Agile UX Design and Innovation with the 10:3:1 Process

Andrew Bragdon @AndrewBragdon

Peter Provost @pprovost
Background
Nathan, to start a program just click on it.
To see the programs in this room, hold down the F1 key.

Other options
How does bad UX happen?
It’s a process problem

- Nobody wants to create bad UX
- The problem is *how* the UX is designed
Typical UX Design Process

- Starting Design
- Feedback & Iteration
- Final Design
Typical UX Design Process

Starting Design

Feedback & Iteration

Final Design

One design that is iterated on
This doesn’t work
It’s a process problem

- The first idea/most obvious idea is usually not even close to optimal
- PMs and engineers often gravitate towards a single solution very early/immediately
- The best idea may have some initial issues/problems that have to be solved to make it possible - causing it to get ruled out early
10:3:1

- Has roots in general design - art, architecture
- Used at Apple
Building 10 Designs

- Go with the right level of fidelity
  - Don’t do a napkin sketch
  - High fidelity often works well
- Think through interaction flows, not just screens
- Choose 10 different designs that adequately explore the design space
- Include risky and low-risk designs
- Try to make all 10 great
10:3:1 Process

Feedback, User Testing & Iteration
Going from 10 to 3
- Get feedback from team, stakeholders, and users
  - Feedback sessions (internal)
  - User studies early-and-often
- Prioritize the 3 best ones - it will usually be clear which ones are the top 3

Improve the 3
- Take feedback that was positive about the other 7 and incorporate it, along with feedback about the 3
10:3:1 Process

Feedback, User Testing & Iteration
Going from 3 to 1

- Get feedback from team, stakeholders, and users
  - Feedback sessions (internal)
  - User studies early-and-often
- Prioritize the best overall design - it will usually be clear which one is best
10:3:1 Process
Why does this work?

○ Framework for innovation
  ○ Not significantly more costly to do
  ○ Many initial problems can be solved
  ○ Framework for early-and-often feedback/iteration

○ Deep exploration of the design space
  ○ Deeply explore ideas that may at first seem bad/easy to dismiss
  ○ Obvious design may not actually be good (indeed, often is not)
  ○ Ideas from discarded designs make their way into final design

○ Have confidence that the design you ended up with is better than the alternatives
  ○ Global optimum vs. local optimum
  ○ Evolution - survival of the fittest
Example: CodeLens
Background

- ~25 engineers working on developer productivity
- 1 designer
- 1 quarter of investment to prove concept
CodeLens

- Started with 50 alternative designs
- Winnowed down to 10
- Finally winnowed down to 1
- Early-and-often user testing throughout
Mapping the Design Space of CodeLens
```csharp
public Order2(Customer customer, string orderNumber) {
    ValidateCustomer(customer);
    this.UserName = cust;
    AddCartItemsToOrder(customer.Cart);
    this.ShippingAmount = customer.Cart.ShippingAmount;
    this.ShippingAddress = customer.Cart.ShippingAddress;
    this.ShippingService = customer.Cart.ShippingService;
    this.OrderNumber = orderNumber;
    this.ID = Guid.NewGuid();

    // new up the lists
    Items = new LazyList<OrderLine>();
    Transactions = new LazyList<Transaction>();

    this.DiscountAmount = 0;
    this.DiscountReason = "--";
    this.DateCreated = DateTime.Now;
    this.CurrentState = new NewOrder(this);
}

void AddCartItemsToOrder(ShoppingCart cart) {
    foreach (ShoppingCartItem item in cart.Items) {
        AddLineItem(item);
    }
}

public void AddLineItem(ShoppingCartItem cartItem) {
    if (cartItem.Product == null)
        throw new InvalidOperationException(Messages.NullCartSelectedItemProduct);
```
/// <summary>
/// Removes all items from cart
/// </summary>

public void ClearItems()
{
    // Check for null
    if (Items != null)
    {
        // Remove items from the shopping cart
        Items.Clear();

        // Re-initialize the shopping cart
        Items = new List<ShoppingCartItem>();
    }
}
public void AddLineItem(ShoppingCartItem cartItem) {
    if (cartItem.Product == null)
        throw new InvalidOperationException(Messages.NullCartItemProduct);

    Product item = cartItem.Product;

    if (CurrentState.CanChangeItems) {
        OrderLine lineItem = new OrderLine(cartItem.DateAdded, cartItem.Quantity, item);
        this.Items.Add(lineItem);
    }
}
public void ClearItems()
{
    // Check for null
    if (Items != null)
    {
        // Remove items from the shopping cart
        Items.Clear();

        // Re-initialize the shopping cart
        Items = new List<ShoppingCartItem>();
    }
}
public void ClearItems()
{
    // Check for null
    if (Items != null)
    {
        // Remove items from the shopping cart
        Items.Clear();

        // Re-initialize the shopping cart
        Items = new List<ShoppingCartItem>();
    }
}
public void ClearItems()
{
    // Check for null
    if (Items != null)
    {
        // Remove items from the shopping cart
        Items.Clear();

        // Re-initialize the shopping cart
        Items = new List<ShoppingCartItem>();
    }
}

public void AddItem(Product product, int quantity, DateTime dateAdded)
{
    // see if this item is in the cart already
    ShoppingCartItem item = FindItem(product);

    if (quantity != 0)
    {
        if (item != null)
        {
            // if the passed in amount is 0, do nothing
            // as we're assuming "add 0 of this item" means
            // do nothing
            if (quantity != 0)
                AdjustQuantity(product, item.Quantity);
        }
    }
}
Feedback

- Strongly positive on inline data about code
- Like textual representations rather than icon representations
- Prefer method-level data to line-level data
- Want to see class-level data
Customer Feedback

**David Wood** @roadz · 1 Aug 2014
Month 7 of #code lens. You will have to pry it out of my lifeless fingers.

**Juan P.** @qjuanp · 16 Jul 2014
Visual Studio 2013 Peaks and CodeLens makes my day! 
blogs.msdn.com/b/zainnab/archive...

