DesigningForAnalytics.com

How to Self-Assess the UI/UX Design of Products Using Analytics

by Brian O'Neill

Thanks for downloading my free guide from designingforanalytics.com. After years of working with a variety of different clients on analytics-driven software products, the display of quantitative data, and dashboards, I've developed a set of axioms you can ask yourself to help you begin evaluating the design of your product. In this guide, "design" refers to comprehensive product design: your product's user interfaces *and* the emergent user experience that occurs as a result of how your software tools are used within a larger context of use that could include offline activities or other online tools. Good product design yields a 10-100x ROI according to IBM. Getting the design and experience right means more customers, and more revenue. Here we go!

Ask yourself…	Improvement Tips and Ways to Explore the Problem
Did you base your design on pre-packaged software, a dashboard template, or a competitor's design?	Templates and libraries aren't necessarily bad, if you have spent the time to ensure the template designs actually satisfy the tasks and goals users have for your product. Most people don't take the time to do this; they start with their data, and try to pipe it into the provided design, assuming that fancy charts and tables will take care of the rest. A dashboard can easily look visually appealing while providing low or no utility, making them potentially good for sales, but terrible as a product. Worse, **many dashboards are focused on reporting quantitative data, but miss the mark if your tool was bought to be used operationally and needs surface outliers, highlight informative data/findings, or bring recommendations to the surface.** Most of the time, the dashboard or report is the beginning of a journey, not the end. A good dashboard demarcates the destinations that need to be visited right now, and provides good routing options for the user. It helps people find the signal in the noise, and surfaces the interesting outliers. And the competition? **Lots of companies–especially those without a mature design department–copy the competition assuming that the competition spent the time to design the right product.** Are you willing to bet your success on whether your competition's customers share the same goals, attitudes and needs as yours, and that they didn't just copy somebody else without any idea as to whether their own design was useful and usable? Here are some other leading indicators of problems inherent with many boxed solutions: ▪ **Large infographics*** – most data-graphics can be significantly shrunk and still provide the same amount of information. Shrinking also enables you to add additional contextual information to improve the story. ▪ **Pie charts or donut charts*** – a lot has been written on this topic, and my brief recommendation is to always use bar charts. They are easier to compare with the eye, and take less space. ▪ **Scaling problems* –** If you aren't controlling for outliers in your data, one wildly large or small outlier value in a data set can end up skewing the histogram chart axes heavily to the point where all the other normal-range values are now obfuscated. ▪ **"Customization" sounds good, but it creates tool time,**

forces the user to make good design choices, and requires a modular design paradigm that is not always effective. Most users aren't designers.

* This is just the surface of many common data visualization problems. Check out Edward Tufte's lecture circuit and books. Tip: get the books free when you attend one of his related workshops (uh…lectures).

Did you try to handle information overload by thinning out the amount of data in your UI?

Information overload is often confused with information density. Overload happens when people cannot make sense of the information. Data density is a good thing, when aligned with need.

Most of the time, *adding* information to the screens will provide more clarity and improve the story.

However, the more data you add, the more critical the design of the information becomes.

Removing information from the screen is only good if the additional information is neither noise nor key evidence that helps contextualize the main theme of the information you're presenting. Remember that every time you force a user to move between screens, or jump from an email to a webpage, the user is being asked to mentally reset their focus; you've interrupted their concentration. "**Drilldown to get details" is not always the best thing to rely on.** In fact, I like to follow the idea that "having to drill down for details is a chore; are the primary user tasks satisfied elegantly without the user having to do this?"

Some of the most important nuggets of useful information you can add to reduce overload and improve clarity are relevant comparisons. (More on using Comparisons on page 5).

As you add density to a screen, **great visual design** is what will enable you to be sure that the primary story the data is supposed to tell is still most present. Wireframes and sketches aren't enough to provide to engineers; the story the design tells can radically change once it is rendered with pixel-level precision. The details of density, and the rendering of that information via the careful choice of fonts, colors, lines, fills, and whitespace can heavily impact the main theme users walk away with.

Finally, don't forget to design with real data, or realistic data, whenever possible. It's not only easier to test against with users, it is much easier to see where the design will "break."

Did you provide useful comparisons when displaying numbers and facts?

When you show any data, especially a number, average, maximum, minimum, median, etc., ask yourself if the visualization answers, "as compared to what?" This is one of the biggest flaws in most dashboards and tools that display quantitative data: they usually don't allow for easy and relevant comparison.

Instead of just showing "7 widgets per month sold," consider providing great comparison. For example:

- Historical trend for this product:
  - "Is our trend good / bad / out of range?"
  - "How are we doing [over some relevant period]?"

