

3.1 Ensuring an Optimal UI via Components & Standards

Our group has set up a team that is developing, and making available to product teams, pre-themed branded instantiations of common UI components, for a select set of UI technologies and platforms. Previously, engineering teams had to implement designs from wireframes and specs created by designers. Under the new system, engineers can create products from pre-built components. This relieves both them and designers of much of the grunt work involved in crafting a UI.

This effort would not be worthwhile, however, if each component were not designed and built to be optimally functional, deliver high-quality interactions, and look, feel, and act like other Citrix applications. Fortunately, our component-production process ensures that this is the case.

Components team composition is a key to the success of this effort. Critically, Product Design UI thought leaders are involved in every step of the component creation process. A designer is of course involved every step of the way to ensure total user experience quality. Full-time team members include a Product Design Group UI architect, who sets the vision and direction and initially acts as the development lead for components. The architect writes and maintains the architecture specification for each component, and writes and reviews functional and design specifications. The architect also creates components and collaborates with part-time development leads in defining common component architecture, and defining services that work with the components. There are full-time component developers who work on creating components and full-time QA personnel who test them.

The team also includes a project manager and a product manager. The project manager ensures that everything runs smoothly, facilitates discussions between teams, and so forth. The project manager also maintains an up to date list of all components and services currently supported and a roadmap for future work with estimated "drop" dates. The product manager writes and maintains the Product Requirements Document (PRD) setting the overall requirements the component library needs to

meet. The product manager also fields feature requests for custom components and feeds them in as requirements.

On the part-time side, UI developers float in and out on an as-needed and as-available basis. They work on component creation as well as specification and code reviews. Others can get involved as they have time and the inclination, because the components effort is run like an open source project. They can write and test components, help with documentation, and build sample applications. If the components library is lacking a particular component, and the components team can't build it in time to be included in a particular product release, the product team is encouraged to write it and later lobby to have it included in the library.

3.2 A Demo Car Built from UI Components

We need not just build components, but ensure their adoption by product teams, and give guidance on how they can and should be used. One way we're doing this is by building and publicizing a "demo car" – a sample application whose purpose is to showcase a set of complex yet widely useful components from the component library. It will have a generic look and feel in line with the desired look and feel of all our applications. All components will be used in the intended fashion and display archetypal interactions. Accompanying the "demo car" will be, for each component, a set of Functional Specifications that detail the public API, and the associated unit tests, and a set of Design Specifications that detail how the component is put together internally. Eventually, the parent UI pattern behind each component will be housed in a linked pattern library. Each pattern definition will include a link to the component, as well as code samples showing how to instantiate and use it.

3.3 Team Activities

Other activities encourage other groups across Citrix to partner with the Product Design group, and ensure the user-experience considerations shape product development at every turn.

3.3.1 Design Process Roadshow

The authors and other design directors recently traveled to Citrix branch offices in Cambridge UK, Bangalore, and Sydney, to advocate in person for the use of common UI components and design patterns, and give updates on the progress of efforts to develop both for use in building applications.

They also educated teams on Citrix's newly rolled out UCD Design Process, which mandates that user needs be clearly identified and taken into full account at every stage, and ensures close, continual collaboration between designers and engineers. Key to both is the mandate that each for each product, a detailed "design brief" be developed and adhered to. Each such brief captures technical as well as user goals. Throughout the design process, it serves as a means of ensuring that both sets of goals

are met by the product design. It also helps prevent confusion, later in the course of product development, about what exactly is to be built and what tools are to be used.

3.3.2 Standards Council Meetings

The new Standards Council exists to define and approve standards and standard UI patterns and components. It also disseminates standards, particularly to external partners, including product managers, QA and customer support personnel, and technical documentation leaders. Emphasis is placed upon "soft enforcement" through partnerships and constructive reviews, not adamant policing with strictly defined scorecards. Any product team resistance is channeled up through our executive support chain, including the VP of Product Design. Weekly Standards Council meetings also serves as a valuable opportunity to survey what is working and what isn't, from a standards perspective, to review new UI requests, communicate UI updates to teams, discuss revised requirements, and so forth. In this way, the Standards Council ensures that standards and components will be used in all UI design and development work, across Citrix.

3.3.3 Tiger Teams

These are temporary, dedicated teams of visual & UI designers, each of which resolves tactical UI implementation issues for products of a similar genre (administrative monitoring consoles, for example). Each team also works with the Standards Council to help the Council extend and revise UI standards to cover cases of the sort it works on. As UI guidelines become more complete, mature, and propagated, the Tigers Teams will work only on emergencies, at executive request.

4 Towards a New Kind of Designer

Our designers do not rely solely on management's help to make this new system work. They also know how to work constructively with engineers, to ensure that their relationship is one of useful collaboration, and meaningful understanding. Particularly valuable here is their level of technical knowledge – at Citrix, we prioritize hiring designers who can code.