**lain Magee** @lainmagee
CodeLens is an awesome feature in #VS2013. Especially with Git integration. Like, “really” awesome!

**Matt Burrell** @burnmatuk
Visual Studio 2013 #CodeLens I’ve missed you!

**Michel Perfetti** @Militch · Aug 4
CodeLens on git repo is just awesome #VS2013Update3

**Howard van Rooijen** @HowardvRooijen · Aug 4
OK CodeLens for Git in VS 2013 Update 3 is pretty cool

**Jerome Cheng** @Ayulln · 10 Dec 2013
VS2013's CodeLens is awesome. Easy access to references is incredibly useful. msdn.microsoft.com/library/dn2692...

**Psy'Aviah** @psy_aviah · 13 Nov 2013
Really like #Vs2013 #code lens & all member info above the methods/properties! Very handy an... twitpic.com/dlf518w

**Brooks Holland** @SQLDBDude · 20 Oct 2013
Code Lense so awesome. You have got to watch this to believe it! channel9.msdn.com/Series/Visual-...
Example: Application Insights
Background

- ~180 engineers working on developer productivity
- 3 designers
Application Insights

- Started with multiple designs, flows and layouts
- Winnowed down to a single design
Northwind Website

ASP.NET MONITORED APPLICATION

Overview

Average response time

- Browser: 122 ms
- Server: 43 ms

Request rate

- Total Requests: 122 ms

Errors

- Total Errors: 67

Availability

- Failed Tests: 67
Response time and load vs. dependencies

DO MY DEPENDENCIES AND RESPONSE TIMES SCALE UNDER LOAD?

**REQUEST RATE**
- 3 rps

**AVG. RESPONSE TIME**
- 3 sec

<table>
<thead>
<tr>
<th>RESOURCE</th>
<th>TYPE</th>
<th>CALLS</th>
<th>WEIGHT</th>
<th>AVG</th>
<th>REQs/SEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grouped Azure blob requests (8)</td>
<td>Azure blob</td>
<td>404</td>
<td>0.65%</td>
<td>35ms</td>
<td>0.0046</td>
</tr>
<tr>
<td>Grouped HTTP requests (2)</td>
<td>HTTP</td>
<td>109</td>
<td>0.57%</td>
<td>116ms</td>
<td>0.0012</td>
</tr>
<tr>
<td>Grouped Sql requests (12)</td>
<td>Sql</td>
<td>75.7k</td>
<td>98.78%</td>
<td>29ms</td>
<td>0.8762</td>
</tr>
</tbody>
</table>

Top 10 slowest requests by issue count

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>AVG</th>
<th>MAX</th>
<th>COUNT</th>
<th>LAST DATE</th>
</tr>
</thead>
</table>

---

Server performance
Feedback

- Like the paired timeline more than the separate charts
- Like a wider view than a taller one
- At-a-glance single overview vs. menu navigation
- Less key charts is better than “too many charts”
- Users wanted customization
Building a Scalable Process at the Organization Level
Challenges with Scale

- De-centralized design can quickly create a product that feels inconsistent, confusing
- Need for team-level agility
- Hard to do 10:3:1 in the context of aggressive timelines, executive pressure, etc.
What’s important

- 10:3:1 process, shielded from “chaos”, designated responsible individual for process
- One UX patterns owner
- Early-and-often user testing & stakeholder feedback during 10:3:1 process
- Sequencing major design work a couple weeks early so that there is time to complete the process
- UX reviews
Use the right process at the right time!
How this plays out

- 80/20
  - 80% of the time teams can use existing patterns
  - 20% of the time teams need a new/updated pattern

- Some features will get built that skirt the process
  (inevitable on a large team, at least at first)
  - For features that deviate in design, UX board walkthroughs
  - For surprise features, UX board walkthroughs
Pitfalls

- **Organizational**
  - Organizational buy-in into process
  - Transparency and feedback into design process
  - People learn by doing

- **Process leadership**
  - Need a process owner
  - Waiting too late to start design process
  - Picking one design too early

- **Anti-patterns**
  - Creating alternative designs to make a favorite design look good
  - Not getting enough feedback/research on alternatives
  - Focusing only on the “easy-to-design” options
Ways to get started

- Pilot with one team first
- Start with something that is a couple of sprints out
  - UX redesign of a website or app
  - New portal or app
  - New feature area
- Do not need a design team to do 10:3:1
Summary
10:3:1 Process Applications

- Works for the design of virtually anything...
  - User experience
    - Visual design
    - Interaction design
  - Trademarks/branding
  - Software architecture/systems design
  - Industrial design
  - Product design
  - Furniture, Architecture...
10:3:1 Process
Recap

- 10:3:1 Design Process
  - Intentional, iterative design process
  - Generalizes to the design of virtually anything (not just UX)
  - Manage the risk inherent in innovation
- Scaling this design process to a large organization
Questions?

Andrew Bragdon @AndrewBragdon   abragdon@twitter.com

Peter Provost @pprovost   peter.provost@microsoft.com