- Compare to most relevant benchmark, sibling, category parent, or competitors value:
  - "How does 7 stack up against the competition?"
  - "How does 7 stack up against the average for widgets in this category?"

- Compare to a macro indicator/index:
  - "Was this product really up 10% on its own accord, or was it really that our entire store showed growth this quarter?"

- Is 7 a magic number?
  - "Should I be taking action on something as a result of us having sold 7 widgets per month?"
  - "Did something happen as a result of us reaching 7, and should I be able to compare that information right here?"

Are you surfacing maximum signal (value) in your data, clearly separating it from all the background noise?

Especially true for big-data products and services with operational tools used frequently, **most of the time, analytics software should be surfacing the most critical and interesting tasks, activities, widgets, or items that need a user's attention.** This data should be clearly demarcated from the noise: the information that hasn't shown meaningful change.

A story about this: I've seen many examples where a user or team responsible for monitoring a huge collection of widgets had grown accustomed to looking at massive, multi-page spreadsheets containing 100s of rows of widgets, and they told me that this is how they check on the health of their widget ecosystem to see what widgets needs attention.

In reality, the users' ritual including scanning only 1 column (a leading-indicator or attribute for the widgets), looking for "magic numbers" that were out of expected range and suggested further exploration.

"If any of the widget's have attribute X above 25-30, I would dig into this item in detail. Above 30 is really bad. "

Putting aside the fact that small bar charts or visual indicators would have been much faster than reading 100s of numbers in this column, in this "helpful report," the other columns of data represented noise. Or rather, "good data at the wrong time."

In reality, this user only needed 1 column/attribute to help them find suspicious widgets. The other columns of data were noise, only being useful if there was a reason to dig deeper.

If a user manages 15,000 widgets, your product is a lot more valuable if you can immediately surface the "above 25-30" culprits. Sometimes that means a notification; other times, a dashboard or just a better on-demand reporting UI is the right tactic.

It's even better if your software self-learned and taught users that "25-30" is the magic number range they should care about (don't assume the user knows that 25-30 is really the important range). **If your product contains time-series data and it can analyze and distinguish between normal and abnormal ranges of data for a given data object, you will be in a powerful position to design more useful experiences for customers.**

Is your design a carefully negotiated compromise of the competing interests? Your large, enterprise product is used by several different customer types with competing needs.

You've got **important stakeholders like executives** who need the best data out of your service…infrequently, but beautifully.

You've got **middle managers** responsible for overseeing a chunk of their company's ecosystem and reporting the results up. Your tool is supposed to help them [look good] by being the messengers of incredible information and drivers of informed change.

You have **operationally-oriented users** who perhaps rely on your data every day for operations efficiency, risk protection, and/or troubleshooting. They know the domain intimately, and love the detail available–much more than everyone else.

It is really hard to design well for multiple personas. A compromised UX that grants all these users a sliver of what they need usually leaves you with a product that has a poor UX for everyone, as it wasn't designed to completely help anyone.

One good recipe for identifying user needs and your business priorities is to organize them into a 2x2 matrix of tasks:

- Identify all of the user types (you should be interviewing customers directly to discover these)
- Make sticky notes of all of the users' tasks and goals
- Assign stickies into a 2x2 whiteboard matrix by frequency (x) and customer-value (y).
- Validate the matrix with stakeholders (or repeat the exercise to see if teams are aligned). The goal is to get everyone to agree on what the "high, high" user tasks are. Expect debate. Maybe some fights. That's good.
- If it's not obvious, measure how well your tool solves "high, high" tasks today so you have a benchmark. You'll find UX and usability problems. That's good: you've just been enlightened on where you need to start improving and can track improvement over time.

This is just one recipe. You can replace the X and Y-axes with other attributes such as "build effort" or "alignment with business goals." Just don't leave the users out.

How do you get from the founder's idea to a minimum *valuable* product? You're a startup with some really cool new analytics nobody else has (or even knew they needed), but you don't want to overwhelm users with all the data.

Startups have unique problems; they may be discovering their users and their needs while they're designing and building. Or, they're trying to help sales get their first sales, or show investors they're making progress.

Some of the most common problems I see with most analytics-driven startup clients is that:

- The product team and business are not aligned on a common set of business and user goals achievable in a reasonable amount of time. Get your sprints aligned with your business objectives.
- **The product team is not routinely interviewing users 1-on-1 to discover latent problems, needs, and business opportunities, and worse, they're missing out on the #1 thing one can learn in interviews: the stuff you don't even know to ask about.**
- The software design process is reverse engineered from the available data or IP that engineering is able to collect.
- The design is oriented toward helping salespeople or investment/financing, without factoring in the risk and debt that is being built up in the form of code baggage, usability, and slower release cycles.