We believe that software designers can no longer get by on their right-brain skills alone. They need not just to mock up attractive interfaces, but also code well enough to build interactive prototypes that show exactly how an application should work. They also need a deep understanding of engineering practices, in order to understand the engineering implications of the design choices they make. By developing these skills, they'll be able to more productively collaborate with engineers, both when working in the sort of cross-disciplinary team we discuss above, and in traditional software environments.

Specifically a hybrid designer should be able to:

- Critically evaluate UI technologies and discuss their merits with engineers
- Translate interface layouts into functional prototypes with clean, modular scripts and code snippets that can be reused for subsequent builds
- Effectively and constructively call out an engineer when that person resists design-driven change to a product or process
- Be able to argue back effectively and generate workaround design alternatives that map to the engineer's technical concerns.

Ultimately HCI professionals need to be more aware of the **capabilities** of a chosen technology and how this affects their design. For instance:

- Can the chosen technology be re-skinned and styled to get the desired design?
- Can it be made responsive enough in the environment we intend to use it?
- When the engineering team says it cannot be done, do you believe them? Are you able to suggest an feasible alternative?
- Is it a problem with their existing architecture than the architecture of the technology? In other words is it really bad design on their part and would it mean a lot of rework?

The thing to remember is that until recently, GUI developers were treated like second-class citizens. Real developers worked on the server and server work took priority. The designer needs to be sufficiently tech savvy to know when they are being deceived, and when they are not.

It's becoming easier for design groups to follow our lead, and hire designers with solid technical skills, and ideally also a background in HCI methods for user analysis and testing. Many kids design school students are already competent coders. Many engineering students are already aware of design and some are good designers. This kind of cross-disciplinary dexterity is exceptionally valuable in software design – think of its usefulness, for example, in discussions with engineers about creating web-based UI dashboards that are nimble, adaptive, and smartly designed.

Also invaluable: designers being willing to mentor developers who want to learn about design. At Citrix, our designers actively do this, inviting developers to join our integrated team on a part-time basis, helping to build components and product designs.

Through hiring, mentoring, and advocacy we are enabling the necessary and frankly inevitable rise of hybrid designers. The benefits are overwhelming: smarter and effective collaboration with engineering, more assurance of building improved products that match to the intended design specs, earning trust & respect with team partners that learn to adjust and improvise as issues arise, thus making the project much more satisfying and fulfilling when the release goals are achieved. And this approach boosts morale and overall improves product quality, thus elevating "design" in external team's eyes.

5 Conclusion

So are the inmates still running the asylum? At many companies, yes. In particular, engineers continue to dominate software product creation by controlling UI technology decisions, and, too often, by making these decisions for reasons that have little or nothing to do with the goal of giving users a great experience. As mentioned above, this is a pervasive problem in the software industry, thanks in part to the explosion in the number of available UI technologies.

At Citrix, we have dealt with this and related problems not simply by taking this decision out of the hands of engineers. Rather, we have taken steps systematically to broaden the role of designers in the product-creation process, while ensuring that they will collaborate with engineers and other stakeholders on a continual basis. We have also sought to ensure that this collaboration will be fruitful, by hiring designers who can code, and by actively seeking opportunities to work, formally and informally, with engineers who want to learn more about design.

We are very much still in the middle of this effort. But already, we see that the benefits of our approach have been noticeable and positive:

- Alignment of design and engineering activities with overall product development cycles
- General acceptance of a user-centered design process that has "teeth" when it comes to revisions or pushback by engineering
- Designers have an enhanced sense of influence, and a feeling of being more respected by their colleagues, particularly engineers

More generally, we have built a stronger, more collaborative relationship with Citrix's engineers. Both parties are now more trusting of one another, are better able to make adjustments on the fly, and feel they can achieve their own goals while helping others do likewise. The result is not just better morale, but also, critically, products that are better because they more closely match design specs, while maintaining the technical sophistication Citrix is known for. And "design" is now known, within Citrix, as much more than pretty window-dressing – as, indeed, a critical element of any effort to make great products. Obviously this has not happened overnight, and we still have much to do. But we are very hopeful that we will continue to see success in this area.

Action Items

For readers facing similar situations, what are some next steps you can take to start the process of balancing ownership of UI technology and forming equal respected partnerships with engineering?

- Start reading up and learning popular UI technologies and tools; make it a quarterly goal with 20 % of your time dedicated to this. (like Google's famous 20% time to do your own projects, which works out to be a single day per week)
- Learn the platform UI guidelines. It really helps when you need to back up what you are telling an engineering team when you can turn to published guidelines from Microsoft or Apple!

- Participate (or at least listen in) on engineering discussions, noting key concepts and phrases and issues. Tap your social network on people and resources that can help you advance your learning.
- Try out sample projects, taking your own designs and making them interactive with technologies like Flash or Silverlight or AIR. Guides, tutorials, and videos are online.
- Attend developer conferences like Microsoft MIX or Adobe MX which often hold hands-on training sessions with sample projects. Map those lessons to what you're doing at work, and take it further.

References

1. Cooper, Alan. The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How to Restore the Sanity. Sams Pearson Eduction (2004)