Your customers probably aren't really buying your service because they want more analytics and data-driven software to deal with. It's not about your data; it's about useful, actionable information one can derive from that data via a useful, usable display that helps people accomplish their tasks and goals. More "time on site" is usually not an indicator of the value you're providing your customers.

Designing on fact (and not on assumption) means going out and talking to users, 1-on-1, and asking "why" constantly. It's also probably the cheapest form of risk reduction for your engineers: you'll build less "wrong" stuff.

Have you designed your software to provide actionable recommendations or predictions in a useful way? Or, does your product require the user to do most of the data analysis and come up with their own conclusions?

Jared Spool, a UX thought-leader, often talks about "tool-time" and "goal-time." Good software minimizes the former, and maximizes the latter.

My suggestion when designing analytics-driven recommendations is to write out in 1-2 sentences of plain English what a kick-ass recommendation would look if time and money were not a factor. Ex: "MyProduct recommends that you [some verb] because [some event] happened [due to this cause][during this time period]. We predict [prediction] based on [evidence]." Then, work your way backwards from this goalpost design towards a "sentence" that is feasible to algorithmically construct, but still provides good value.

Many engineering and technical companies are afraid to make estimates and predictions, because they often intimately know the flaws and gaps in the algorithm or the analytics. In reality, **most users would probably trade a reduced-accuracy, software-generated recommendation over a tool that requires them to do all of their own data analysis from scratch.**

Here are some tips for improving the UX of any data-driven recommendations your software calculates for users:

- Does your recommendation lead with a conclusion and only gradually provide detail ("data evidence") as needed to support the conclusion?
- Is enough evidence provided to help the user understand your software's confidence levels when
    - They first begin making recommendations or predictions? (You're building trust here).
    - There is significant risk associated with your data being inaccurate? (Your Terms of Service aren't the place to hide this info.)
- Can users input their own data into your system (such as notes on a time-series chart) to either help improve the quality of your recommendation algorithm, or provide a better experience when they get subsequent automated recommendations?
- If you don't have the ability or risk tolerance to deliver software-based recommendations and predictions, have you designed an information-dense UI that reduces the time required for a human to do manual analysis? Have you visualized the quantitative evidence as best as possible? Have you minimized the amount of "ping-ponging" between screens required to help a user derive a conclusion?

Have you designed a killer onboarding and honeymoon experience?

Does your service require an initial period of metrics collection (data warehousing) before it begins to provide customer value? Have you designed the product to work during this "empty" or "honeymoon," state? **Many products are built for the "Nth" day of use, and forget the early experience, or "honeymoon" as I call it.** For analytics products, consider the following recommendations:

- Make sure your UI and intended UX still make sense if you have limited, reduced, or no data. **How will the screens and emails look 5 minutes after I sign up/install the service?** Have you designed and implemented good empty, broken, edge, and limited-data states of your interface components?
- **Is the product smart enough to disable certain features (e.g. messaging/notification or recommendations) until it has collected enough information to work properly?** Does it inform the user that a collection is happening and set expectations as to when the product will begin to operate normally and provide value?
- Does your product take advantage of this "downtime" period to encourage other activities in the product such as
  - Completing registration more thoroughly?
  - Inviting collaborators?
  - Encouraging users to do tedious tool activities that will show reward in the future, such as manual categorization, organization, attaching metadata, or integrating 3rd-party services?

Are you providing your analytics, metrics, or insights at the right time, with context?

"Right information, at the right time." Solid information is half the equation; the other half is delivering it at the right time, in the right context.

- Are you relying on users to just "wander back" to your product, or does the tool notify users when it's time to look at something?
- Is your experience respectful and responsive to real user attributes (when relevant) such as
  - Their current location
  - Their historical usage trends
  - Their usage, UI, or timing preferences
- When sending notifications, is your tool smart enough to batch up, delay, or change pending communication when appropriate? Is it aware that I already got 2 emails this morning?
- When a given feature provides a prediction, recommendation, or notification, is your tool also factoring in other features that may also use these tactics such that it does not misinform, confuse, or provide conflicting information? A good product tells me it will rain tomorrow. A great product tells me it will rain tomorrow, and I should consider rescheduling my 7pm outdoor soccer game.

More About Designing for Analytics:

- **Mailing List**
  Join the designingforanalytics.com mailing list to get updates, more tips, webinars, and podcast announcements at designingforanalytics.com
- **Have questions?**
  brian@designingforanalytics.com | 210.538.4237
- **Twitter**
  @rhythmspice

For more information, please write or call me in beautiful Cambridge, MA, USA:

1.210.538.4237
brian@designingforanalytics.com
Twitter: @rhythmspice